SIAM J. SCI. COMPUT.
Vol. 23, No. 1, pp. 1–18

A CENTRAL-DIFFERENCE SCHEME FOR A PURE STREAM FUNCTION FORMULATION OF INCOMPRESSIBLE VISCOUS FLOW*

RAZ KUPFERMAN†

Abstract. We present a numerical scheme for incompressible viscous flow, formulated as an equation for the stream function. The pure stream function formulation obviates the difficulty associated with vorticity boundary conditions. The resulting biharmonic equation is discretized with a compact scheme and solved with an algebraic multigrid solver. The advection of vorticity is implemented with a high-resolution central scheme that remains stable and accurate in the presence of large gradients. The accuracy and robustness of the method are demonstrated for high Reynolds number flows in a lid-driven cavity.

Key words. incompressible Navier–Stokes equations, vorticity, stream function, biharmonic equation, algebraic multigrid, high-resolution schemes

AMS subject classifications. 65M06, 65M55, 76D05

PII. S1064827500373395

1. Introduction.

The vorticity formulation of the Navier–Stokes equations is a classical starting point for approximation methods, due to the distinguished role of vorticity in high Reynolds number flows. The difficulty with a vorticity formulation is the lack of natural boundary conditions; the no-slip boundary conditions do not have a simple counterpart in terms of vorticity. In the context of computational methods, the problem of vorticity boundary conditions has a long history, dating back to the 30's [22]; it has received much attention within the context of vortex methods [6], and there exists a substantial amount of recent work (see, e.g., Goodrich and Soh [11], Auteri and Quartapelle [3], Anderson and Reider [2], and E and Liu [8, 7]).

Recently, Ben-Artzi, Fishelov, and Trachtenberg have developed a method of vorticity/stream function dynamics [4]. This method uses explicitly the space of functions in which the dynamics take place. Specifically, the stream function dynamics take place in the Sobolev space H_0^2, whereas the vorticity field resides in the image of H_0^2 under the action of the Laplace operator. At the end of every time step a provisional solution is projected back onto the right functional space, in analogy with the projection onto the space of divergence-free velocity fields in the primitive-variable formalism [5]. As a result, no reference to vorticity boundary conditions is needed, and instead, natural boundary conditions are imposed (as an integral part of the dynamics space) on the stream function. The scheme we present in this paper belongs to this category. Like the scheme in [4], it evolves the stream function within the above-mentioned functional space, but rather than using a "predictor-corrector" approach, it does it via implicit time stepping. In both cases, the computational complexity is dominated by the solution of a linear system of biharmonic type.

The advection of vorticity has been implemented using the Kurganov–Tadmor (KT) scheme [16], which was developed in the context of hyperbolic conservation laws; this scheme has been shown to remain accurate and robust in the presence of large

*Received by the editors June 7, 2000; accepted for publication (in revised form) December 4, 2000; published electronically May 10, 2001. This research was supported in part by the LBNL Directed Research and Development Program.

http://www.siam.org/journals/sisc/23-1/37339.html

†Institute of Mathematics, The Hebrew University, Jerusalem 91904 Israel (raz@math.huji.ac.il).

gradients; at the same time, it shares the relative simplicity of the central differencing framework. In addition, the KT scheme has a well-behaved semidiscrete limit, and as a result, time stepping is not tied to the spatial discretization (except for the standard stability requirements). The independence of the spatial and temporal discretizations adds a degree of modularity that may greatly simplify subsequent adaptations and improvements.

The biharmonic viscous term is discretized by means of a compact stencil (see [1]), which simplifies the treatment of boundary conditions. The fourth-order elliptic equation is then solved with an algebraic multigrid (AMG) solver [13]. This technique can be adapted with little modification to more complicated systems and geometries.

In section 2 we present the flow equations in vorticity-stream function formulation and describe the difficulty associated with boundary conditions. In section 3 we investigate a linear model equation, $u_{xxt} - u_{xxxx} = 0$, inspired by the stream function formulation of the Navier–Stokes equations. This system is simple enough to be completely tractable but is still rich enough to capture the issue of boundary conditions. We prove the convergence of an implicit scheme that uses a compact stencil. The convergence is with respect to the H^2 norm, which is the appropriate norm for a variable analogous to the stream function. In section 4 we extend the scheme of section 3 to the Navier–Stokes equations in a two-dimensional bounded domain. In section 5 we present numerical results for a classical benchmark problem: flow in a two-dimensional lid-driven cavity. The method is found to be accurate and robust up to a regime of high Reynolds numbers, in which the flow becomes highly unstable and generates convoluted vorticity patterns. The scheme seems to be able to resolve vorticity patterns almost down to the scale of a single mesh size.

2. The vorticity-stream function formulation.

We consider incompressible viscous flow in a two-dimensional domain. The motion of the fluid is governed by the Navier–Stokes equations,

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\boldsymbol{\nabla})\mathbf{u} &= -\boldsymbol{\nabla} p + \nu\Delta\mathbf{u}, \\ \boldsymbol{\nabla}\cdot\mathbf{u} &= 0, \end{aligned} \tag{2.1}$$

where $\mathbf{u} = \mathbf{u}(\mathbf{x},t) = (u(\mathbf{x},t), v(\mathbf{x},t))$ is the Eulerian velocity field, $p = p(\mathbf{x},t)$ is the pressure, and ν is the kinematic viscosity. In a bounded domain Ω enclosed by rigid walls, the impermeability of the walls and the no-slip condition imply

$$\mathbf{u}(\mathbf{x},t) = \mathbf{U}(\mathbf{x},t), \qquad \mathbf{x}\in\partial\Omega, \quad t>0, \tag{2.2}$$

where $\mathbf{U}$ is the velocity of the wall.

In terms of the vorticity field $\omega = (\boldsymbol{\nabla}\times\mathbf{u})_z = \partial_x v - \partial_y u$, the flow equations read

$$\frac{\partial\omega}{\partial t} + (\mathbf{u}\cdot\boldsymbol{\nabla})\omega = \nu\Delta\omega, \tag{2.3}$$

where $\mathbf{u}$ is obtained from ω through the div-curl relations

$$\begin{aligned} \boldsymbol{\nabla}\cdot\mathbf{u} &= 0, \\ (\boldsymbol{\nabla}\times\mathbf{u})\cdot\hat{z} &= \omega. \end{aligned} \tag{2.4}$$

The divergence condition implies that the flow field is derivable from a scalar stream function, $\psi(\mathbf{x},t)$,

$$\mathbf{u} = \boldsymbol{\nabla}^{\perp}\psi = \left(-\frac{\partial\psi}{\partial y}, \frac{\partial\psi}{\partial x}\right), \tag{2.5}$$

which, substituted into the curl condition, yields the Poisson equation

$$(\boldsymbol{\nabla} \times \mathbf{u}) \cdot \hat{z} = \Delta\psi = \omega. \tag{2.6}$$

Finally, the boundary conditions (2.2) translate into boundary conditions for the stream function

$$\boldsymbol{\nabla}^{\perp}\psi = \mathbf{U}, \qquad \mathbf{x} \in \partial\Omega. \tag{2.7}$$

The set of equations (2.3), (2.5), and (2.6), together with the boundary conditions, (2.7), is known as the vorticity-stream function formulation of the Navier–Stokes equations.

The classical difficulty with the vorticity-stream function formulation is the improper partition of boundary conditions. The presence of a dissipative term in (2.3) requires the specification of boundary conditions for the vorticity, but these are not prescribed explicitly. Vorticity boundary conditions are extremely important from a physical point of view as they represent the mechanism of vorticity generation at the boundary. On the other hand, the Poisson equation (2.6) is overdetermined by both Neumann *and* Dirichlet boundary conditions (2.7).

This difficulty is immediately removed if the vorticity equation (2.3) is interpreted instead as an equation for the stream function

$$\frac{\partial}{\partial t}\Delta\psi + \Big[(\boldsymbol{\nabla}^{\perp}\psi) \cdot \boldsymbol{\nabla}\Big]\,\Delta\psi = \nu\Delta^2\psi. \tag{2.8}$$

This equation contains a biharmonic operator so that the boundary conditions (2.7) are the natural ones with no over- or underdetermination.

3. A linear model equation. The issue of vorticity boundary conditions can be illustrated by considering a simple model equation inspired by (2.8): a fourth-order linear equation for a one-dimensional scalar field $u(x,t)$,

$$\begin{cases} u_{xxt} = u_{xxxx}, & x \in (0,1), \\ u(0,t) = u(1,t) = 0, \\ u_x(0,t) = u_x(1,t) = 0, \\ u(x,0) = u_0(x), \end{cases} \tag{3.1}$$

where subscripts denote differentiation. Here u plays a role analogous to the stream function, and u_{xx} is the analogue of vorticity. We consider homogeneous boundary conditions; inhomogeneous ones are readily reduced to the homogeneous case by a standard change of variables [12].

Equation (3.1) is solvable by standard techniques, and its solution can be represented as a Fourier sine-series

$$\begin{aligned} u(x,t) = \sum_{q=1}^{\infty} a_q \sin(\pi q x)\, e^{-\pi^2 q^2 t} + \int_0^t b_1(s) \sum_{q=1}^{\infty} \sin(\pi q x)\, e^{-\pi^2 q^2 (t-s)}\, ds \\ + \int_0^t b_2(s) \sum_{q=1}^{\infty} (-1)^q \sin(\pi q x)\, e^{-\pi^2 q^2 (t-s)}\, ds, \end{aligned} \tag{3.2}$$

where

$$a_q = \int_0^1 u_0(x)\, \sin(\pi q x)\, dx$$

and $b_1(t)$,$b_2(t)$ are functions to be determined. The sine-series (3.2) automatically satisfies the Dirichlet boundary conditions $u(0,t) = u(1,t) = 0$. The role of the functions $b_1(t)$,$b_2(t)$ is to enforce the Neumann boundary conditions; they are determined implicitly by the conditions $u_x(0,t) = u_x(1,t) = 0$. Note that

$$\lim_{s\to t} \quad \sum_{q=1}^{\infty} \sin(\pi q x)\, e^{-\pi^2 q^2 (t-s)} = \delta(x),$$
$$\lim_{s\to t} \quad \sum_{q=1}^{\infty} (-1)^q \sin(\pi q x)\, e^{-\pi^2 q^2 (t-s)} = \delta(1-x),$$

which means that $b_1(t)$,$b_2(t)$ can be regarded as the strength of point sources that are concentrated on the left and right boundary, respectively; they play a role analogous to vortex sheets in fluid mechanics.

The solution (3.2) can also be expanded in eigenfunctions,

$$u(x,t) = \sum_q \alpha_q \varphi_q(x) e^{\Omega_q t},$$

where the index q runs over a discrete set of wavenumbers, and $\Omega_q = -\pi^2 q^2$ is the corresponding amplification rate. The eigenfunctions, φ_q, divide into two families:

$$\varphi_q^{(1)}(x) = \left[1 - \cos(\pi q x)\right], \tag{3.3}$$

where $q = 2, 4, \dots$, and

$$\varphi_q^{(2)}(x) = (2/\pi q)\sin(\pi q x) - \cos(\pi q x) - 2x + 1, \tag{3.4}$$

where the wavenumbers q are solutions of the transcendental equation

$$\tan\frac{\pi q}{2} = \frac{\pi q}{2}.$$

Such wavenumbers are typical to a system with mixed boundary conditions [12]. The eigensolutions $\varphi_q^{(1)}$ correspond to the case where the boundary terms are identically zero: $b_1(t) = b_2(t) \equiv 0$ (no generation of "vortex sheets").

A natural approach in approximating (3.1) is to view it as an implicit equation for $u_t(x,t)$; this is analogous to the choice of stream function variables in fluid mechanics. For simplicity, it is sufficient to consider schemes that are first-order in time; the generalization to higher-order is straightforward. For example, a backward-Euler scheme reads

$$\frac{u_{xx}^{n+1} - u_{xx}^{n}}{k} = u_{xxxx}^{n+1},$$

where $k = t^{n+1} - t^n$ is the time step interval.

We discretize the unit segment using a regular mesh of $N+1$ points, with the first and last points coinciding with the left and right boundaries, $x_0 = 0$, $x_N = 1$; the mesh spacing is $h = 1/N$. The standard discretizations of second and fourth derivatives involve stencils of three and five points, respectively, which implies that boundary conditions need to be prescribed at two points near each boundary. Alternatively, it is possible to use a compact 3-point stencil by introducing an auxiliary field v that approximates u_x. Following [1], we propose the following scheme:

$$\begin{aligned} D^0\left(v_j^{n+1}-v_j^n\right) &= 12\lambda\left(D^0 v_j^{n+1} - D^+D^-u_j^{n+1}\right), \\ D^0 u_j^{n+1} &= \left(I+\frac{1}{6}h^2D^+D^-\right)v_j^{n+1}, \end{aligned} \qquad j=1,\dots,N-1, \tag{3.5}$$

where D^0,$D^{\pm}$ are the standard central-, forward-, and backward-difference operators, $\lambda = k/h^2$, and $u_0 = u_N = v_0 = v_N = 0$ at the boundary points. Equation (3.5) is a discrete *differential-algebraic* system; the first equation is an evolution equation, whereas the second is a constraint.

To prove that the numerical scheme (3.5) is convergent we first analyze its consistency and stability properties. Convergence follows from a generalization of Lax's theorem.

LEMMA 3.1. *The numerical scheme* (3.5) *is consistent with truncation error* $\tau = O(h^2,k)$.

Proof. Let $u(x,t)$ be a smooth solution of $u_{xxt} = u_{xxxx}$, and let the discrete auxiliary field $v_j(t)$ be defined implicitly by

$$D^0u(x_j,t) = \left(I+\frac{1}{6}h^2D^+D^-\right)v_j(t)$$

with $v_0(t) = v_N(t) = 0$. A Taylor expansion gives

$$\left(I+\frac{1}{6}h^2D^+D^-\right)[u_x(x_j,t)-v_j(t)] = O(h^4),$$

from which we conclude that

$$v_j(t) = u_x(x_j,t) + O(h^4). \tag{3.6}$$

Substituting (3.6) into the first equation in (3.5) and performing another Taylor expansion, we finally obtain

$$D^0\left[v_j(t+k)-v_j(t)\right] = 12\lambda\left[D^0v_j(t+k) - D^+D^-u(x_j,t+k)\right] + k\,\tau_j^n$$

with $\tau_j^n = O(h^2,k)$. □

LEMMA 3.2. *The numerical scheme* (3.5) *is unconditionally stable.*

Proof. It is possible to construct two families of eigenvectors analogous to (3.3), (3.4) that span the space of solutions of (3.5). Specifically,

$$u_j^n = \sum_q \alpha_q \tilde{\varphi}_q(x_j)\,\tilde{\Omega}_q^n,$$

where the eigenvectors that correspond to (3.3) are

$$\tilde{\varphi}_q^{(1)}(x_j) = 1-\cos(\pi q x_j) \tag{3.7}$$

for $q = 2,4,\dots,N-1$, and the eigenvectors that correspond to (3.4) are

$$\tilde{\varphi}_q^{(2)}(x_j) = (2/A_q)\,\sin(\pi q x_j) - \cos(\pi q x_j) - 2x_j + 1 \tag{3.8}$$

with

$$\tan\frac{\pi q}{2} = \frac{A_q}{2}, \qquad A_q = \frac{3\,\sin(\pi q h)}{h\,[2+\cos(\pi q h)]}.$$

The amplification factor $\tilde{\Omega}_q$ is given in both cases by

$$\Omega_q = \left(1 + \frac{4k}{h^2}\tan^2\frac{\pi q h}{2}\right)^{-1}.$$

It is readily verified that (3.7) and (3.8) form a total of $n-1$ eigenvectors and thus span the space of solutions. For all values of q the amplification factor $|\Omega_q|$ is strictly less than one, from which follows that all eigenmodes decay and the scheme is stable. □

THEOREM 3.3. *The numerical scheme* (3.5) *is convergent.*

Proof. The proof is essentially a generalization of Lax's theorem. It relies on the facts that the truncation errors are small (consistency), and that there is no mechanism that amplifies errors (stability).

Let $u(x,t)$ be a smooth solution of (3.1), let u_j^n be a numerical solution of (3.5), and let $e_j^n = u(jh,nk) - u_j^n$ be the global error. By virtue of Lemma 3.1 and the linearity of (3.5), we obtain the equation for the error

$$\begin{aligned} D^0\left(w_j^{n+1} - w_j^n\right) &= 12\lambda\left(D^0 w_j^{n+1} - D^+D^- e_j^{n+1}\right) + k\,\tau_j^n, \\ D^0 e_j^{n+1} &= \left(I + \frac{1}{6}h^2 D^+D^-\right) w_j^{n+1}, \end{aligned} \tag{3.9}$$

where w_j^n is the auxiliary field associated with e_j^n.

We next expand the error in the discrete eigenmodes (3.7), (3.8),

$$e_j^n = \sum_q \alpha_q^n \tilde{\varphi}_q(x_j). \tag{3.10}$$

Substitution of (3.10) into the auxiliary equation in (3.9) gives

$$w_j^n = \sum_q \alpha_q^n \tilde{\psi}_q(x_j), \tag{3.11}$$

where

$$\begin{aligned} \tilde{\psi}_q^{(1)}(x_j) &= A_q\,\sin(\pi q x_j), \\ \tilde{\psi}_q^{(2)}(x_j) &= 2\,\cos(\pi q x_j) + A_q\,\sin(\pi q x_j) - 2 \end{aligned}$$

are the two families of functions that correspond to $\tilde{\varphi}_q^{(1)}$ and $\tilde{\varphi}_q^{(2)}$. Noting that

$$D^0\tilde{\psi}_q(x_j) = \frac{hA_q}{4}\,\frac{\sin(\pi q h)}{\sin^2(\frac{1}{2}\pi q h)}\,D^+D^-\tilde{\varphi}_q(x_j) = \frac{3\,\cos^2(\frac{1}{2}\pi q h)}{1 + 2\,\cos^2(\frac{1}{2}\pi q h)}\,D^+D^-\tilde{\varphi}_q(x_j),$$

we substitute (3.11) into the first equation in (3.9) and obtain after some basic manipulations

$$\sum_q \left(\tilde{\Omega}_q^{-1}\alpha_q^{n+1} - \alpha_q^n\right) D^0\tilde{\psi}_q(x_j) = k\,\tau_j^n, \qquad j = 1,2,\ldots,N-1. \tag{3.12}$$

Now let $\chi_q(x_j) = D^0\tilde{\psi}_q(x_j)$; these functions span the space of grid functions defined over the $N-1$ inner points x_j, $j = 1, 2, \ldots, N-1$. Let $(f, g) = \sum_{j+1}^{N-1} h f_j g_j$ denote the discrete inner product, and let B be the $(N-1) \times (N-1)$ matrix whose entries are $B_{q,q'} = (\chi_q, \chi_{q'})$.

To obtain an explicit recursion relation for the α_q^n, we take the scalar product of (3.12) with $\chi_{q'}$ and invert by multiplying on the left by B^{-1}; thus

$$\alpha_q^{n+1} = \tilde{\Omega}_q \alpha_q^n + k\, \tilde{\Omega}_q \sum_{q'} B^{-1}_{q,q'} (\chi_{q'}, \tau^n),$$

which by the discrete Duhammel principle gives

$$\alpha_q^n = \tilde{\Omega}_q^n \alpha_q^0 + k \sum_{r=0}^{n-1} \tilde{\Omega}_q^{n-r} \sum_{q'} B^{-1}_{q,q'} (\chi_{q'}, \tau^r). \tag{3.13}$$

The first term of the right-hand side represents the amplification of the initial error, whereas the second represents the accumulation of the local truncation errors.

The vector which we are going to estimate is $D^0 w^n = \sum_q \alpha_q^n \chi_q$, which is equivalent to the second derivative of the error e^n. If $\left\| D^0 w^n \right\|_2 \to 0$ as $h, k \to 0$, then the scheme converges in the H^2 norm, which is indeed the relevant norm for u [4]. Using (3.13), the Cauchy–Schwarz inequality, and the fact that $0 < \Omega_q < 1$, we find

$$\begin{aligned} \left\| D^0 w^n \right\|_2^2 &= \sum_{q,q'} \alpha_q^n B_{q,q'} \alpha_{q'}^n \\ &\le \left\| D^0 w^0 \right\|_2^2 + 2\,kn \left\| D^0 w^0 \right\|_2 \|\tau\|_2 + (kn)^2 \|\tau\|_2^2, \end{aligned}$$

where $\|\tau\|_2 = \max_n \|\tau^n\|_2$. We need only the initial conditions to converge at least as $O(h^2)$ to conclude with the aid of Lemma 3.2 that the scheme converges in H^2, and that the order of convergence is $O(h^2, k)$. □

4. The numerical scheme. Inspired by the model equation presented in the previous section, we construct an approximation scheme for (2.8). The temporal and the spatial discretizations are considered separately; this is legitimate when the scheme has a well-behaved semidiscrete limit [16].

4.1. Temporal discretization. Let ψ^n denote the stream function at time t^n. We approximate (2.8) by a discretization that is second-order in time:

$$\begin{aligned} \left(\Delta - \frac{1}{4}\nu k\, \Delta^2\right) \psi^{n+\frac{1}{2}} &= \left(\Delta + \frac{1}{4}\nu k\, \Delta^2\right) \psi^n - \frac{1}{2} k\, \left[(\mathbf{u}\cdot\boldsymbol{\nabla})\omega\right]^n, \\ \left(\Delta - \frac{1}{2}\nu k\, \Delta^2\right) \psi^{n+1} &= \left(\Delta + \frac{1}{2}\nu k\, \Delta^2\right) \psi^n - k\, \left[(\mathbf{u}\cdot\boldsymbol{\nabla})\omega\right]^{n+\frac{1}{2}}; \end{aligned} \tag{4.1}$$

that is, we use Crank–Nicholson for the viscous term and a midpoint rule for the advection term.

4.2. Spatial discretization. We discretize the system on a rectangular grid with fixed mesh spacing, $\Delta x = \Delta y = h$; a generalization to more complicated metrics will be presented elsewhere. The examples below are for a square domain, where the outermost grid points coincide with the boundaries of the system. We assume that at the beginning of each time step we possess second-order approximations for ψ and its first derivatives—the two velocity components—at the grid points (x_i, y_j), which we denote by $\psi_{i,j}$, $u_{i,j}$, and $v_{i,j}$, respectively.

4.2.1. The advection term. We start with the advection term, which describes the conservative transport of vorticity along streamlines. Due to the incompressibility of the flow, it can be written in an equivalent conservative form,

$$(\mathbf{u} \cdot \boldsymbol{\nabla})\omega = \boldsymbol{\nabla} \cdot (\omega \mathbf{u}),$$

where the vector field $\omega\mathbf{u}$ is the vorticity flux.

The numerical analysis of nonlinear advection has been studied extensively in the context of hyperbolic systems of conservation laws (see, e.g., [10, 17]). Considerable effort has been devoted to the construction of so-called high-resolution schemes, which are designed to capture the structure of singularities, such as shocks and rarefaction waves. Although incompressible flows do not form shocks, experience shows that a careful treatment of the advection is still of primary importance in the presence of sharp gradients. Indeed, sharp gradients seem as discontinuities on the scale of a mesh spacing.

Our discretization of the advection term is based on the central-difference scheme introduced by Kurganov and Tadmor (KT) [16]. Central schemes tend to be simpler than their upwind counterpart and can more easily be used and adapted as "black box" solvers. The KT scheme was found to introduce less numerical viscosity than earlier central schemes [19, 14]; its other advantage is that it can be brought to a simple semidiscrete formulation by letting the time step k tend to zero; thus it is possible to consider the spatial discretization independently from the temporal discretization, which can then be implemented by any standard ODE solver.

Conservative schemes are based on an integral representation of the conservation law; the discrete variables represent averages of the conserved quantities—here vorticity—over control cells. Due to conservation, the rate of change of the mean vorticity equals to the integral of the vorticity flux over the cell's boundaries. We take for control cells squares centered at the grid points. Every time step consists of the following steps. (i) Reconstruction of point values from the given cell averages; for a second-order scheme the reconstructed field is piecewise-linear. (ii) Evaluation of the fluxes at the cell's boundaries; because the reconstructed solution might be discontinuous, a careful treatment is necessary. The KT scheme introduces at the cells' interfaces local control volumes of adaptive size over which the discontinuous behavior is integrated; thus, Riemann solvers are avoided. (iii) Update of cell averages by integrating the fluxes using an appropriate quadrature rule.

Specifically, the spatial discretization of $\boldsymbol{\nabla} \cdot (\omega\mathbf{u})$ proceeds as follows.

- Using the standard 5-point Laplacian, we obtain a second-order approximation for the cell-average vorticity,

$$\omega_{i,j} = \Delta_h \psi_{i,j} = \frac{1}{h^2} \left(\psi_{i+1,j} + \psi_{i-1,j} + \psi_{i,j+1} + \psi_{i,j-1} - 4\,\psi_{i,j} \right), \tag{4.2}$$

  valid in all interior cells, $i,j = 1, \ldots, N-1$; our scheme does not require the evaluation of vorticity at boundary cells.
- We proceed with a piecewise-linear reconstruction of the vorticity,

$$\omega(x,y) = \sum_{i,j} \left[\omega_{i,j} + (\omega_x)_{i,j}(x - x_i) + (\omega_y)_{i,j}(y - y_j) \right] \chi_{i,j}(x,y),$$

  where $\chi_{i,j}$ is the indicator function of the (i,j) cell. The numerical slopes, $(\omega_x)_{i,j}$ and $(\omega_y)_{i,j}$, can be evaluated by simple central differencing when the solution is smooth (on the scale of a mesh spacing). Otherwise, nonlinear slope limiters should be used—for example, the min-mod limiters

$$(\omega_x)_{i,j} = \frac{1}{h}\,\text{minmod}\left[\theta\left(\omega_{i+1,j} - \omega_{i,j}\right), \frac{1}{2}\left(\omega_{i+1,j} - \omega_{i-1,j}\right), \theta\left(\omega_{i,j} - \omega_{i-1,j}\right)\right]$$

with $1 < \theta < 2$. In particular, we evaluate the vorticity at the centers of the cells' edges:

$$\begin{aligned}\omega_{i,j}^{E,W} &= \omega_{i,j} \pm \frac{1}{2}h(\omega_x)_{i,j},\\ \omega_{i,j}^{N,S} &= \omega_{i,j} \pm \frac{1}{2}h(\omega_y)_{i,j}\end{aligned}$$

with the superscripts W="west," E="east," S="south," and N="north" referring to the orientations of the four edges.

- We then evaluate the normal velocities on the cells' edges by simple second-order averaging [18]:

$$\begin{aligned}u_{i-\frac{1}{2},j} &= \frac{1}{2}\left(u_{i,j} + u_{i-1,j}\right),\\ v_{i,j-\frac{1}{2}} &= \frac{1}{2}\left(v_{i,j} + v_{i,j-1}\right).\end{aligned}$$

- At each edge we define a numerical flux

$$\begin{aligned}H_{i-\frac{1}{2},j} &= \frac{1}{2}\left(\omega_{i,j}^{W} + \omega_{i-1,j}^{E}\right)u_{i-\frac{1}{2},j} - \frac{1}{2}|a_{i-\frac{1}{2},j}|\left(\omega_{i,j}^{W} - \omega_{i-1,j}^{E}\right),\\ H_{i,j-\frac{1}{2}} &= \frac{1}{2}\left(\omega_{i,j}^{S} + \omega_{i,j-1}^{N}\right)v_{i,j-\frac{1}{2}} - \frac{1}{2}|a_{i,j-\frac{1}{2}}|\left(\omega_{i,j}^{S} - \omega_{i,j-1}^{N}\right),\end{aligned}$$

where the first term on the right-hand side is the average of the one-sided flux evaluations, whereas the second term is a correction that arises from the more precise treatment at the discontinuous boundaries; the prefactors $a_{i-\frac{1}{2},j}$ and $a_{i,j-\frac{1}{2}}$ correspond to the local characteristic speeds at the cells' interfaces, which in our case are simply the normal velocities, u and v, respectively.

- Finally, the divergence of the flux is approximated by

$$[\boldsymbol{\nabla}\cdot(\omega\mathbf{u})]_{i,j} = \frac{H_{i+\frac{1}{2},j} - H_{i-\frac{1}{2},j}}{h} + \frac{H_{i,j+\frac{1}{2}} - H_{i,j-\frac{1}{2}}}{h}.$$

4.2.2. The viscous term. We next address the spatial discretization, Δ_h^2, of the biharmonic operator. The standard second-order discretization uses a 13-point stencil. Noncompact stencils are problematic from the point of view of linear solvers. An alternative representation of the discrete biharmonic operator that uses a compact 9-point stencil was developed by Altas et al. [1], and its one-dimensional version was presented in section 3. The idea is to express $\Delta_h^2\psi_{i,j}$ in terms of the grid values of ψ and its first derivatives, $\psi_x = v$ and $\psi_y = -u$. A second-order approximation of the biharmonic operator can then be written as

$$\Delta_h^2\psi_{i,j} = \frac{12}{h^2}\left(-\frac{4}{3}\Delta_h\psi_{i,j} + \frac{1}{3}\tilde{\Delta}_h\psi_{i,j} + D_x^0 v_{i,j} - D_y^0 u_{i,j}\right), \tag{4.3}$$

where $D^{0;\pm}_{x,y}$ are the standard differencing operators and $\tilde{\Delta}_h$ is the star-Laplacian

$$\tilde{\Delta}_h \psi_{i,j} = \frac{1}{2\,h^2}\left(\psi_{i+1,j+1} + \psi_{i-1,j+1} + \psi_{i+1,j-1} + \psi_{i-1,j-1} - 4\,\psi_{i,j}\right).$$

In addition, we need fourth-order expressions for $u_{i,j}$ and $v_{i,j}$:

$$\begin{aligned} \left(I + \frac{1}{6}h^2 D_x^+ D_x^-\right) v_{i,j} &= +D_x^0 \psi_{i,j}, \\ \left(I + \frac{1}{6}h^2 D_y^+ D_y^-\right) u_{i,j} &= -D_y^0 \psi_{i,j}. \end{aligned} \tag{4.4}$$

These approximations are valid for all interior points; at boundary points ψ, u, and v are prescribed by the boundary conditions.

4.3. The linear solver. From a computational point of view, the most time-consuming part of the computation is the solution of a linear system of the form

$$(\Delta_h - \alpha\,\Delta_h^2)\psi_{i,j} = \text{rhs}_{i,j},$$

which results from the spatial discretization of (4.1); two such linear systems need to be solved at every time step. Standard iterative methods are known to converge very slowly, if at all, for biharmonic operators.

Biharmonic systems that use the compact stencil representation (4.3), (4.4) can be solved very efficiently with AMG solvers. AMG methods are powerful techniques for the solution of sparse linear systems. They are "black box" solvers, in the sense that they treat the problem to be solved as a pure algebraic system, without reference to the geometrical interpretation of the transition between coarse and fine grids. The advantage of such an approach is that it is readily portable to more complicated systems of coordinates and geometries.

The principle of AMG methods can be summarized as follows. Given an n-dimensional linear system $Ax = b$, an $m \times n$ restriction matrix R is generated by an algorithm that inspects the graph of the matrix A. The m-dimensional vector Rx is the projection of x on the restricted ("coarse") subspace. The transpose of the restriction matrix, $I = R^T$, is used as an interpolation matrix to revert back to the original ("fine") space. AMG methods are based on the presumption that if the right-hand side vector, b, is sufficiently "smooth" (in a sense that needs to be specified), then the solution x is close to the range of the interpolation matrix I, i.e., there exists an m-dimensional vector y such that $x \approx Iy$. Multiplying the system $Ax = b$ by R on the left and approximating x by Iy, we obtain the restricted system

$$(RAI)y = Rb.$$

This two-level approach can be applied recursively to form a multilevel method. It then remains to introduce an appropriate "smoother" to operate on the solution before and after being projected to the lower-dimensional subspace.

AMG solvers vary in the way they generate the restriction matrix, in the choice of smoothers, and in the choice of multigrid cycles. Our coarsening method is based on a red-black coloring algorithm developed by Kickinger [13]. Gauss–Seidel iterations have been used for smoothing. Each multigrid cycle starts from the finest level down to the coarsest level and back up (V-cycle). For a 128 × 128 grid (that is, a linear system of dimension 3 × 128 × 128), about ten multigrid cycles with two pre- and postsmoothing steps were needed to reduce the error norm to 10^{-8}.

There are a number of implementational issues. As long as the time step is not modified, the linear operator is unchanged. In such a case, it is efficient to compute the set of restriction matrices, R, with the corresponding linear operators, RAI, once, and store them. Most of the computational time is then spent on sparse matrix-vector multiplications, which can be parallelized easily.

5. Numerical results. We have tested our numerical scheme on a classical benchmark problem: flow in a lid-driven cavity. The fluid is confined in a square domain, $\Omega = [0,1]^2$, and is driven by the transversal motion of its boundaries. This setup is a challenging test problem, in particular, because the velocity field is discontinuous at the corners adjacent to the moving boundaries, and the viscous stresses diverge logarithmically. (In a method that uses the primitive variables $(\mathbf{u}, p)$, one faces the logarithmic divergence of the pressure.)

We first conducted convergence tests to obtain error estimates and assess the order of accuracy. In Table 1 we show the discrete L_2 norm $\|\psi_{N+1} - \psi_{2N+1}\|_2$, where ψ_M denotes the computational solution on an $M \times M$ grid. The initial conditions are

$$\psi(x,y,0) = \frac{1}{\pi} \sin^2(\pi x) \sin^2(\pi y), \tag{5.1}$$

and the boundaries are stationary; the Reynolds number here is 10^3. For short times the convergence rate seems to be less than expected; this is because the errors are very small and therefore dominated by the tolerance specified for the linear solver. For times longer than $t = 0.4$ we get an estimated second-order convergence, as expected. Similar results were found for a range of Reynolds numbers between 10^2 and 10^4.

Table 1

Error estimate and convergence test for the initial conditions (5.1) *and Reynolds number* 10^3.

Time	$\|\psi_{33} - \psi_{65}\|_2$	$\|\psi_{65} - \psi_{129}\|_2$	Rate
0.1	4.1×10^{-5}	1.3×10^{-5}	1.64
0.2	7.2×10^{-5}	2.1×10^{-5}	1.81
0.3	9.5×10^{-5}	2.5×10^{-5}	1.90
0.4	1.2×10^{-4}	2.9×10^{-5}	1.99
0.5	1.4×10^{-4}	3.3×10^{-5}	2.08

We next display results for lid-driven flows. For low enough Reynolds numbers the flow approaches a steady state; the lower the Reynolds number is, the shorter the transient is. In Figure 1(a) we display stream function contour lines in the steady state for Re = 400. The fluid is initially at rest, and it is driven impulsively by the rightward motion of the top boundary. For comparison we display the same level sets as in [9, Figure 3 and Table III]. In Figure 1(b), (c) we plot the steady-state profile of the $u(v)$ component of the velocity as a function of $y(x)$ at $x = 0.5$ ($y = 0.5$). The solid line represents our results, whereas the symbols are data reported in [9]). The agreement is excellent. In Table 2 we list the minimum value of the stream function, which takes place in the core of the primary vortex, at different times and for three different grid sizes. For a grid size of 128 × 128 the results seem to have fully converged. It takes about 35 time units to reach a steady flow; the minimum value of the stream function is then −0.1140; in [9] the reported value is −0.1139; the same value has also been reported by Pan and Glowinski for a slightly regularized flow [20]. We also compare extremal values of the stream function for the secondary

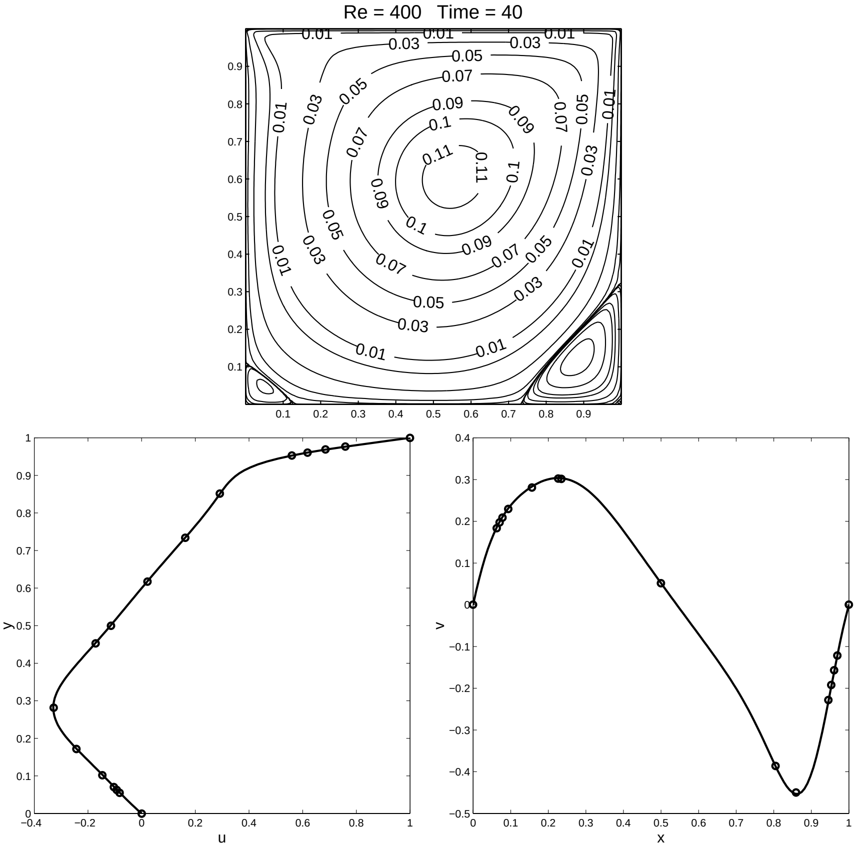


FIG. 1. (a) *Contour plots of the stream function at time $t = 40$ after an impulsive start at Reynolds number $Re = 400$. The top boundary moves to the right with velocity $u = 1$.* (b) *Steady-state profile of the u velocity component as a function of y at $x = 0.5$.* (c) *Steady-state profile of the v velocity component as a function of x at $y = 0.5$.*

TABLE 2

Minimum value of the stream function at different times and for different mesh sizes. The Reynolds number is $Re = 400$.

Time	64×64	96×96	128×128
$t = 5.0$	-0.09062	-0.09074	-0.09076
$t = 15.0$	-0.11164	-0.11173	-0.11174
$t = 25.0$	-0.11378	-0.11385	-0.11385
$t = 35.0$	-0.11393	-0.11400	-0.11401

vortices. For the bottom-right secondary vortex the maximum value of the stream function is 6.579×10^{-4}, and it is 1.404×10^{-5} for the bottom-left vortex; the numbers reported in [9] are 6.423×10^{-4} and 1.419×10^{-5}, respectively. In [11] the time to reach a steady state was estimated to be about 46 time units. A precise quantitative comparison is hard to perform due to the arbitrary nature of the stopping criterion.

Similar calculations were carried out for Re = 5000. In Figure 2(a) we plot the minimum value of the stream function versus time. Steady-state velocity profiles are shown in Figure 2(b), (c). Snapshots of stream function contour lines are presented in Figure 3. Note the much longer transient; its takes about 340 time units to reach

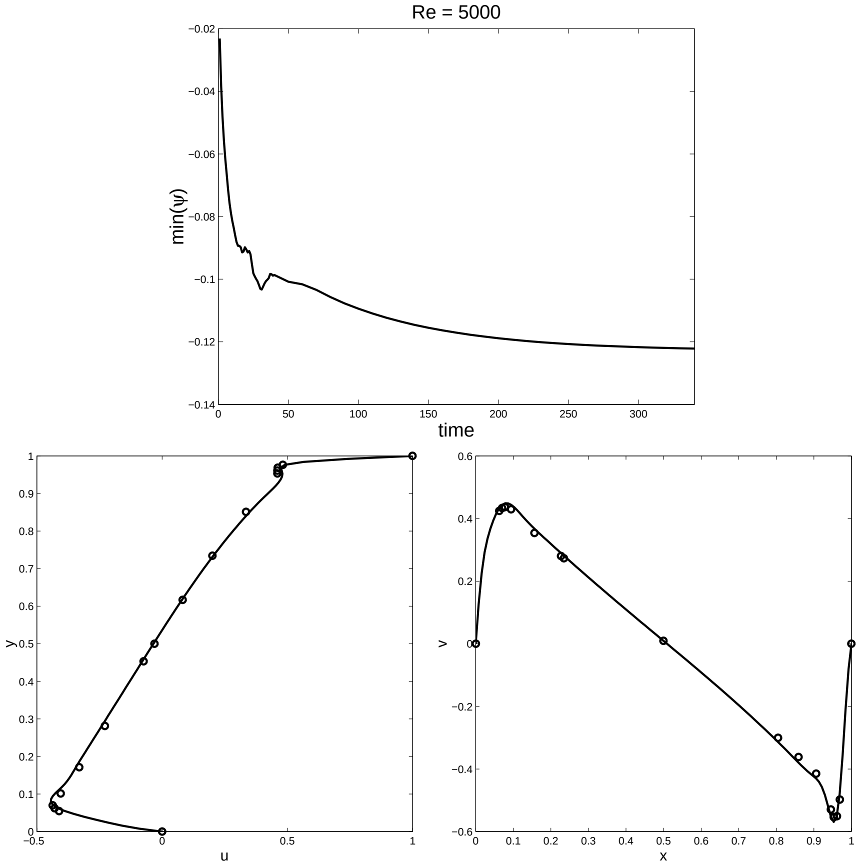


FIG. 2. (a) *Minimum value of the stream function versus time for Re* = 5000. (b) *Steady-state profile of the u velocity component as a function of y at* $x = 0.5$. (c) *Steady-state profile of the v velocity component as a function of x at* $y = 0.5$.

a steady flow. Note also the nonmonotonic behavior of the minimum value of the stream function, which reflects the fact that recirculation zones are created and annihilated along the side and bottom walls until the final vorticity pattern emerges. Such dynamics were also reported in [11] and are consistent with subcritical behavior prior to an oscillatory instability, whose occurrence has been predicted in [21, 20]. Eventually, the stream function reaches the value of -0.122160; in [9] the predicted value is -0.118966, whereas in [20] it is -0.121218. The velocity profiles are again in excellent agreement with the data reported in [9].

In a recent paper, Pan and Glowinski [20] obtained limit cycle solutions for Re = 8500. The occurrence of a Hopf bifurcation has been speculated before [21] but was believed to take place at a significantly higher Reynolds number. Our results for Re = 8500 support the findings of [20]. In Figure 4 we plot the time evolution of the kinetic energy over a time interval of $t = 3$; the function is oscillatory with a period of about 2.5; the period reported in [20] is 2.27. In Figure 5 we show a complete cycle of stream function contours during a time interval of 2.5. The primary vortex remains practically unchanged, variations being noticeable only within the secondary vortices. It takes about 200 time units to reach this state starting from a fluid at rest.

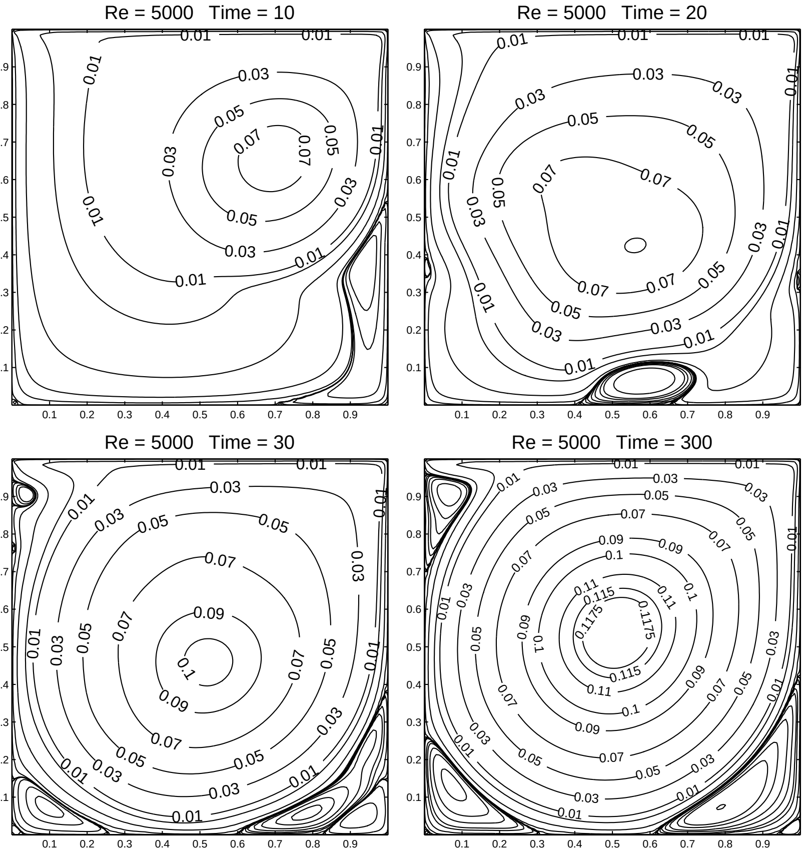


FIG. 3. *Contour plots of the stream function. The top boundary moves to the right with velocity* $u = 1$*. The Reynolds number is* $Re = 5000$*. The contours are shown for time* $t = 10$*,* $t = 20$*,* $t = 30$*, and* $t = 300$*. The grid size is* 128×128*.*

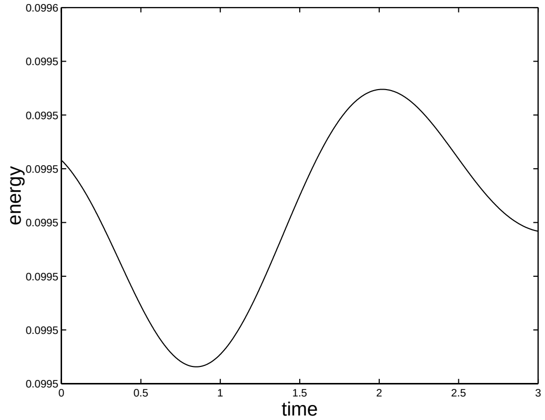


FIG. 4. *Kinetic energy versus time for* $Re = 8500$*.*

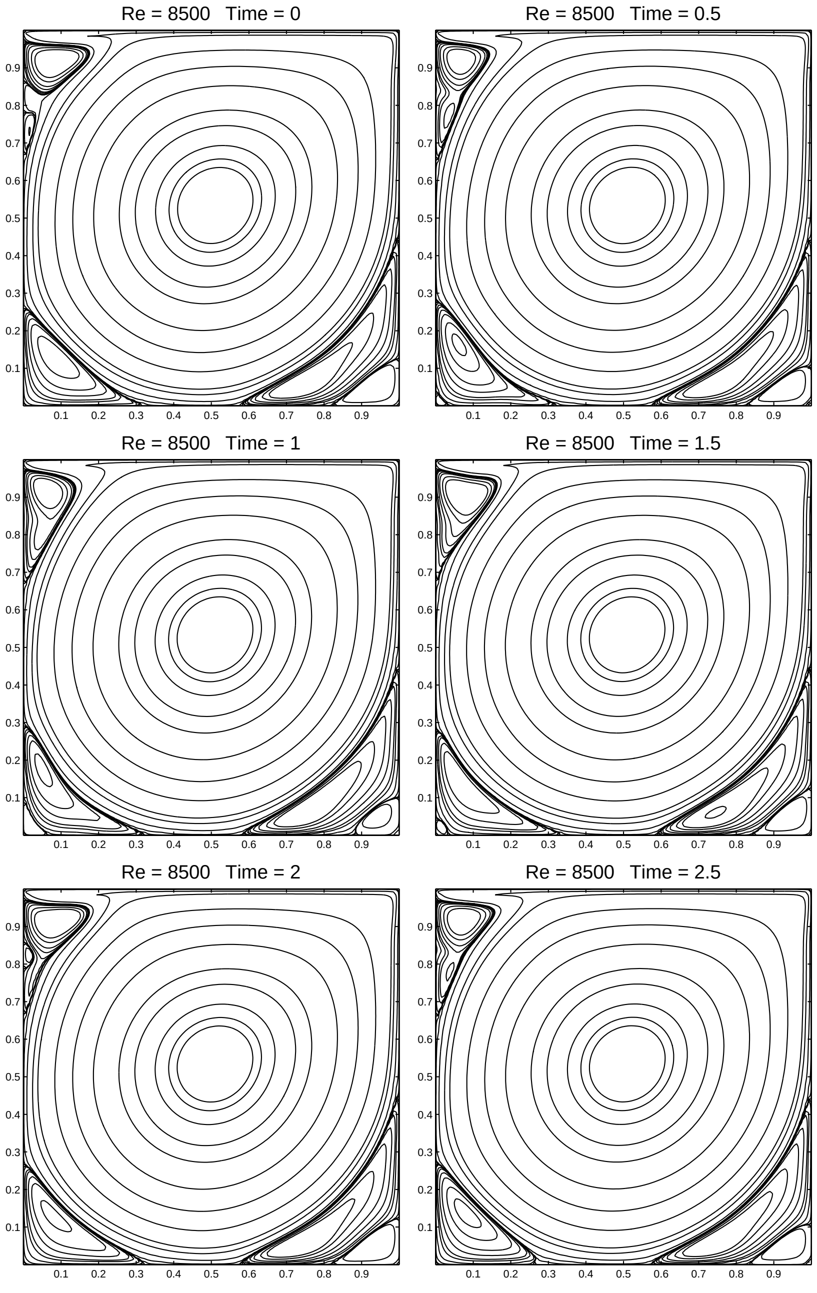
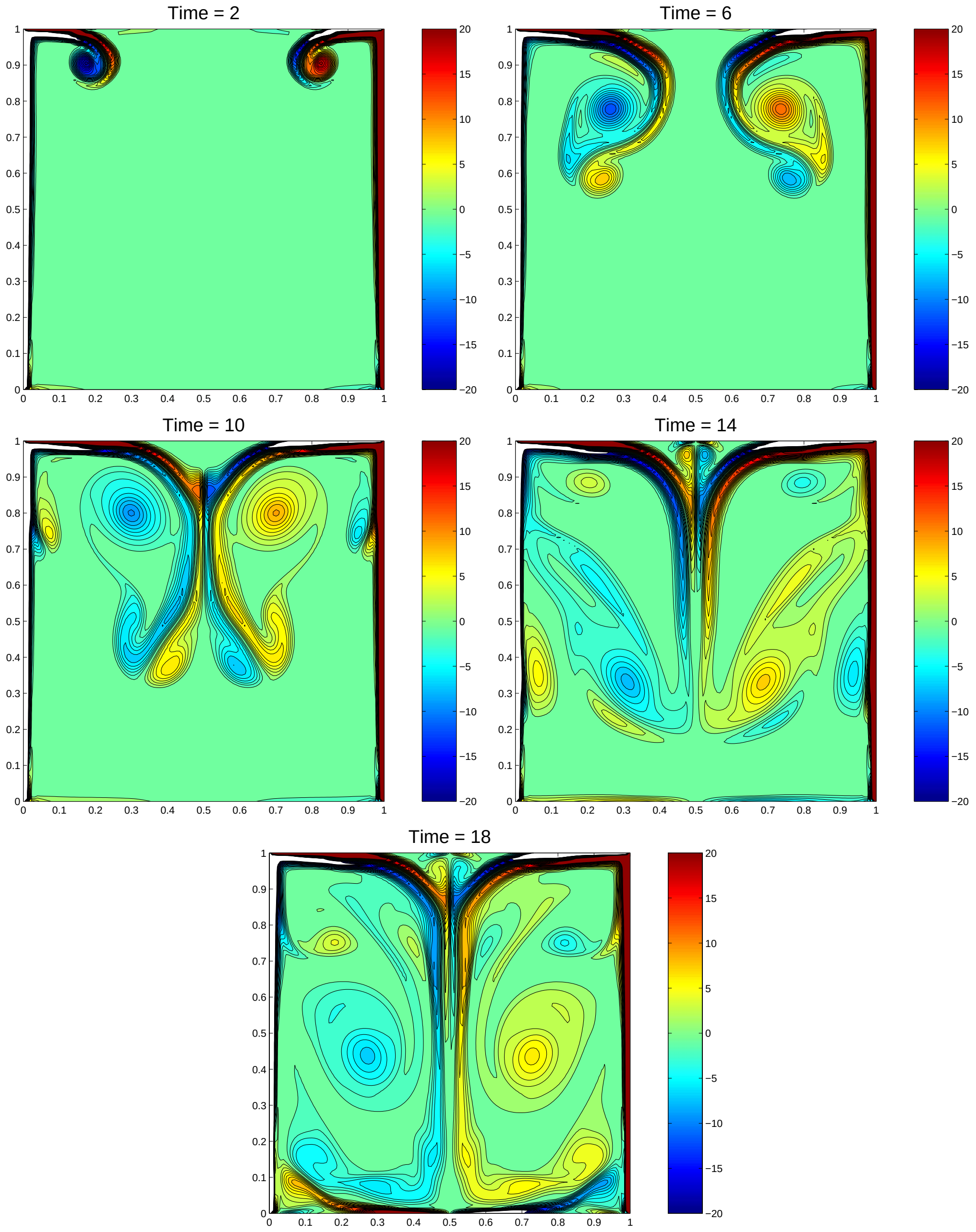


FIG. 5. *Contour plots of the stream function. The top boundary moves to the right with velocity* $u = 1$*. The Reynolds number is* $Re = 8500$*. This sequence covers one period of the limit cycle. The grid size is* 128×128*.*

For even higher Reynolds numbers the flows are much more complex. An Re = 20000 flow is shown in Figure 6, where we display snapshots of vorticity contour lines for a fluid that is driven by the upward motion of its left and right boundaries. Narrow and concentrated shear layers are generated along the moving boundaries and


Time = 2

Time = 6

Time = 10

Time = 14

Time = 18

FIG. 6. *Contour plots of vorticity. The left and right walls move upward with velocity* $v = 1$. *The Reynolds number is* $Re = 20000$. *The contours are shown for time* $t = 2$, $t = 6$, $t = 10$, $t = 14$, *and* $t = 18$. *The grid size is* 128×128.

transported with the flow. The shear layers are unstable and rapidly intermingle to form a convoluted vorticity pattern. This sequence of vorticity contours demonstrates the robustness of the scheme. The vorticity gradients are large with sharp variations over single cells.

6. Concluding remarks. The present scheme is based on the paradigm that has been established in [4], whereby vorticity dynamics should be viewed as a projection of stream function dynamics; thus vorticity boundary conditions are totally avoided, and natural boundary conditions are imposed on the stream function. Numerical methods based on stream function variables are by themselves not a novel idea. What have been missing for many years are accurate and efficient ways of implementation. We make no claim, however, that methods based on vorticity boundary conditions are invalid. Such methods have been proven to work within the frameworks of both difference schemes and vortex methods.

An important property of the proposed scheme is its modularity. It is not restricted to a specific type of hyperbolic or biharmonic solver, and each of its elements can be implemented in various ways. In particular, higher-order spatio-temporal discretizations are relatively easy to implement (see, e.g., [1] for a fourth-order discretization of the biharmonic equation and [15] for a third-order version of the KT scheme). Finally, an extension to three dimensions seems realizable.

Acknowledgments. I am grateful to Professors M. Ben-Artzi, D. Fishelov, and S. Trachtenberg for introducing me to stream function dynamics and for making their work and data accessible. This paper has been motivated and influenced by their work. I have benefited from useful discussions with Professor A. Chorin, Mr. Y. Farjoun, Professor R. Glowinski, Professor O. Hald, Professor E. Tadmor, and Dr. D. von Terzi. Special thanks to Mr. Y. Farjoun, who implemented the multigrid solver.

REFERENCES

[1] I. Altas, J. Dym, M. M. Gupta, and R. P. Manohar, *Multigrid solution of automatically generated high-order discretizations for the biharmonic equation*, SIAM J. Sci. Comput., 19 (1998), pp. 1575–1585.

[2] C. Anderson and M. Reider, *A high order explicit method for the computation of flow about a circular cylinder*, J. Comput. Phys., 125 (1996), pp. 207–224.

[3] F. Auteri and L. Quartapelle, *Galerkin spectral method for the vorticity and stream function equations*, J. Comput. Phys., 149 (1999), pp. 306–332.

[4] M. Ben-Artzi, D. Fishelov, and S. Trachtenberg, *Vorticity dynamics and numerical resolution of the incompressible Navier-Stokes equations*, Math. Model. Numer. Anal., 35 (2001), pp. 313–330.

[5] A. Chorin, *On the convergence of discrete approximations to the Navier-Stokes equations*, Math. Comp., 22 (1969), pp. 745–762.

[6] A. Chorin, *Numerical study of slightly viscous flow*, J. Fluid Mech., 57 (1973), pp. 785–796.

[7] W. E and J.-G. Liu, *Essentially compact schemes for unsteady viscous incompressible flows*, J. Comput. Phys., 126 (1996), pp. 122–138.

[8] W. E and J.-G. Liu, *Vorticity boundary condition and related issues for finite difference schemes*, J. Comput. Phys., 124 (1996), pp. 368–382.

[9] U. Ghia, K. Ghia, and C. Shin, *High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method*, J. Comput. Phys., 48 (1982), pp. 387–411.

[10] E. Godlewski and P.-A. Raviart, *Hyperbolic Systems of Conservation Laws*, Math. Appl. 3–4, Ellipses, Paris, 1991.

[11] J. Goodrich and W. Soh, *Time-dependent viscous incompressible Navier-Stokes equations: The finite difference Galerkin formulation and stream function algorithms*, J. Comput. Phys., 84 (1989), pp. 207–241.

[12] B. Gustafsson, H.-O. Kreiss, and J. Oliger, *Time Dependent Problems and Difference Methods*, John Wiley and Sons, New York, 1995.

[13] F. Kickinger, *Algebraic multigrid for discrete elliptic second-order problems*, in Multigrid Methods, Proceedings of the Fifth European Multigrid Conference, Stuttgart, Germany, Springer-Verlag, Berlin, 1996, pp. 157–172.

[14] R. Kupferman and E. Tadmor, *A fast high-resolution second-order central scheme for incompressible flows*, Proc. Natl. Acad. Sci. USA, 94 (1997), pp. 4848–4852.

[15] A. Kurganov and D. Levy, *Third-order semidiscrete central scheme for conservation laws and convection-diffusion equations*, SIAM J. Sci. Comput., 22 (2000), pp. 1461–1488.

[16] A. Kurganov and E. Tadmor, *New high-resolution central schemes for nonlinear conservation laws and convection-diffusion equations*, J. Comput. Phys., 160 (2000), pp. 214–282.

[17] R. LeVeque, *Numerical Methods for Conservation Laws*, Birkhauser Verlag, Basel, 1992.

[18] D. Levy and E. Tadmor, *Non-oscillatory central schemes for the incompressible* 2*-D Euler equations*, Math. Res. Lett., 4 (1997), pp. 1–20.

[19] H. Nessyahu and E. Tadmor, *Non-oscillatory central differencing for hyperbolic conservation laws*, J. Comput. Phys., 87 (1990), pp. 408–463.

[20] T. Pan and R. Glowinski, *A projection/wave-like equation method for the numerical simulation of incompressible viscous fluid flow modeled by the Navier-Stokes equations*, Comput. Fluid Dyn. J., 9 (2000), pp. 28–42.

[21] J. Shen, *Hopf bifurcation of the unsteady regularized driven cavity flow*, J. Comput. Phys., 95 (1991), pp. 228–245.

[22] A. Thom, *The flow past circular cylinders at low speeds*, Proc. Roy. Soc. London Sect. A, 141 (1933), p. 651.

SIAM J. SCI. COMPUT.
Vol. 23, No. 1, pp. 19–45

TWO-DIMENSIONAL SIMULATIONS OF VALVELESS PUMPING USING THE IMMERSED BOUNDARY METHOD*

EUNOK JUNG† AND CHARLES S. PESKIN‡

Abstract. Flow driven by pumping without valves is examined, motivated by biomedical applications: cardiopulmonary resuscitation (CPR) and the human fetus before the development of the heart valves. The direction of flow inside a loop of tubing which consists of (almost) rigid and flexible parts is investigated when the boundary of one end of the flexible segment is forced periodically in time. Despite the absence of valves, net flow around the loop may appear in these simulations. The magnitude and even the direction of this flow depend on the driving frequency of the periodic forcing.

Key words. valveless pumping, immersed boundary method, frequency, CPR

AMS subject classifications. 76D05, 76Z99, 92-08, 92B99, 92C05, 92C50

PII. S1064827500366094

1. Introduction.

Pumping blood in one direction is the main function of the heart, which is equipped with valves that ensure unidirectional flow. Is it possible, though, to pump blood without valves? This paper is intended to show by numerical simulation the possibility of a net flow which is generated by a valveless mechanism in a circulatory system. Simulations of valveless pumping are motivated by physical experiments of Kilner [12], which had been developed from experiments by Liebau [13, 14, 15]. Kilner observed net flow in one direction which depends on the location of periodic forcing in his experiments. We have examined flows driven by pumping without valves in Liebau's model, a loop of tubing of which part is almost rigid and the other part is flexible. In agreement with Kilner, we find that net flow can indeed be driven around such a loop by periodic forcing at one location, but we also find something new: the direction of the flow depends on the driving frequency of the periodic forcing.

As reviewed by Moser et al. [18], there have been several earlier investigations of valveless pumping. Harvey (1628), Weber (1834), Donders (1856), and Thomann [26] suggest theories of valveless pumping, and Ozanam (1881) and Liebau [13, 14, 15] make the various types of physical experiments to explain the mechanism of valveless pumping. Moser et al. [18] try to identify the responsible mechanism and conditions under which this mechanism operates. Their proposed mechanism involves a difference in impedance of two pathways between compliant reservoirs.

One of the applications of valveless pumping may turn out to be cardiopulmonary resuscitation (CPR). The blood flow during CPR has been explained by two theories:

*Received by the editors February 1, 2000; accepted for publication (in revised form) June 8, 2000; published electronically May 10, 2001. Preliminary accounts of this work have appeared in the conference paper of the International Conference on Mechanics in Medicine and Biology, Maui, Hawaii, 2000, pp. 41–44. This work was supported by the National Science Foundation under research grant DMS-9626104. Computation was performed at the Applied Mathematics Laboratory, New York University, and also in part on the Cray T90 computer at the San Diego Supercomputer Center under a grant of resources MCA93S004 from the National Resource Allocations Committee (NRAC).

http://www.siam.org/journals/sisc/23-1/36609.html

†Oak Ridge National Laboratory, P.O. Box 2008, Bldg. 6012, MS-6367, Oak Ridge, TN 37831-6367 (junge@ornl.gov).

‡Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, New York, NY 10012 (peskin@cims.nyu.edu).

the thoracic pump and cardiac compression mechanisms. In support of the thoracic pump model, it has been reported that the heart is "a passive conduit for blood flow" during chest compression [1, 3, 10], with an open mitral valve throughout the cardiac cycle and anterograde (forward) transmitral blood flow even during chest compression. Werner et al. [27] also report that the mitral valve remains open throughout the entire compression-release cycle of CPR while the aortic valve opens during the compression phase of CPR and closes during the release phase. Thus, the left side of the heart appears to act as a conduit for passage of blood, and mitral valve closure is not necessary for forward blood flow during CPR. Despite the observed lack of valve function, some patients with cardiac arrest are successfully resuscitated by external chest-compression CPR [25]. These findings are controversial, however. Other researchers such as Feneley et al. [4] report results that are inconsistent with the thoracic pump theory and support direct cardiac compression as the primary mechanism of blood flow with the high-impulse manual CPR technique. These investigators favor the cardiac compression theory, in which the heart acts as a pump and its valves function normally. It is possible, of course, that both theories are correct, each in a different set of circumstances. Our computational model of valveless pumping might be applicable to the thoracic pump model and help to understand the thoracic pump mechanism. If the magnitude and even the direction of flow in valveless pumping are indeed frequency dependent, as our results seem to indicate, it is of obvious importance to know what frequency of chest compression will produce the most effective CPR.

Another biological example of valveless pumping may occur in the human embryo at the end of the third week of gestation. At this stage of development, the valves of the heart have not yet formed. Nevertheless, there is a net flow in the circulatory system that is somehow generated by the beating of the heart.

An industrial application of valveless pumping is in microelectromechanical system (MEMS) devices [17], where there is a need to produce fluid motion without moving anything inside the fluid. MEMS devices could be built that incorporate flexible flow channels. In that case, our findings might be applicable to the design of valveless pumps for MEMS devices.

We do not attempt to construct a theory of valveless pumping in this report. Instead, we use numerical simulation as an "experimental" tool to study this mysterious phenomenon.

The rest of the paper is organized as follows. In section 2 the immersed boundary method will be introduced. Section 3 is devoted to the description of the two-dimensional valveless pumping model. The results will be discussed in section 4. In section 5 several special cases will be observed. Finally, some conclusions will be drawn in section 6.

2. The immersed boundary method.

2.1. Mathematical formulation. The immersed boundary method is applicable to problems involving an elastic structure interacting with a viscous incompressible fluid. It has been applied to a variety of problems, particularly in biophysics, including two-dimensional and three-dimensional simulations of blood flow in the heart [19, 20, 9, 21, 22], the design of prosthetic cardiac valves [16], platelet aggregation during blood clotting [7], wave propagation in the cochlea [2], the flow of suspensions [8], peristaltic pumping of solid particles [6], and aquatic animal locomotion [5]. The version of the immersed boundary method used in this work is that of [23], except that here we are in two space dimensions.

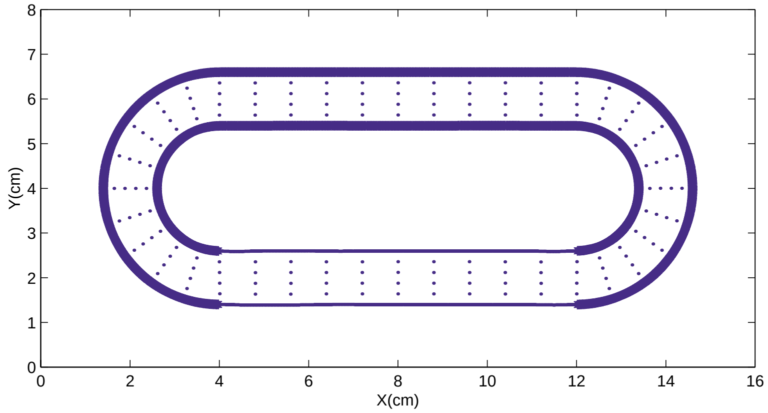


FIG. 1. *Initial position of two-dimensional valveless pumping: flexible boundary (thin lines), almost rigid (thick lines), and fluid markers (dots).*

The philosophy of the immersed boundary method is that the elastic material is treated as a part of the fluid in which singular forces are applied. The fluid and the elastic immersed boundary constitute a coupled mechanical system: The motion of the fluid is influenced by the force generated by the immersed boundary on the fluid, but at the same time the immersed boundary moves at the local fluid velocity and exerts singular forces locally on the fluid. The strength of this method is that it can handle the complicated and time dependent geometry of the elastic immersed boundary which interacts with the fluid, and that it does so while using a fixed regular lattice for the fluid computation.

Consider a viscous incompressible fluid which fills a periodic rectangular box Ω and an immersed boundary S_b in the shape of a racetrack which is contained in the box, where $b = 1$ (inner immersed boundary) or $b = 2$ (outer immersed boundary). Figure 1 displays the two-dimensional model in which we shall simulate valveless pumping.

We shall now consider the mathematical formulation of the equations of motion for the fluid-immersed boundary system. Let ρ be the constant fluid density and μ be the constant viscosity. The equations of motion are then as follows:

$$\rho\left(\frac{\partial \boldsymbol{u}(\boldsymbol{x},t)}{\partial t} + (\boldsymbol{u}(\boldsymbol{x},t)\cdot\nabla)\boldsymbol{u}(\boldsymbol{x},t)\right) + \nabla p(\boldsymbol{x},t) = \mu\nabla^2\boldsymbol{u}(\boldsymbol{x},t) + \boldsymbol{F}(\boldsymbol{x},t), \tag{2.1}$$

$$\nabla\cdot\boldsymbol{u}(\boldsymbol{x},t) = 0, \tag{2.2}$$

$$\boldsymbol{F}(\boldsymbol{x},t) = \int_{S_b} \boldsymbol{f}_b(s,t)\,\delta^2(\boldsymbol{x} - \boldsymbol{X}_b(s,t))\,ds, \tag{2.3}$$

$$\boldsymbol{U}_b(s,t) = \int_{\Omega} \boldsymbol{u}(\boldsymbol{x},t)\,\delta^2(\boldsymbol{x} - \boldsymbol{X}_b(s,t))\,d\boldsymbol{x}, \tag{2.4}$$

$$\frac{\partial \boldsymbol{X}_b(s,t)}{\partial t}(s,t) = \boldsymbol{U}_b(s,t), \tag{2.5}$$

$$\boldsymbol{f}_b(s,t) = -(\kappa_t)_b(\boldsymbol{X}_b(s,t) - \boldsymbol{Z}_b(s,t)) + \kappa_c\left(\frac{\partial^2 \boldsymbol{X}_b(s,t)}{\partial s^2}\right). \tag{2.6}$$

Equations (2.1) and (2.2) are the fluid (Navier–Stokes) equations in Eulerian form. The fluid velocity $\boldsymbol{u}(\boldsymbol{x},t)$, fluid pressure $p(\boldsymbol{x},t)$, and singular force density $\boldsymbol{F}(\boldsymbol{x},t)$ are unknown functions of $(\boldsymbol{x},t)$, where $\boldsymbol{x} = (x,y)$ are fixed Cartesian coordinates and t is the time. Equations (2.5) and (2.6) are the immersed boundary equations in Lagrangian form. The configurations of the immersed boundaries are described by the unknown functions $\boldsymbol{X}_b(s,t)$, where $b = 1$ (inner immersed boundary) or 2 (outer immersed boundary), and $0 \leq s \leq L_1$ for the inner immersed boundary and $0 \leq s \leq L_2$ for the outer immersed boundary. L_1 and L_2 are the unstressed lengths of the inner and outer boundaries, respectively. The boundary force densities $\boldsymbol{f}_b(s,t)$ and the boundary velocities $\boldsymbol{U}_b(s,t)$, for the inner and outer boundaries $b = 1$ and $b = 2$, are also unknown functions of s and t. A fixed value of the Lagrangian parameter s marks a material point of the immersed boundary.

In (2.6) the boundary force is computed as a sum of two terms. In the first term, the given function $\boldsymbol{Z}_b(s,t)$ is called the *target position* of the immersed boundary. This first term provides a restoring force that keeps the boundary points near their target positions. Target positions are used for two purposes in this work: first to maintain the shape of the flow loop, and second, by allowing $\boldsymbol{Z}_b(s,t)$ to change with time, to apply periodic forcing to the immersed boundaries. The curvature term in (2.6) models an elastic membrane under tension. Together, the two terms model the boundaries as tethered elastic membranes.

Note that the curvature term actually is the response to stretching of the membrane; it is not a response to curvature per se. It can be derived from an energy function which is just the sum of the squares of the lengths of the individual segments. Also, the curvature term has little effect on the parts of the racetrack that are stiffly pinned to target points, since those forces dominate there. It is important only on the straight flexible segment.

Equations (2.4) and (2.5) are, in effect, the no-slip condition, since they state that the immersed boundary moves at the local fluid velocity. Equations (2.3) and (2.4) are the interaction equations in mixed Eulerian and Lagrangian form. The core of the immersed boundary method is the delta function, which describes the interaction between the fluid and the immersed boundary. Both of the interaction equations are in integral form with kernel $\delta^2(\boldsymbol{x} - \boldsymbol{X}_b(s,t))$. The integral in (2.4) is taken over the two-dimensional space occupied by the fluid. However, in (2.3), the integral is taken over a one-dimensional space, the immersed boundary, but the delta function is a product of two one-dimensional delta functions: $\delta^2(\boldsymbol{x}) = \delta(x)\delta(y)$. Therefore, $\boldsymbol{F}(\boldsymbol{x},t)$ is a singular force density. The total force is finite despite the singularity in the force density $\boldsymbol{F}(\boldsymbol{x},t)$.

2.2. Numerical method. In this section, we present the summary of the immersed boundary method to find a numerical solution to the system of equations (2.1)–(2.6). For details of this numerical method, see [11, 23]. Let superscripts and subscripts denote the time step index and the spatial discretization, respectively. Let the time proceed in steps of duration Δt, let Δx and Δy be the fluid-lattice spacing, and let Δs_b be the unstressed distance between material points of the immersed boundary. The fluid equations (2.1) and (2.2) in Eulerian form are discretized on a fixed rectangular lattice at time $t = n\Delta t$: $\boldsymbol{x}^n_{jk} = \boldsymbol{x}(j\Delta x, k\Delta y, n\Delta t)$, where $j = 0,\dots,N_x - 1$, $k = 0,\dots,N_y - 1$, and $n = 0,1,\dots$. The immersed boundary equations (2.5) and (2.6) in Lagrangian form are discretized on a collection of moving points in the immersed boundary at time $t = n\Delta t$: $\boldsymbol{X}_{bl}^n = \boldsymbol{X}_b(l\Delta s_b, n\Delta t)$ which do not coincide with the fluid lattice, where $l = 1, \dots,\ M_1$ and $L_1 = M_1\Delta s_1$ for $b = 1$ (inner boundary) or $l = 1, \dots,\ M_2$ and $L_2 = M_2\Delta s_2$ for $b = 2$ (outer boundary).

Our goal is to compute the update $\boldsymbol{u}^{n+1}$, $\boldsymbol{X}_b^{n+1}$ from given $\boldsymbol{u}^n$, $\boldsymbol{X}_b^n$. This is done as follows:

For simplicity, let $N_x = N_y = N$ and choose $\Delta x = \Delta y = h$.

Step 1. Find the force $\boldsymbol{f}_b^n$ on the immersed boundary from the given boundary configuration $\boldsymbol{X}_b^n$.

For $l = 1,\ \dots,\ M_b$, and $b = 1$ (inner boundary) or 2 (outer boundary),

$$\boldsymbol{f}_{bl}^n = -(\kappa_t)_l(\boldsymbol{X}_{bl}^n - \boldsymbol{Z}_{bl}^n) + \kappa_c \left(\frac{\boldsymbol{X}_{b(l+1)}^n - 2\boldsymbol{X}_{bl}^n + \boldsymbol{X}_{b(l-1)}^n}{\Delta s_b^2} \right), \tag{2.7}$$

where $\boldsymbol{Z}_b^n$ is a target position at $t = n\Delta t$, Δs_b is an arc length, κ_t is a stiffness constant, and κ_c is another stiffness constant for the curvature force term.

Note that the subscript arithmetic on l in (2.7) has to be interpreted in a periodic sense, since the boundary is closed: when $l = M_b$, $l + 1 = 1$; when $l = 1$, $l - 1 = M_b$. The target positions $\boldsymbol{Z}_b^n$ are calculated by the given boundary configuration $\boldsymbol{X}_b^n$. The formulations of the target positions will be given in the following section.

Step 2. Spread the boundary force into the nearby lattice points of the fluid using the δ function.

$$\boldsymbol{F}_{jk}^n = \sum_{b=1}^{2}\sum_{l=1}^{M_b} \boldsymbol{f}_{bl}^n \delta_h^2(\boldsymbol{x}_{jk} - \boldsymbol{X}_{bl}^n)\Delta s_b \qquad \text{for} \quad j, k = 0, 1, \dots, N-1, \tag{2.8}$$

where $\boldsymbol{x}_{jk} = (jh, kh)$ and δ_h^2 is a smoothed approximation to the two-dimensional Dirac delta function:

$$\delta_h^2(\boldsymbol{x}) = \frac{1}{h^2}\phi(x/h)\phi(y/h),$$

where

$$\phi(r) = \begin{cases} \frac{3-2|r|+\sqrt{1+4|r|-4r^2}}{8} & \text{if } |r| \le 1, \\ \frac{5-2|r|-\sqrt{-7+12|r|-4r^2}}{8} & \text{if } 1 \le |r| \le 2, \\ 0 & \text{if } 2 \le |r|. \end{cases}$$

The motivation for this particular choice of $\phi(r)$ is given in [22].

Step 3. Solve the Navier–Stokes equations on the rectangular lattice to get the update $\boldsymbol{u}^{n+1}$ and p^{n+1} from $\boldsymbol{u}^n$ and $\boldsymbol{F}^n$. The periodic boundary conditions for the computational domain are imposed. These equations are solved by the following implicit first order scheme in time and space:

$$\rho \left(\frac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^n}{\Delta t} + \boldsymbol{u}^n \cdot \nabla_h^{\pm} \boldsymbol{u}^n \right) + \boldsymbol{D}^0 p^{n+1} = \mu \Delta_h \boldsymbol{u}^{n+1} + \boldsymbol{F}^n, \tag{2.9}$$

$$\boldsymbol{D}^0 \cdot \boldsymbol{u}^{n+1} = 0. \tag{2.10}$$

The difference operators in these equations are constructed as follows. First, the forward (D^+), backward (D^-), and centered (D^0) difference operators are defined in the standard way. Then $\boldsymbol{D}^0$ is defined by $\boldsymbol{D}^0 = (D_x^0, D_y^0)$. This is used in the discrete divergence and gradient. Next, the discrete Laplacian for the viscous term is defined by $\Delta_h \boldsymbol{u} = \sum_{\alpha=1}^{2} D_\alpha^+ D_\alpha^- \boldsymbol{u}$. Finally, the upwind difference operator $\boldsymbol{u} \cdot \nabla_h^\pm = \sum_{\alpha=1}^{2} \boldsymbol{u}_\alpha D_\alpha^\pm$, where

$$u_\alpha D_\alpha^\pm = \begin{cases} u_\alpha D_\alpha^+ & \text{if } u_\alpha < 0, \\ u_\alpha D_\alpha^- & \text{if } u_\alpha > 0. \end{cases}$$

This difference scheme is stable (provided that $\sum_{\alpha=1}^{2} |u_\alpha| \Delta t < h$) because of the choice of the upwind scheme for the convection terms and the backward Euler differencing used for the Stokes system.

Because of the periodic boundary conditions of the computational domain, it is natural to use the fast fourier transform (FFT) algorithm to solve (2.9) and (2.10) for the unknowns $(\boldsymbol{u}^{n+1}, p^{n+1})$. To get the update $\boldsymbol{u}^{n+1}$ and p^{n+1}, first take the discrete Fourier transformation of (2.9) and (2.10). Then, the system can be solved for $\widehat{u}_{lm}$, $\widehat{v}_{lm}$, and $\widehat{p}_{lm}$ for each l and m, $0 \le l, m \le N-1$. Finally, evaluate $\boldsymbol{u}^{n+1}$ and p^{n+1} by applying the inverse FFT algorithm to $\widehat{p}^{n+1}$ and $\widehat{\boldsymbol{u}}^{n+1}$.

Step 4. Once the updated fluid velocity, $\boldsymbol{u}^{n+1}$, has been determined, we can find the velocity, $\boldsymbol{U}_b^{n+1}$, and then the new position, $\boldsymbol{X}_b^{n+1}$, of the immersed boundary points. This is done using a discretization of (2.4) and (2.5).

The difference approximations to the interpolation equation and no-slip condition are expressed as follows:

For $l = 1, \ldots, M_b$, and $b = 1$ (inner boundary) or 2 (outer boundary),

$$\boldsymbol{U}_{bl}^{n+1} = \sum_{j,k=0}^{N-1} \boldsymbol{u}_{jk}^{n+1} \delta_h^2 (\boldsymbol{x}_{jk} - \boldsymbol{X}_{bl}^n) h^2, \tag{2.11}$$

$$\boldsymbol{X}_{bl}^{n+1} = \boldsymbol{X}_{bl}^n + \Delta t \boldsymbol{U}_{bl}^{n+1}. \tag{2.12}$$

Note that we use the same delta function in (2.11) as the one in the interaction equation for the force term, (2.8).

This completes the description of the process (Steps 1–4, above) by which the quantities $\boldsymbol{u}$ and $\boldsymbol{X}$ are updated.

3. Two-dimensional model of valveless pumping.

In this section, we shall introduce a two-dimensional computational model of valveless pumping. The initial configuration of our model is presented first. Then we present the motions of target positions to investigate fluid motions around the flow loop. In particular, we explain how the time dependent target positions are used to provide the periodic forcing which is applied on the one end of the immersed boundary. Finally, we display the physical and computational parameters which are used in our numerical experiments.

3.1. Initial position.

Consider an incompressible viscous fluid with a constant density ρ and viscosity μ in a periodic rectangular box which contains an immersed elastic boundary. Figure 1 shows the initial configuration of the immersed boundary of two-dimensional valveless pumping in our numerical experiments. In this two-dimensional model, the immersed boundary consists of two closed curves, each in the form of a racetrack. The part of each curve shown with thick lines in Figure 1 is almost rigid and the other part with thin lines is flexible. The fluid fills the entire box. Fluid markers, however, are only shown inside the flow loop, since that is the region of interest.

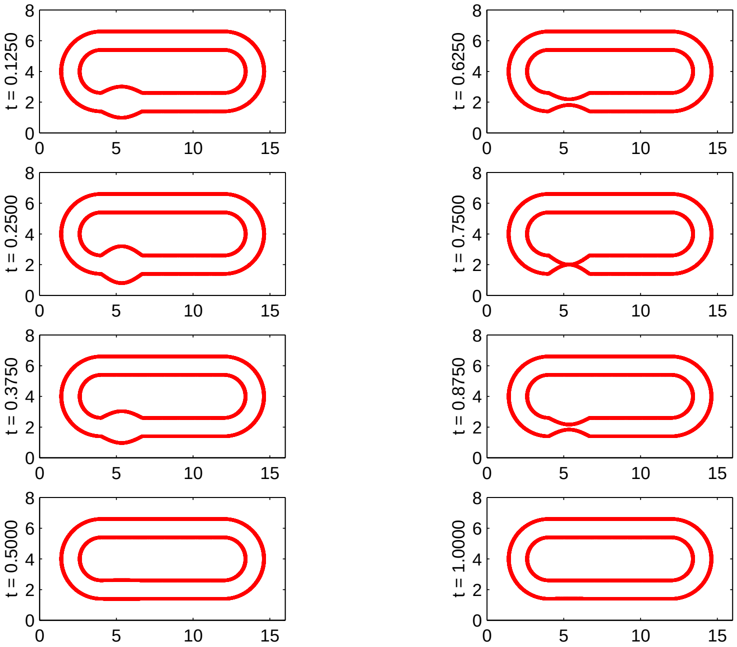


FIG. 2. *The target positions during one cycle: The motion of fluid inside the loop is driven by the periodic vertical expansion and contraction of the target configuration of the tube. This motion is confined to the left* 1/3 *of the flexible segment of tubing.*

Throughout this paper, we assume that the motions are driven by periodic vertical oscillations of the left 1/3 of the flexible tube boundary. Details of how these oscillations are applied will be described next.

3.2. Target positions and parameters. Recall that the equation for the force on the immersed boundary (2.6) involves target positions $\boldsymbol{Z}_b(s,t)$. After discretization, these become $\boldsymbol{Z}_{bl}^n$. For most of the flow loop, these are independent of time and serve the purpose of maintaining the racetrack shape of the flow loop. Time dependent target positions are used in the left 1/3 of the flexible segment of the flow loop (as shown in Figure 2) in order to provide periodic forcing to the flow. Figure 2 displays the target positions at eight equally spaced times over one cycle.

Now we describe the mathematical formulations of the time dependent target positions in the left 1/3 of the flexible segment of tubing.

Let $\boldsymbol{Z}_b(s,t) = (Z_{xb}(s), Z_{yb}(s,t))$, where s is restricted to the range of values that defines the left 1/3 of the flexible segment of tubing. This may be a different range of s values in the case $b = 1$ (inner boundary) than in the case $b = 2$ (outer boundary). Note that the x component of the target position $\boldsymbol{Z}_b(s,t)$ is independent of t, whereas the y component varies with time in the manner that we prescribe.

Define

$$A(s,t) = A_0 \sin\left(\frac{2\pi t}{T}\right) \sin\left(\pi \frac{Z_{xb}(s) - 0.25X_{scale}}{\frac{1}{3}0.5X_{scale}}\right),$$

where A_0 is the amplitude of the target position motion, T is its period, and X_{scale} is the length of the computational domain (i.e., its size in the x direction).

TABLE 1
Physical parameters.

Physical parameters	Symbol	
Density	ρ	1 g/cm^3
Viscosity	μ	0.01 g/cm $\cdot s$
Circumference of a loop	D	28.57 cm
Diameter of tube	d	0.6 cm
Computational domain	$X_{scale} \times Y_{scale}$	16 cm × 8 cm
Period	T	0.05 s $\sim$ 4 s
Amplitude(target)	A_0	0.4 cm and 0.6 cm
Duration of experiment	t_{max}	150 s
Stiffness constant(almost rigid)	κ_t	26000 g/s$^2 \cdot$ cm
Stiffness constant(flexible)	κ_t	900 g/s$^2 \cdot$ cm
Stiffness constant(curvature)	κ_c	120 $g\cdot$ cm/s^2

TABLE 2
Computational parameters.

Computational parameters	Symbol	
Fluid lattice	$N_x \times N_y$	256 × 128
Number of immersed boundary points	$M_1 + M_2$	3654
Meshwidth	$h = \Delta x = \Delta y$	0.0625 cm
Initial distance between boundary points	$\Delta s_1 = \Delta s_2$	$h/4 = 0.0156$ cm
Time step duration	Δt	0.5 h^2 = 0.00195 s

The flexible segment begins at $x = 0.25X_{scale}$ and ends at $x = 0.75X_{scale}$. Thus, the whole flexible segment has length $0.5X_{scale}$, and the part of it in which we allow the target position to move has length $\frac{1}{3}0.5X_{scale}$.

With $A(s,t)$ defined as above, let $Z_{by}(s,t)$ be defined as follows:

$$Z_{by}(s,t) = \begin{cases} 0.25Y_{scale} + 0.5d + A(s,t) & \text{if } b = 1 \text{ (inner boundary)}, \\ 0.25Y_{scale} - 0.5d - A(s,t) & \text{if } b = 2 \text{ (outer boundary)}, \end{cases}$$

where d is the resting diameter of the tube, Y_{scale} is the width of the computational domain (i.e., its size in the y direction), and s is again restricted to the range of values that defines the left 1/3 of the flexible segment of tubing.

The flows that we have investigated are all driven by these periodic motions of the time dependent target positions in the left 1/3 of the flexible segment of tubing.

Note that the target tube, taken as a whole, has nonconstant volume. This is reasonable, since the physical tube is only connected to the target tube by springs and does not follow the target motion in detail; see Figures 10, 11, and 12. In fact, the volume conservation of the physical tube is valid (see Figures 4 and 5) and does not reflect the periodic volume changes imposed on the target tube.

In this work, we use CGS units, but to give a sense of the dimensionless character of the flow, we sometimes report results in terms of a Reynolds number, which is defined by $Re = \frac{\rho U d}{\mu} = \frac{\rho \Phi}{\mu}$, where U is a time-averaged velocity, d is a diameter of the tube, ρ is a constant density, μ is viscosity, and Φ is a time-averaged flux. Note that this Reynolds number refers to the time-averaged velocity and flux, so any nonzero value indicates that valveless pumping has occurred. The Reynolds number, so defined, has varied in our computations between 0 and about 160.

Tables 1 and 2 display the physical and computational parameters, respectively. As indicated in Table 1, the two physical parameters that we systematically vary

Table 3
The ratios of the L_2 difference of velocities.

L_2 difference ratio	
$\|u_{64}\text{-}u_{128}\|_2/\|u_{128}\text{-}u_{256}\|_2$	1.910726
$\|u_{128}\text{-}u_{256}\|_2/\|u_{256}\text{-}u_{512}\|_2$	1.995359
$\|v_{64}\text{-}v_{128}\|_2/\|v_{128}\text{-}v_{256}\|_2$	1.662576
$\|v_{128}\text{-}v_{256}\|_2/\|v_{256}\text{-}v_{512}\|_2$	1.869106

in this work are the period and amplitude of the prescribed motion of the target positions. The resulting flows are definitely dependent on these two parameters. In particular, they determine not only the amount of net flow that develops but even the direction of the net flow around the loop.

4. Results and discussion. The two main results of this paper are as follows: First, a net flow around the loop is produced by the periodic forcing on one end of the flexible boundary, despite the absence of valves. Previous investigators have observed this phenomenon in physical experiments [12], and a theoretical explanation based on a lumped parameter (ODE) model has been proposed [18], but this is the first time, to our knowledge, that valveless pumping has been demonstrated by a computer simulation based on the Navier–Stokes equations. Second, we find that the direction of flow around the loop is determined not only by the position of the periodic compression (as in [12]) but also by the amplitude and frequency of the driving force. This is a new, unexpected phenomenon, not previously reported, and the most important prediction of our model.

Preliminary physical experiments performed in the Courant Institute WetLab confirm that the flow can be driven in either direction from the same location depending on the details of the forcing. Experiments will be described in a future publication with Jun Zhang.

In this section, we first justify our numerical method and then show that the amplitude and the frequency are the crucial parameters to determine the direction of a net flow around the loop. Some special cases of valveless pumping are then discussed.

4.1. Checks on the numerical method. We report two checks on the validity of the numerical method. One check on the computation is to show that our numerical scheme has first order accuracy in time and space. Another check is to see whether the volume of the closed flow loop is conserved.

Numerical convergence. To test accuracy of our numerical scheme, we perform the same computation on the successive lattice refinements and compare the results in the L_2 norm. The physical parameters of this computation are as follows: period = 1.55 s, amplitude of the target position = 0.6 cm, and number of cycles = 96. The other physical parameters are the same as ones in Table 1. We consider the three successive mesh sizes within a fixed size physical domain: $N_x \times N_y = 128\times64$, 256×128, and 512×256. The ratio of the time step duration to the meshwidth is kept fixed throughout this study: $\Delta t/\Delta x = 0.0312$ s/cm. As N_x and N_y vary, the number of points on the immersed boundary changes in proportion to N_x or N_y, which, of course, are changing in proportion to each other. Specifically, we choose an initial distance between immersed boundary points which is equal to $\Delta x/4$.

Table 3 shows the results of the numerical convergence. The ratio of the L_2 norms on the difference of velocities, $\boldsymbol{u} = (u, v)$, at the successive lattice refinements

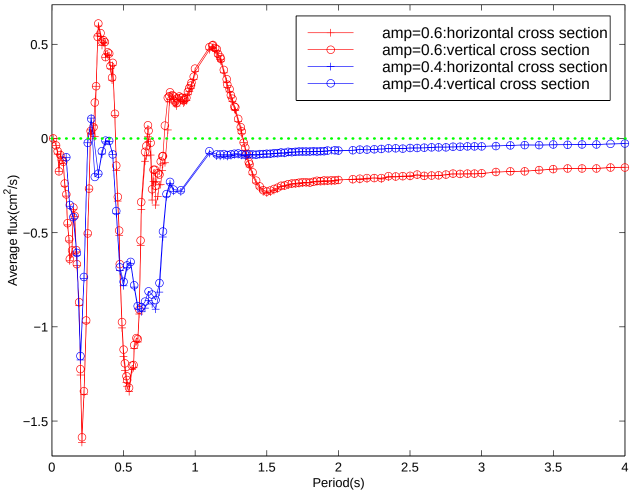


FIG. 3. *Average flow versus period (1/frequency): Flows with two different amplitudes of target positions, $A_0 = 0.6$ cm and $A_0 = 0.4$ cm, are compared. The plus data points denote fluxes computed on a horizontal cross section through the middle of the curved segment of tubing on the right side of the racetrack, and the circle data points denote fluxes computed on a vertical cross section through the middle of the straight segment of tubing at the top of the racetrack. In both cases, the time-averaged flux is plotted as a function of the period of the imposed oscillation in target position that drives the flow. Each pair of data points summarizes a separate numerical experiment, the duration of which is* 150 *s. Positive flux denotes clockwise net flow around the loop of tubing; negative flux denotes counterclockwise net flow. The existence of net flow in these numerical experiments is evidence of valveless pumping. This figure shows that the frequency is a crucial factor to determine the direction and magnitude of flow, and also shows the conservation of volume (area) by the comparison of time-averaged fluxes at two locations.*

are compared in Table 3. Since the asymptotic ratio is almost 2, our numerical method has almost first order accuracy. Presumably, the numbers in the table are converging to 2, but it would take computations on finer grids to show this.

Conservation of volume (area). The volume (area) should be conserved in time, since the fluid is incompressible. The conservation of volume (area) is checked in the following two ways: First, the time-averaged flux on two different cross sections of flow loop are compared. Second, the area inside the flow loop is computed as a function of time to see how much it varies. The time-averaged flux is defined by the mean flux computed on a cross section through the middle of the curved (or straight) segment of tubing on the racetrack over the simulated time.

Figure 3 displays the time-averaged flux, which is the main output of our numerical experiments concerning valveless pumping, plotted as a function of the period of the imposed oscillation in target position that drives the flow. To check volume conservation, these fluxes have been computed on two different cross sections of the tube: a vertical cross section in the middle of the straight segment that forms the top of the tube and a horizontal cross section in the middle of the curved segment of tubing on the right. Two different amplitudes of the target positions, $A_0 = 0.6$ cm and $A_0 = 0.4$ cm, are chosen. The time-averaged fluxes at each of these two cross sections practically coincide, over a wide range of periods. (Figure 3 contains four plots but appears to contain only two because the agreement of flows measured at different cross sections is so good.) Other physical and numerical parameters are as

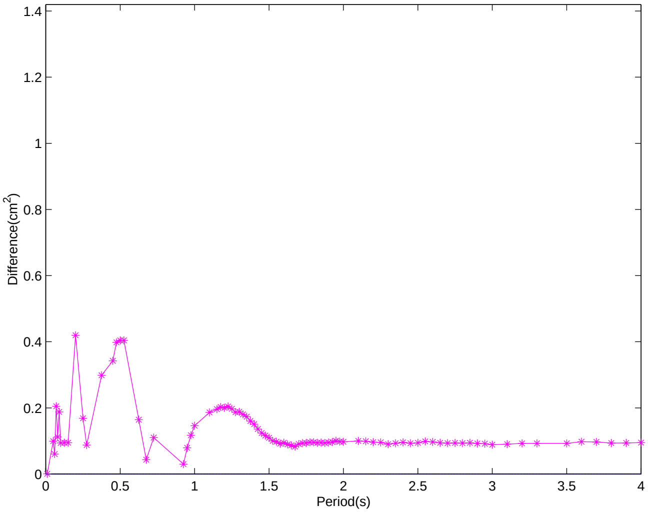


FIG. 4. *Conservation of volume (area): The difference between the time-averaged area and the initial area inside the loop versus the period of the driving oscillation. Each data point summarizes a different numerical experiment of* 150 *s duration (simulated time). The area of the loop is* 34.2796 *cm*^{2} *initially. The maximum error occurs at a driving period of* 0.2 *s, and is equal to* 0.4196 *cm*^{2}*, which is* 1.22% *of the initial area.*

shown in Tables 1 and 2.

Figure 4 shows the difference between the time-averaged area inside the flow loop and the initial area inside the flow loop plotted as a function of the period of the driving oscillation. Each data point summarizes a different numerical experiment of 150 s duration (simulated time). The parameters are also given in Tables 1 and 2 except the amplitude of the driving oscillation, which is 0.6 cm. The initial area of the flow loop is 34.2796 cm^2. The maximum difference between the time-averaged area and the initial area occurs at period 0.2 s, and it is only 0.4196 cm^2, which is 1.22% of the initial area inside the flow loop.

As a further check on the volume (area) conservation, we plot the area within the flow loop as a function of time. This is done for only one case, the driving period of 0.2 s, at which the maximum difference between the time-averaged area and the initial area occurs. Even in this worst case, the area as a function of time is nearly constant; see Figure 5, and note the expanded scale of the plot.

The volume errors we observed in this subsection are judged to be acceptable, but it could be further reduced if desired by using the method of Peskin and Printz [24].

4.2. The time-averaged flow around the loop as a function of the amplitude of the driving oscillation. In this section, we investigate the influence of the amplitude of the driving oscillation (i.e., the amplitude of the prescribed target position motion) on the magnitude and direction of the net flow around the loop of simulated tubing. Four different periods of the driving oscillation are considered: $T = 0.3$ s, 0.375 s, 0.525 s, and 1.7 s. Figure 6 displays the time-averaged flux as a function of the amplitude of the driving oscillation at these four chosen periods. Other parameters besides amplitude and period are given in Tables 1 and 2. Positive flow values denote clockwise net flow, and negative values denote counterclockwise net flow.

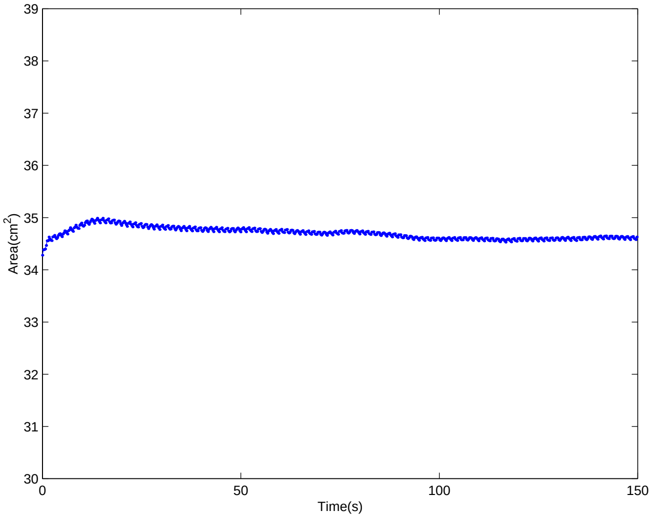


FIG. 5. *Conservation of volume (area): Area versus time at a driving period of* 0.2 *s. The area inside the flow loop is almost constant in time (note the expanded scale of the plot:* 0 *is way offscale) even though the chosen period of* 0.2 *s is the one at which the maximum difference between the time-averaged area and the initial area occurs.*

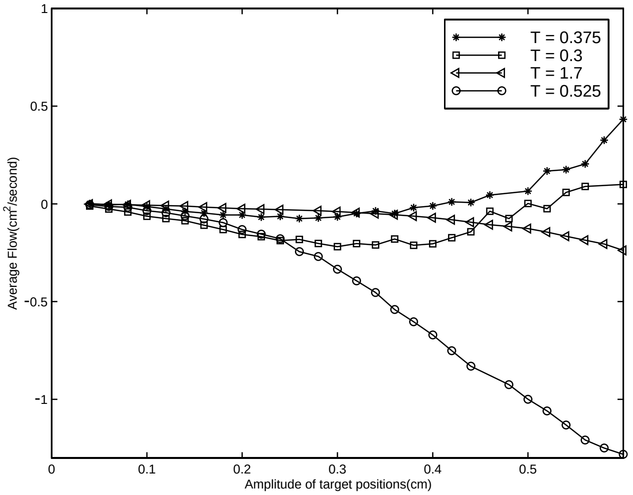


FIG. 6. *Average flux versus amplitude: The flows at four different periods, $T = 0.3$ s, 0.375 s, 0.525 s, and 1.7 s, are compared. Each data point summarizes a separate numerical experiment of 150 s duration (simulated time). For all periods shown, the flow is negative (counterclockwise) at low amplitude, but for some periods ($T = 0.3$ s and $T = 0.375$ s) it reverses and becomes positive (clockwise) at high amplitude.*

The results plotted in Figure 6 show the following features:

- At low amplitude of the driving oscillation, net flow is always in the counterclockwise direction. Its magnitude at any given amplitude depends on the period of the driving oscillation. Of the four examples given in the figure, the periods $T = 0.375$ s and $T = 1.7$ s result in only weak counterclockwise flow, whereas $T = 0.3$ s and $T = 0.525$ s result in much stronger counterclockwise flow.

- As the amplitude increases a qualitative distinction between the different cases appears. For $T = 0.525$ s and $T = 1.7$ s, the counterclockwise flow simply gets stronger monotonically as the amplitude of the driving oscillation increases. But for $T = 0.3$ s and for $T = 0.375$ s, the flow changes direction at some critical amplitude and becomes clockwise at high amplitude. Note that one cannot predict from the strength of the counterclockwise flow at low amplitude which of the cases will have clockwise flow at high amplitude: Of the two cases that have clockwise flow at high amplitude, one had a strong counterclockwise flow and the other had a weak counterclockwise flow at low amplitude.

Overall, there seems to be a preference for counterclockwise flow in these results. All periods generate counterclockwise flow at low amplitude, and only some periods generate flows that reverse and become clockwise at high amplitude. We can speculate on the reason for this, as follows. Recall that the driving oscillation is imposed at the left end of the flexible segment of tubing, which forms the lower straight segment of the racetrack; see Figure 2. If waves propagate from this source to the right along the flexible segment, these would tend to generate counterclockwise flow by a peristaltic mechanism. This argument leaves open the question of why the flows reverse and become clockwise, for some periods of the driving oscillation, when the amplitude of the driving oscillation becomes sufficiently large.

4.3. The time-averaged flow around a loop as a function of frequency. In this subsection, we present a new, unexpected phenomenon which is the most important prediction of our model: the driving frequency (1/period) is a crucial parameter to determine the magnitude and even the direction of a net flow generated by valveless pumping.

The time-averaged flow around a loop as a function of the period of the driving oscillation is investigated for two different amplitudes of the driving oscillation, $A_0 = 0.4$ cm and $A_0 = 0.6$ cm. In Figure 3, we plot the time-averaged flux versus period for these two cases. Each data point is the result of a separate numerical experiment, and the parameters are the same as in Tables 1 and 2. As before, positive flow is clockwise, and negative flow is counterclockwise.

The result that is obvious from a glance at Figure 3 is that valveless pumping has a strong dependence on the frequency of the driving oscillation. Indeed, there appear to be resonances at rather specific frequencies, which are most effective in driving the flow in one direction or the other. At the lower amplitude, the net flow is almost always counterclockwise, so these peaks are in the negative direction. As we shift to the higher amplitude, the negative peaks seem to be preserved, but now positive peaks emerge as well. Another indication of the dynamic character of valveless pumping is that it seems to disappear at the extremes of frequency. In the high-frequency (low-period) limit, it is clear from Figure 3 that the net flow approaches zero. This also seems to be true in the low-frequency (high-period) limit, although for the higher amplitude data one cannot be sure whether the net flux is approaching zero or some negative value. In any case, strong valveless pumping happens at specific frequencies that are neither too large nor too small.

Since the driving frequency is an important parameter of valveless pumping, it may be of interest to interpret our results in terms of the Womersley number $Wo = d\sqrt{\omega/\nu}$, where d is the tube diameter (0.6 cm), ν is the kinematic viscosity ($\nu = \mu/\rho =$

0.01 cm^2/s), and ω is the driving frequency in radians/s ($\omega = 2\pi/T$). For example, the maximum average clockwise flow occurs at a period of $T = 0.325$ s, which is a Womersley number of $Wo = 26$, and the maximum average counterclockwise flow occurs at a period of $T = 0.21$ s, corresponding to a Womersley number of $Wo = 33$. In both cases, the Womersley number is substantially larger than 1, which means that the velocity profile is far from parabolic.

5. Case studies. In this section, several special cases of valveless pumping are studied. Recall Figure 3 in the previous section. We chose the special cases based on the results from that figure. The amplitude of the driving oscillation, $A_0 = 0.6$ cm, is chosen, since a qualitative distinction between the different cases appears as the amplitude increases, and $A_0 = 0.6$ cm is large enough to show that distinction. The following three cases are considered:

- Maximum average clockwise flow ($T = 0.325$ s).
- Almost zero flow ($T = 1.34$ s).
- Maximum average counterclockwise flow ($T = 0.21$ s).

As before, the fluid motions are driven by the oscillations in target positions which are imposed along the 1/3 left end of the flexible segment of tubing, which forms the lower straight segment of the racetrack. The parameters are as given in Table 1 and 2.

These three cases have been investigated and compared qualitatively in the following ways.

- The angles from the center of the computational domain, $(x, y) =$ (8 cm, 4 cm), to the current positions of the fluid markers inside the flow loop are measured in order to determine the direction of the flow.
- Flowmeter fluxes computed on the vertical cross section through the middle of the straight segment of tubing at the top of the racetrack as functions of time are measured to test whether the fluid motion is in a periodic steady-state through the final duration, $t_{max} = 150$ s, and to show the nature of the oscillation and the net progress of the fluid motions.
- The wave motions along the top of the flexible boundaries over one cycle of the periodic steady-state are investigated in order to determine whether the motion looks like a traveling wave or a standing wave (or some other more complicated kind of wave motion).
- The target positions and the real physical positions of the immersed boundary, in particular the flexible segment, are compared in order to see how the time dependent target positions affect the motions of the real physical boundary.
- The velocity vector fields and pressure contours of the maximum clockwise and the maximum counterclockwise cases at 4 different phases over one period after the periodic steady-state are presented.
- Changing the direction of the flow by changing the period *during* a computer experiment.
- Zero net flux for the symmetric driving force.

5.1. Three cases. Angle. Here we examine the net progress of the flow by following the angular position of selected fluid markers. The angles are measured from the center of the computational domain, $(x, y) =$ (8 cm, 4 cm), to the positions of the fluid markers as functions of time. The angle is increased as the position of the fluid marker is changed in the clockwise direction. We choose arbitrarily 6 fluid markers around the flow loop for each case. These 6 fluid markers are located at the same position initially in all three cases.

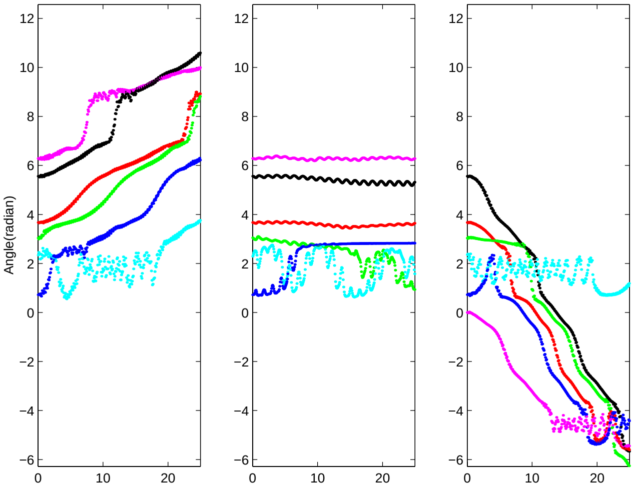


FIG. 7. *The angles of fluid markers are plotted as functions of time. The leftmost and rightmost figures show the case of maximum average flow in the clockwise direction (positive slope) and the case of maximum average flow in the counterclockwise direction (negative slope), respectively. The middle figure shows the case of almost zero flow (almost zero slope).*

Figure 7 displays the change of the positions of 6 fluid markers by measuring the angle as a function of time for the three cases. The angles are plotted at every 10 time steps up to $t_{max} = 25$ s. Since there are some vortices inside the flexible segment, some fluid markers get trapped and take time to escape the segment. Once fluid markers do escape, however, they move much faster along the rigid part of the racetrack. Examples are the sixth fluid marker in the leftmost frame, the second one in the middle frame, and the third one at the rightmost frame. Some other fluid markers stick to the immersed boundary. Ignoring these details and looking at the general trend, we can see that there is net clockwise motion of the markers in the leftmost frame, no net motion in the middle frame, and net counterclockwise motion in the rightmost frame of Figure 7.

Flowmeter. Figure 8 displays flowmeter results, which are the fluxes computed on the vertical cross section through the middle of the straight segment of tubing at the top of the racetrack. These results are plotted as functions of time over the last 5 cycles in each case. The positive values denote clockwise flow and the negative denote counterclockwise flow. In all three cases the flow is oscillatory, and the oscillation has settled down to a periodic steady state. The flow changes direction with a positive phase and a negative phase during each cycle. In two of the three cases (top and bottom in Figure 8) there is a nonzero mean flow superimposed upon the oscillatory motion. This nonzero mean flow is the phenomenon of valveless pumping.

The motions of wave along the flexible segment. We observe another interesting phenomenon of valveless pumping by investigating the wave motions along the flexible boundary. Figure 9 displays the motions of wave along the top of the flexible segment. Sixteen equal-time snapshots of the wave motions along the top of flexible segment over one cycle of the periodic steady-state are plotted for the three cases. In the top frame, there is a standing wave pattern with two nodes (at about 6.3 cm and 9.8 cm). For reasons that we do not understand, this standing wave

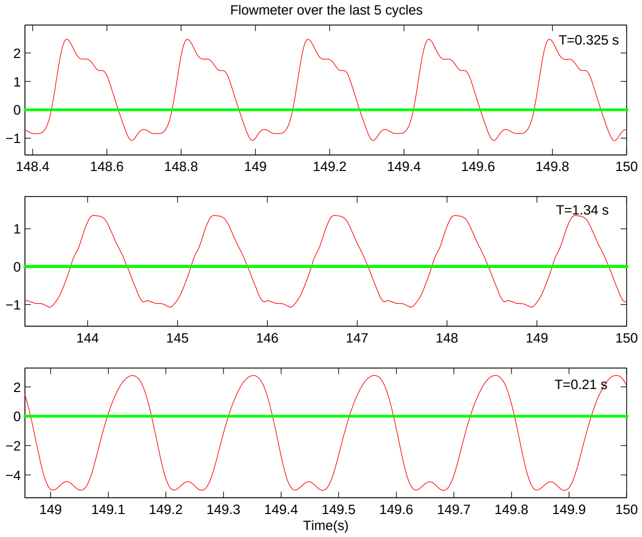


Fig. 8. *Flowmeter. The fluxes are computed on the vertical cross section through the middle of the straight segment of tubing at the top of the racetrack. They are plotted as functions of time are plotted over the last five cycles in each case (note the different time scales). The case of maximum average flow in the clockwise direction, almost zero flow, and maximum flow in the counterclockwise direction are considered from top and bottom. Note that the mean flow is positive (clockwise) in the top frame and negative (counterclockwise) in the bottom frame.*

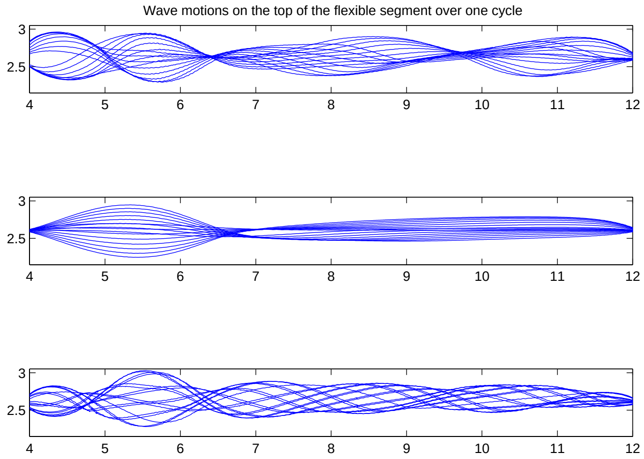


Fig. 9. *Sixteen equal-time snapshots over one cycle of the periodic steady-state wave motions along the top of the flexible segment are plotted. The top frame shows the case of maximum average flow in the clockwise direction. The middle frame shows the case of almost zero net flow. The bottom frame shows the case of maximum average flow in the counterclockwise direction. In all these cases, the source of vibration is confined to the left* 1/3 *of the flexible segment, i.e., to the interval from* 4 *cm to* 6.7 *cm.*

pattern is associated with maximum clockwise flow. In the middle frame, there again seems to be a standing wave pattern with just one node (at about 6.7 cm). Note,

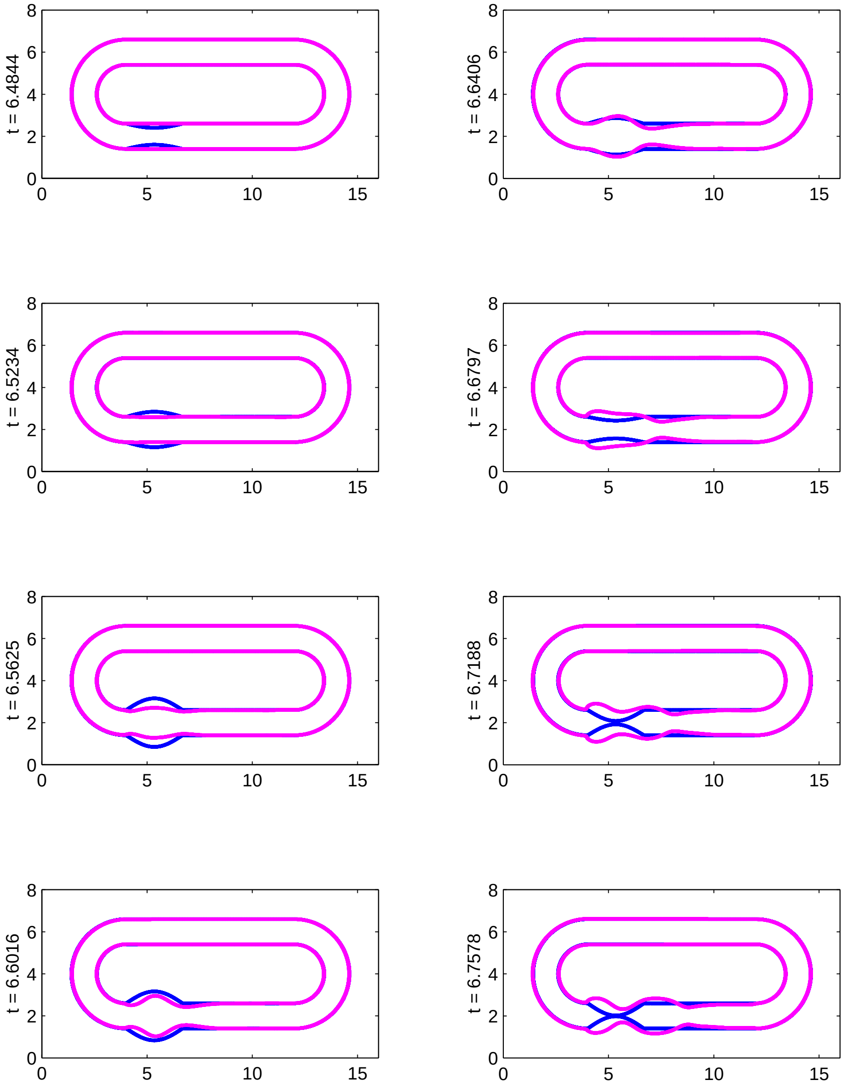


FIG. 10. *Comparison of the motions of the target positions (dark) and the physical boundary (light) for the case of maximum average clockwise flow.*

however, that the location of this node coincides with the edge of the driven part of the flexible boundary, i.e., the part where an oscillation of the target positions is imposed. Thus, it seems that the driven part of the flexible boundary is oscillating in one phase, and that the rest of the flexible boundary is oscillating in antiphase with flexible part. This wave pattern seems to be associated with the absence of valveless pumping, i.e., with zero net flow. In the bottom frame we see traveling waves (note the absence of nodes) propagating to the right, away from the driven part of the flexible boundary. Such traveling waves might be expected to pump fluid in the direction of propagation by a peristaltic mechanism, and indeed what we see in this case is maximum counterclockwise flow.

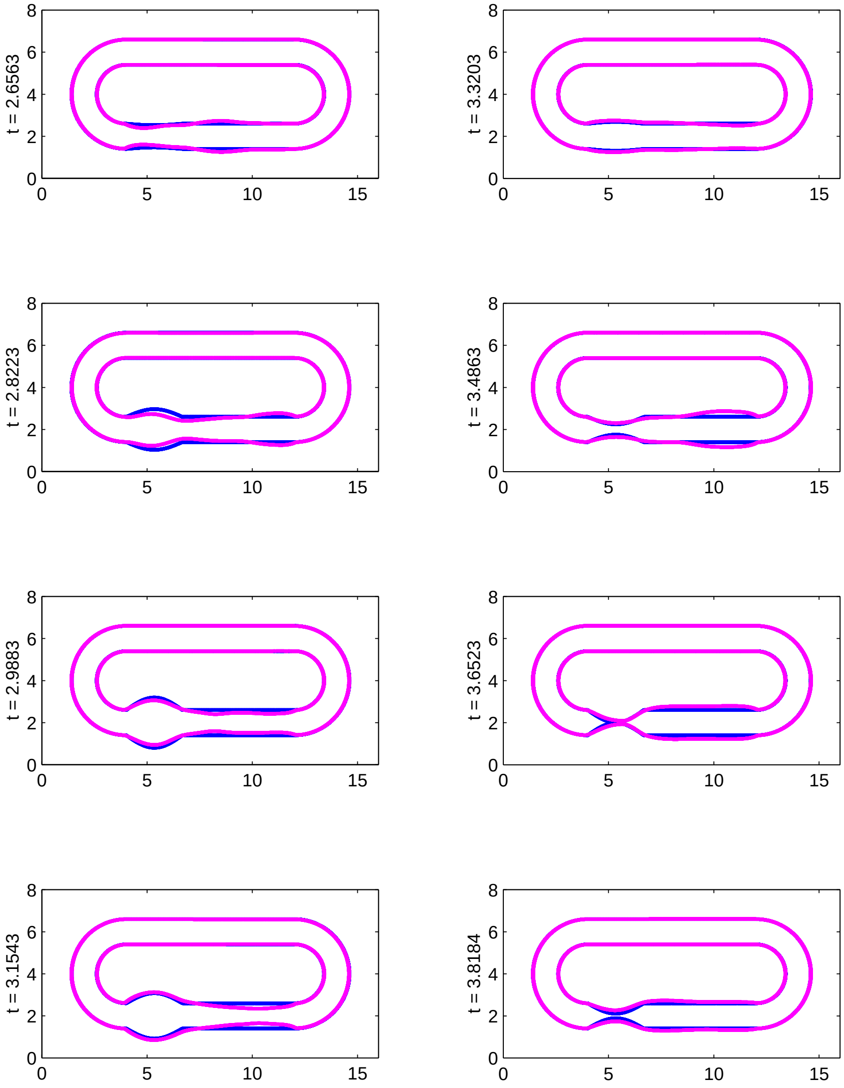


FIG. 11. *Comparison of the motions of the target positions (dark) and the physical boundary (light) for the case of almost zero flow.*

Comparison of the motions of the target positions and the real physical boundary. Here we compare the motions of the target positions and the physical boundary for the three cases. This is done in Figure 10 for the case of maximum average clockwise flow, in Figure 11 for the case of almost zero net flow, and in Figure 12 for the case of maximum average counterclockwise flow. The target positions (dark) are the same in all three figures, since the target motion is specified in advance and differs in three cases only with respect to time scale. Note also that the target positions are time dependent only in the left 1/3 of the flexible segment, which form the bottom of the racetrack.

Only in the case of zero net flow (Figure 11) does the physical boundary motion

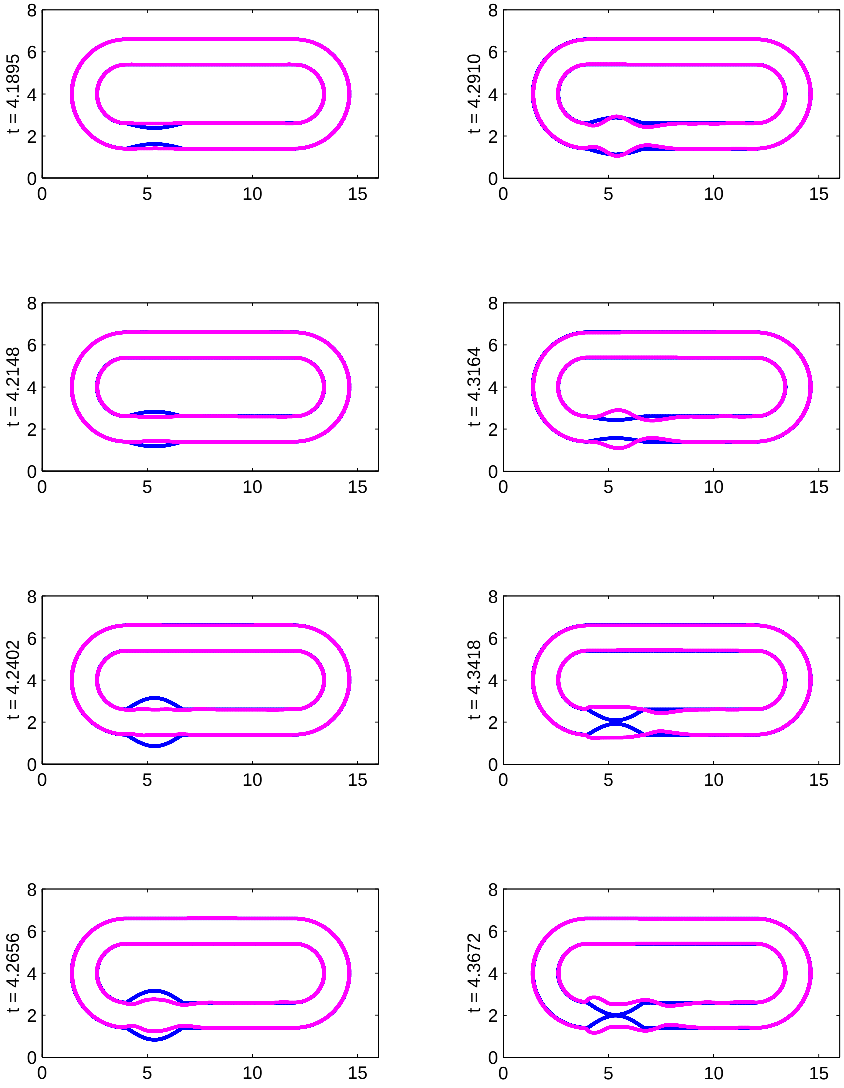


FIG. 12. *Comparison of the motions of the target positions (dark) and the physical boundary (light) for the case of maximum average counterclockwise flow.*

track the target position motion. This is probably because the frequency is too low for inertia to introduce any phase lags. In the other two cases, there are substantial phase differences (presumably consequences of fluid inertia) between the target position and the physical boundary position.

The velocity vector fields and pressure contours. Figures 13 and 14 display the velocity vector fields of the maximum clockwise flow (period = 0.325 s). Four equal-time snapshots over one cycle of the periodic steady-state are plotted. Figures 15 and 16 display the velocity vector fields of the maximum counterclockwise flow (period = 0.21 s). Four equal-time snapshots over one cycle of the periodic steady-state are also plotted.

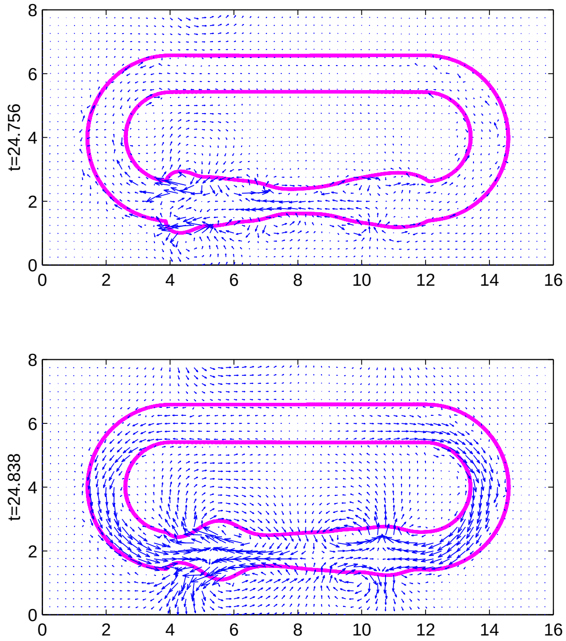


Fig. 13. *Velocity vector fields of the maximum clockwise flow. Four equal-time snapshots over one cycle of the periodic steady-state are plotted here and in Figure* 14.

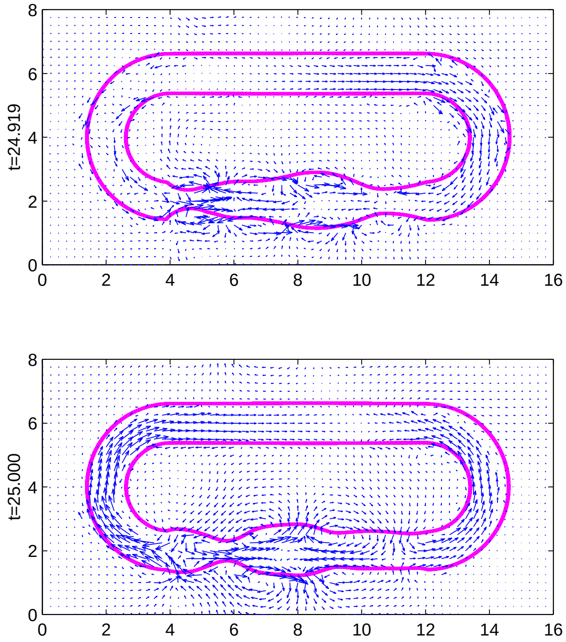


Fig. 14. *Continuation of Figure* 13.

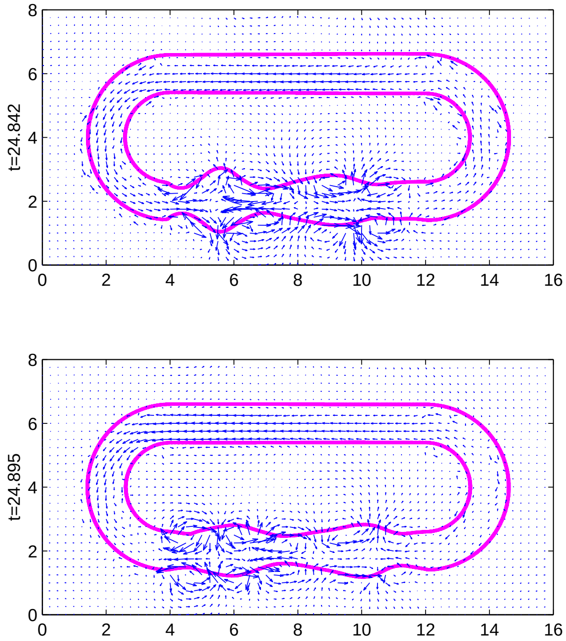


FIG. 15. *Velocity vector fields of the maximum counterclockwise flow. Four equal-time snapshots over one cycle of the periodic steady-state are plotted here and in Figure* 16.

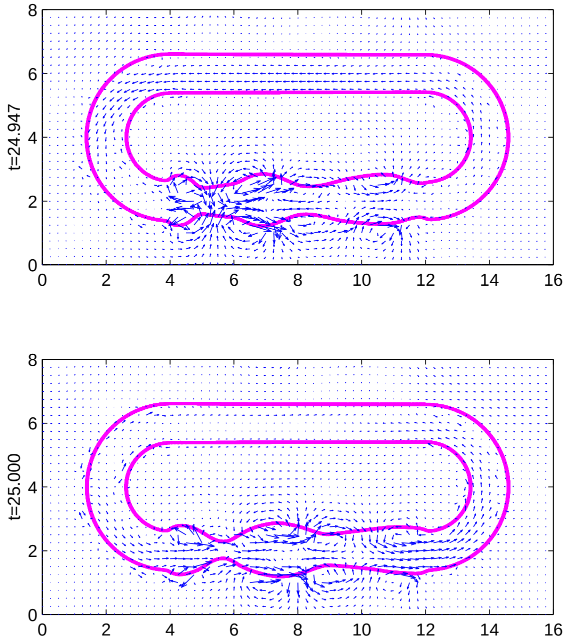


FIG. 16. *Continuation of Figure* 15.

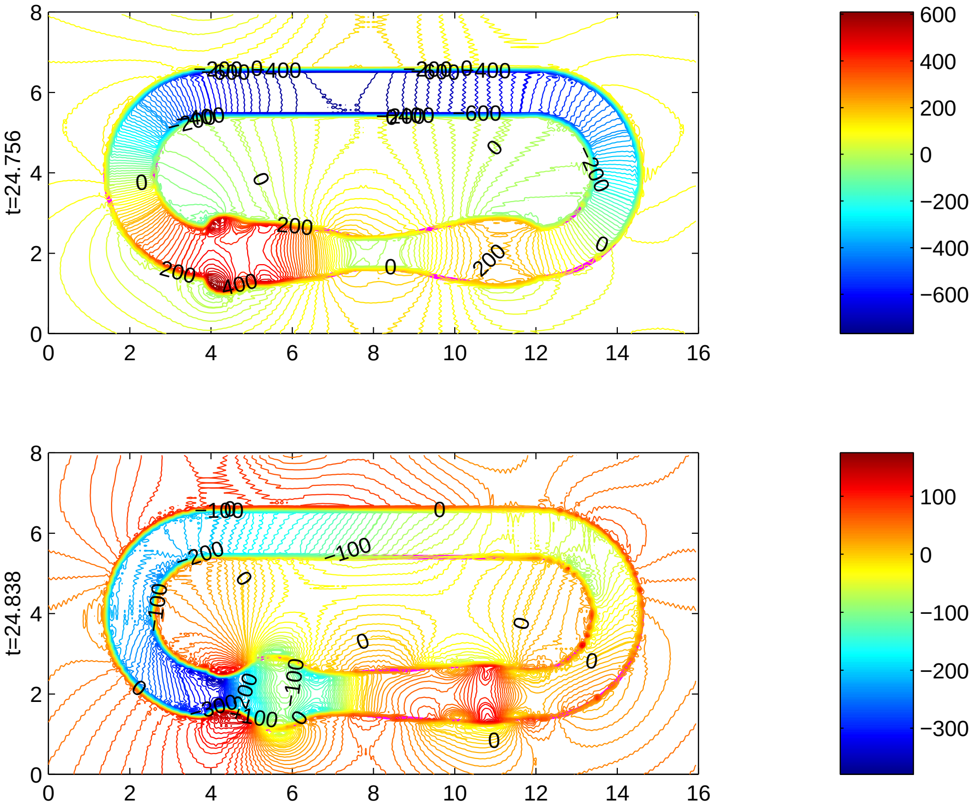


Fig. 17. *Pressure contours of the maximum clockwise flow. Four equal-time snapshots over one cycle after the periodic steady-state, plotted here and in Figure* 18. *The units of pressure are dynes/cm*^{2}.

Figures 17 and 18 display the pressure contours of the maximum clockwise flow. Four equal-time snapshots over one cycle of the periodic steady-state are plotted. Figures 19 and 20 display the pressure contours of the maximum counterclockwise flow. Four equal-time snapshots over one cycle of the periodic steady-state are also plotted.

Note that it is true that the positions of the immersed boundary are influenced not only by the fluid inside of the loop but also by the fluid outside of the loop. However, motions of the fluid inside the loop seem to be dominant, since that is where the larger velocities and pressure gradients are typically seen in Figures 13–20.

5.2. Further case studies. Flow which is changing the direction by changing the period *during* a computer experiment (periods, $T = 0.325$ s and $T = 0.21$ s). In this section, we present two more interesting cases. In Figure 21, we show that the result does not depend on initial conditions. One might worry that once the flow starts going one way it will keep going that way, but this case shows this is *not* true. In order to show that the crucial parameter to decide the direction of a net flow is frequency, we have examined the situation in which the period of the driving oscillation is 0.325 s for the first half of the simulated experiment, and then

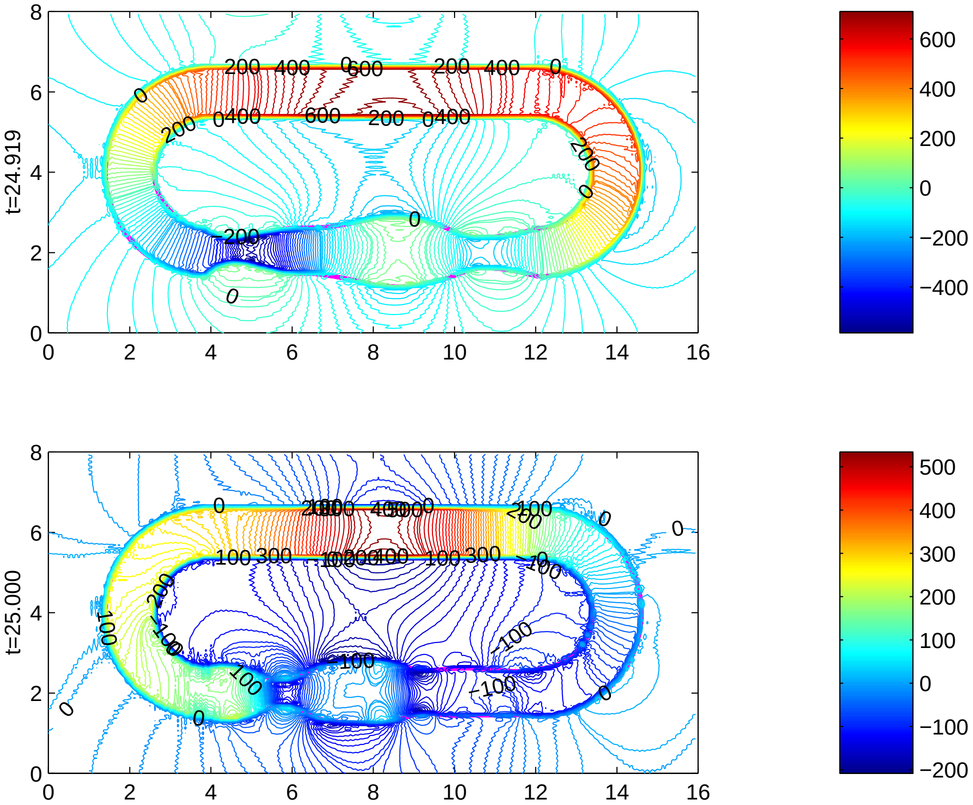


FIG. 18. *Continuation of Figure* 17.

changes to 0.21 s for the balance of the simulated experiment. Note that these are the periods which generated maximum net flow in the clockwise and counterclockwise direction, respectively. All other parameters except the period are fixed during the simulation.

Figure 21 displays the changing of the positions of two arbitrary fluid markers inside the flow loop by measuring the angles from the center, (x, y) = (8 cm, 4 cm), to the current positions of the markers. We plot angles as functions of time at every 10 time steps up to t = 20 s for each fluid marker. The curves in Figure 21 are changing from increasing (clockwise direction) to decreasing (counterclockwise direction) around 10 s.

Zero net flux for the symmetric driving force. Is there still a net flow if the system of valveless pumping would be symmetric? We consider the following special case to show that there is almost zero net flow when the periodic driving forcing is imposed on the center of the flexible segments. All other parameters are chosen for the case of the maximum counterclockwise flow, $T = 0.21$ s. This experiment is run until the periodic steady-state $t = 150$ s. The time-averaged flux for this case is -0.023625 cm^2/s. This shows that there is almost zero net flow when the system is symmetric.

From this case and the previous one, we see that valveless pumping does *not* represent an instability of a symmetric situation. On the contrary, the direction of

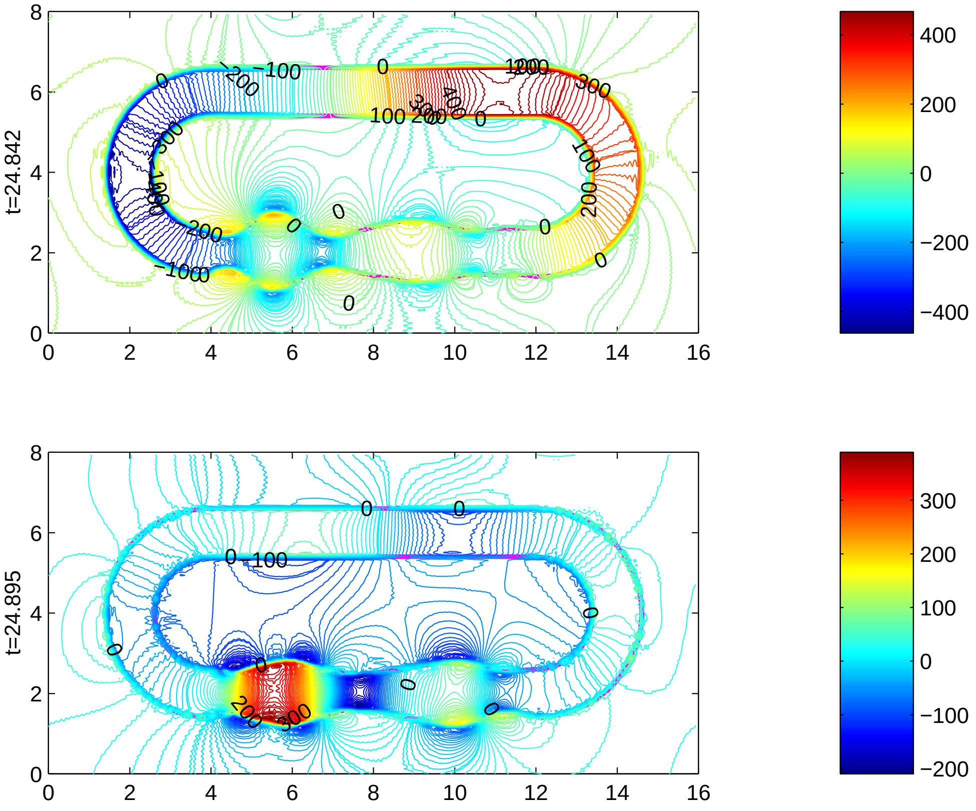


FIG. 19. *Pressure contours of the maximum counterclockwise flow. Four equal-time snapshots over one cycle after the periodic steady-state, plotted here and in Figure* 20. *The units of pressure are dynes/cm*^{2}*.*

the mean flow is determined by the asymmetry of the problem but in a frequency and amplitude dependent manner.

6. Conclusions.

We have presented numerical experiments concerning "valveless pumping" in the two-dimensional case using the immersed boundary method. As in the earlier papers and physical experiments of valveless pumping, we have also observed the existence of a net flow. Furthermore, we have presented the new, unexpected result that the direction of the flow around the loop of tubing is decided not only by the position of the driving oscillations but also by the *frequency* and the *amplitude* of the driving oscillations. Since CPR may involve valveless pumping, it is of obvious importance to know what frequency and amplitude of chest compression will produce the most effective CPR. Of course we cannot hope to answer this question quantitatively with such an idealized model, but perhaps we have shown qualitatively what phenomena may be expected as the frequency and amplitude of the driving oscillation are varied. We have put special emphasis on the conditions that generate maximum net flow, since that is the goal of CPR.

In studying these cases, we have found an interesting phenomenon: the clockwise net flow seems to be associated with a standing wave in the flexible segment of tubing,

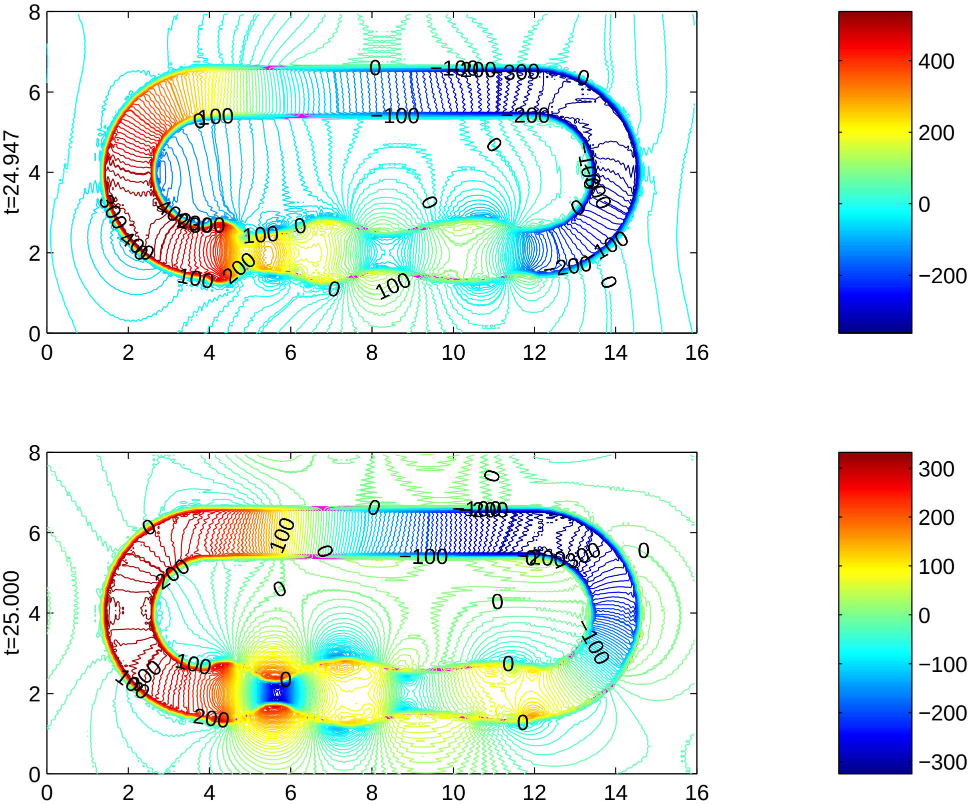


FIG. 20. *Continuation of Figure* 19.

whereas the counterclockwise net flow seems to be associated with a traveling wave. In the clockwise case (standing wave) the flow in the flexible segment of tubing is going toward the site at which the periodic forcing is applied, but in the counterclockwise case (traveling wave) it is going away from that site and in the same direction as the traveling wave. Therefore, we believe that the counterclockwise flow is driven by the traveling wave via a peristaltic mechanism, but we have no explanation for the clockwise flow in the standing-wave case.

The immersed boundary methodology used here may also be applicable to other biological instances of valveless pumping, such as the blood circulation within the human embryo at the end of the third week of gestation, and to engineering applications such as the design of MEMS.

We are confident that numerical experiments such as those begun in this paper will help answer many questions about the mechanism of valveless pumping. Even though the results demonstrate success in modeling valveless pumping, there is still much future work that remains to be done, such as giving a theoretical explanation for this mysterious phenomenon, and extending this model to the three-dimensional case in order to make it more realistic and more applicable to real-world biomedical problems, like CPR.

A physical experiment of valveless pumping is being constructed at the Courant

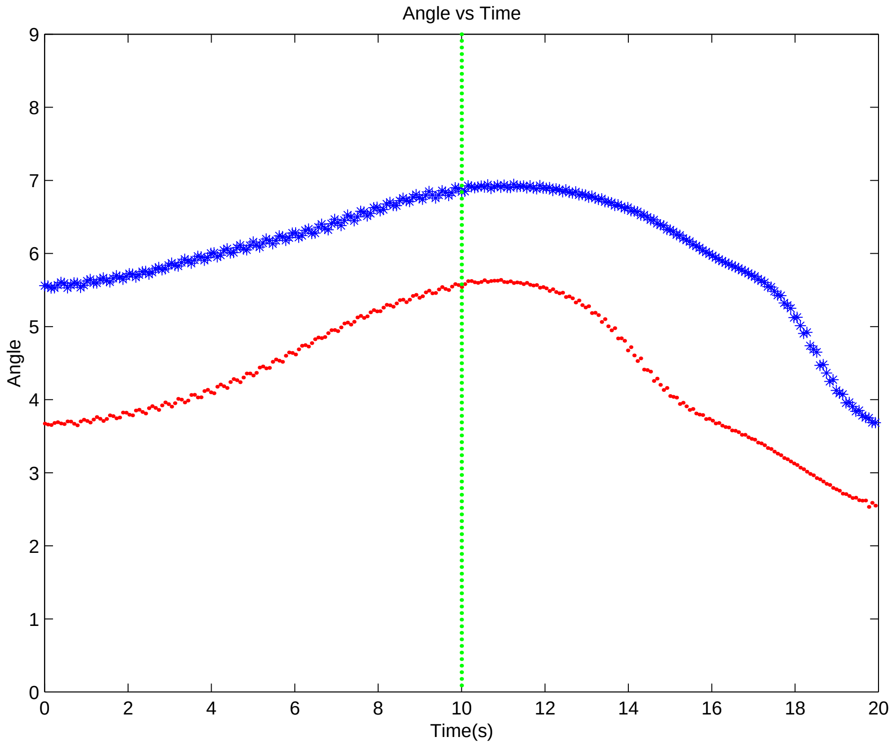


FIG. 21. *Flow which is changing the direction by changing the period* during *a computer experiment: The angles of the positions of two fluid markers are changed from increasing (clockwise) to decreasing (counterclockwise) in time by changing the period from* 0.325 *s to* 0.21 *s at time* = 10 *s. This result shows that the result does not depend on the initial condition.*

Institute WetLab. An important part of the future work will be the comparison of computed and experimental results.

Acknowledgments. The authors are especially grateful to David M. McQueen for many helpful discussions and to Jun Zhang for making the physical experiment of valveless pumping. Thanks also to Simcha Milo for pointing out the problem, to Philip Kilner for the experiments that inspired this research project and for helpful discussions, and to Mory Gharib for further experimental insight into valveless pumping.

REFERENCES

[1] C. Beattie, A. D. Guerci, T. Hall, A. M. Borkon, W. Baumgartner, R. S. Stuart, J. Peters, H. Halperin, and J. L. Robotham, *Mechanisms of blood flow during pneumatic vest cardiopulmonary resuscitation*, J. Appl. Physiol., 70 (1991), pp. 454–465.

[2] R. P. Beyer, *A computational model of the cochlea using the immersed boundary method*, J. Comput. Phys., 98 (1992), pp. 145–162.

[3] J. M. Criley, J. T. Niemann, J. P. Rosborough, S. Ung, and J. Suzuki, *The heart is a conduit in CPR*, Crit. Care Med., 9 (1981), p. 373.

[4] M. P. Feneley, G. M. Maier, J. W. Gaynor, S. G. Gall, J. K. Kisslo, J. W. Davis, and J. S. Rankin, *Sequence of mitral valve motion and transmitral blood flow during manual cardiopulmonary resuscitation in dogs*, Circulation, 76 (1987), pp. 363–375.

[5] L. J. Fauci and C. S. Peskin, *A computational model of aquatic animal locomotion*, J. Comput. Phys., 77 (1988), pp. 85–108.

[6] L. J. Fauci, *Peristaltic pumping of solid particles*, Comput. & Fluids, 21 (1992), pp. 583–598.

[7] A. L. Fogelson, *A mathematical model and numerical method for studying platelet adhesion and aggregation during blood clotting*, J. Comput. Phys., 56 (1984), pp. 111–134.

[8] A. L. Fogelson and C. S. Peskin, *A fast numerical method for solving the three-dimensional Stoke's equations in the presence of suspended particles*, J. Comput. Phys., 79 (1988), pp. 50–69.

[9] S. Greenberg, D. M. McQueen, and C. S. Peskin, *Three-dimensional fluid dynamics in a two-dimensional amount of central memory*, in Wave Motion: Theory, Modelling, and Computation, Math. Sci. Res. Inst. Publ. 7, Springer-Verlag, New York, 1987, pp. 85–146.

[10] H. R. Halperin, J. E. Tsitlik, R. Beyar, N. Chandra, and A. D. Guerci, *Intrathoracic pressure fluctuations move blood during CPR: Comparison of hemodynamic data with predictions from a mathematical model*, Ann. Biomed. Engrg., 15 (1987), pp. 385–403.

[11] E. Jung, 2-*D Simulations of Valveless Pumping Using the Immersed Boundary Method*, Ph.D. thesis, Courant Institute of Mathematical Sciences, New York University, New York, 1999.

[12] P. J. Kilner, *Formed flow, fluid oscillation and the heart as a morphodynamic pump (abstract)*, European Surgical Research, 19 (1987), pp. 89–90.

[13] G. Liebau, *Die Bedeutung der Tragheitskrafte für die Dynamik des Blutkreislaufs*, Zs Kreislaufforschung, 46 (1957), pp. 428–438.

[14] G. Liebau, *Die Stromungsprinzipien des Herzens*, Zs Kreislaufforschung, 44 (1955), pp. 677–684.

[15] G. Liebau, *Über ein Ventilloses Pumpprinzip*, Naturwissenschsften, 41 (1954), pp. 327–328.

[16] D. M. McQueen, C. S. Peskin, and E. L. Yellin, *Fluid dynamics of the mitral valve: Physiological aspects of a mathematical model*, Amer. J. Physiol., 242 (1982), pp. H1095–H1110.

[17] S. Miyazaki, T. Kawai, and M. Araragi, *A piezo-electric pump driven by a flexural progressive wave*, in Proceedings of the IEEE Micro Electro Mechanical Systems Workshop, Nara, Japan, 1991, pp. 283–288.

[18] M. Moser, J. W. Huang, G. S. Schwarz, T. Kenner, and A. Noordergraaf, *Impedance defined flow, generalisation of William Harvey's concept of the circulation*—370 *years later*, Internat. J. Cardiovascular Medicine and Science, 1 (1998), pp. 205–211.

[19] C. S. Peskin, *Flow Patterns Around Heart Valves: A Digital Computer Method for Solving the Equations of Motion*, Ph.D. thesis, Albert Einstein College of Medicine, New York, 1972.

[20] C. S. Peskin, *Numerical analysis of blood flow in the heart*, J. Comput. Phys., 25 (1977), pp. 220–252.

[21] C. S. Peskin and D. M. McQueen, *A three-dimensional computational method for blood flow in the heart: Immersed elastic fibers in a viscous incompressible fluid*, J. Comput. Phys., 81 (1989), pp. 372–405.

[22] C. S. Peskin and D. M. McQueen, *A general method for the computer simulation of biological systems interacting with fluids*, Symposia of the Society for Experimental Biology, 49 (1995), pp. 265–276.

[23] C. S. Peskin and D. M. McQueen, *Fluid dynamics of the heart and its valves*, in Case Studies in Mathematical Modeling—Ecology, Physiology, and Cell Biology, 1996, pp. 309–337.

[24] C. S. Peskin and B. F. Printz, *Improved volume conservation in the computation of flows with immersed elastic boundaries*, J. Comput. Phys., 105 (1993), pp. 33–46.

[25] M. T. Rudikoff, W. L. Maughan, M. Effron, P. Freund, and M. L. Weisfeldt, *Mechanisms of blood flow during cardiopulmonary resuscitation*, Circulation, 61 (1980), p. 345.

[26] H. Thomann, *A simple pumping mechanism in a valveless tube*, J. Appl. Math. Phys., 29 (1978), pp. 169–177.

[27] J. A. Werner, H. L. Greene, C. L. Janko, and L. A. Cobb, *Visualization of cardiac valve motion in man during external chest compression using two-dimensional echocardiography: Implications regarding the mechanism of blood flow*, Circulation, 63 (1981), pp. 1417–1421.

SIAM J. Sci. Comput.
Vol. 23, No. 1, pp. 46–65

TIME MARCHING MULTILEVEL TECHNIQUES FOR EVOLUTIONARY DISSIPATIVE PROBLEMS[*]

B. COSTA[†], L. DETTORI[‡], D. GOTTLIEB[§], AND R. TEMAM[¶]

Abstract. In this article we use a pseudospectral Fourier discretization in conjunction with a multilevel splitting of high and low modes to solve dissipative partial differential equations. We develop unconditionally stable explicit techniques for the temporal integration of the linear terms and apply them to the high modes equation, improving the overall temporal stability of the multilevel method and resulting in a competitive fully explicit numerical scheme for nonlinear problems. In the cases where the linear term determines the time step restriction, numerical experiments with the Burgers equation in one and two dimensions showed substantial CPU cost reduction when comparing the resulting method with the standard spectral collocation associated with regular temporal integration schemes.

Key words. Runge–Kutta methods, nonlinear Galerkin, Fourier collocation, Burgers equation

AMS subject classifications. 65M70, 65L06, 65L20, 76M25

PII. S1064827598339967

1. Introduction. For partial differential equations modeling dissipative problems, the large scale and the small scale components of the unknown play different roles in the dynamics of their solutions. When solving such problems with a multilevel scheme of the nonlinear Galerkin type (NLG) or with a more general collocation splitting, the unknown u is decomposed in the physical space into its low modes component y and high modes component z, and the governing equation is split into two equations, one containing only the low modes and the other containing only the high modes. This separation of modes allows the use of distinct time integration techniques for each equation.

Certain problems require a minimum number of modes to be contained in the numerical approximation of their solutions. However, increasing the number of modes in dissipative problems decreases the size of the time step when using explicit temporal integration. In this article, we show that by using the splitting of modes we are able to relax the stability constraint of the high modes equation, allowing the use of time steps larger than the ones permitted by the standard time integration of the original variable u.

In the original NLG method, the high frequency component z of the solution was computed as a function of the low frequency component y through the utilization of an approximate inertial manifold (see [21]). This has proven to yield very unstable

[*] Received by the editors June 8, 1998; accepted for publication (in revised form) January 8, 2001; published electronically May 10, 2001. The second and third authors were partly supported by AFORS grant F49620-96-10150 and NSF grant DMS-9500814.

http://www.siam.org/journals/sisc/23-1/33996.html

[†] Institute of Applied Mathematics, Indiana University, Bloomington, IN 47406 and Departamento de Matemática Aplicada, IM-UFRJ, Caixa Postal 68530, Rio de Janeiro, RJ, C.E.P. 21945-970, Brazil (bcosta@ufrj.br). This author was partly supported by the CNPq, Brazil, under grant 200431/93-5, by the National Science Foundation under grant NSF-DMS-9705229, and by the Research Fund of Indiana University.

[‡] School of CTI, DePaul University, 243 S. Wabash Avenue, Chicago, IL 60604 and Division of Applied Mathematics, Brown University, Providence, RI 02912 (ldettori@cs.depaul.edu).

[§] Division of Applied Mathematics, Brown University, Providence, RI 02912.

[¶] Departament of Mathematics & The Institute for Scientific Computing and Applied Mathematics, Indiana University, Bloomington, IN 47406.

schemes (see [19]). In the schemes that were subsequently considered (see [9]) the temporal variation term z_t was maintained, leading to much more stable discretizations than the original NLG with the same computational cost. Nevertheless, the essential idea of computing the low and high modes differently was retained, since their physical significances are different.

The collocation version of NLG was introduced in [6] and [7] for the Fourier and Chebyshev bases, respectively. There, the original approach of earlier papers in NLG (see [8] and [20]) was maintained. This approach, which consists in disregarding the nonlinear interactions of high and low modes and, sometimes, even the temporal evolution of the high modes, although leading to better accuracy with respect to the standard collocation method (SCM), would do so at a higher computational cost (see [13] and [14]). In [2] it was shown that by keeping all terms in both equations, the resulting splitting scheme would generate the same numerical solution as the SCM with equivalent computational effort. In this article we propose temporal evolution methods which integrate the low and high modes equations of the splitting scheme of [2] in different ways.

The general idea stems from the fact that high modes carry very little energy and, when computing steady state solutions, they evolve much faster than the low modes to an equilibrium state. This fast convergence at the transient stage of the temporal evolution requires small time intervals in the numerical integration process in order to capture the fast changes of the high modes. Therefore, artificially slowing this convergence to the same pace as the low modes enables the use of time intervals as large as the ones determined by the low modes only. Note that although we aim to approximate time dependent solutions, we believe that a proper approximation of the stationary solution is a necessary requirement for such a numerical scheme.

From the numerical point of view, the speed of convergence of the high modes is determined by the size of the corresponding eigenvalues of the dissipative operator. As hinted in [6], the size of these eigenvalues can be decreased by shifting the upper part of the spectrum through an exponential transformation in the z variable, hereafter referred to as the eigenvalues shifting technique. This involves the computation of a parameter, which determines the intensity of the shifting and can be chosen in a way of turning the high modes equation unconditionally stable. However, this technique has the drawback of altering the steady state solution of the high modes. A similar alternative way of shifting the spectrum that fixes this problem is proposed, and we call it the implicit correction technique. This technique, which is similar to previous ones developed in the study of stiff ODEs (see [10], [11], [17], [18], [22], and [24]), is applied through a modification of the temporal scheme, in our case, a Runge–Kutta scheme, allowing the use of a much larger time step. This method is consistent at steady state with the original method and has time accuracy at the transient stage.

A necessary observation is that we propose in this work a multilevel method for nonlinear problems, but since the stability analysis for these is mathematically too involved to be presented in this article, we will only apply the above techniques to the linear terms of the corresponding equations, relaxing only the time step restriction imposed by the dissipative linear operator. Nevertheless, the numerical results show computational advantages even in the cases of a weak dissipation.

This article is organized as follows. In section 2 we present the Runge–Kutta method modified by the eigenvalues shifting and analyze its stability and generation of steady state solutions. Section 3 presents the implicit correction technique and its application to some temporal integration schemes. Stability and error analysis for the modified schemes are also presented. Numerical experiments are shown in section 4.

2. The eigenvalues shifting technique. We start by considering the Burgers equation in one dimension, with periodic boundary conditions as below:

$$\begin{cases} u_t - \nu u_{xx} + \frac{1}{2}(u^2)_x = f, \quad x \in (0, 2\pi), \quad t > 0, \\ u(0,t) = u(2\pi, t), \quad t > 0. \end{cases} \tag{1}$$

The multilevel splitting procedure applied to this equation yields the following system for the low and high modes components y and z:

$$\begin{cases} y_t - \nu y_{xx} + \frac{1}{2} J_N((y+z)^2)_x = J_N f, \\ z_t - \nu z_{xx} + \frac{1}{2} G_M((y+z)^2)_x = G_M f, \end{cases} \tag{2}$$

where J_N and G_M are the projectors onto the low and high modes spaces, respectively (see [2] and [6]).

In the next two sections, we will introduce three modified time integration schemes to be applied in conjunction with the above decomposition. If the numerical solution contains $M = 2N$ modes, the y component corresponds to the first N modes and the z component to the last N modes. The stability condition imposed by the linear term of the low modes equation yields a time step proportional to $\frac{1}{N^2}$, while the time step for the high modes should be proportional to $\frac{1}{M^2} \ll \frac{1}{N^2}$. The idea of the methods presented below is to modify the time integration of the high modes equation in a way that reduces its stability constraint to that of the low modes equation allowing the choice of a much larger Δt. Thus, for the sake of simplicity, and since we are only interested in changing the time step restriction coming from the linear terms, we will consider for the numerical analysis that follows the linear equation

$$u_t = u_{xx} + f \tag{3}$$

with periodic boundary conditions and its corresponding multilevel splitting

$$\begin{cases} y_t = \nu y_{xx} + J_N f, \\ z_t = \nu z_{xx} + G_M f. \end{cases} \tag{4}$$

Nevertheless, the analysis that follows remains valid for nonlinear problems where the linear stability restriction is dominant. In these cases, the methods proposed in this section lead to fully explicit schemes with time steps much larger than the ones allowed by the standard explicit integration methods, as we shall see in the numerical experiments of section 4.

Consider a high modes function z in the form

$$z(x) = e^{iNx} \sum_{k=1}^{N} \hat{z}_k^M e^{ikx} \tag{5}$$

and the homogeneous linear equation

$$z_t = z_{xx}. \tag{6}$$

Multiplying (6) by $e^{\beta N^2 t}$, $\beta > 0$, and defining the new variable $w = e^{\beta N^2 t} z$, we have

$$w_t = w_{xx} + \beta N^2 w. \tag{7}$$

Substituting $\hat{w}_k = e^{\beta N^2 t}\hat{z}_k$ into (7), we obtain the following system of equations in the phase space:

$$\frac{d}{dt}\hat{w}_k = \lambda_k^\beta \hat{w}_k, \quad k = 1, \dots, N, \tag{8}$$

where

$$\lambda_k^\beta = -((N+k)^2 - \beta N^2), \quad k = 1, \dots, N. \tag{9}$$

To improve stability we need to choose a β that decreases the absolute values of the eigenvalues (9). As shown in [6], $\beta = \frac{5}{2}$ minimizes $\max_k \ |\lambda_k^\beta|$, yielding

$$\max_k \ |\lambda_k^{5/2}| = \frac{3}{2}N^2, \quad k = 1, \dots, N,$$

allowing the use of a time step proportional to $\frac{2}{3N^2}$. Hereafter, we denote this transformation of variables as the eigenvalues shifting (ES) technique.

A straightforward application of this idea to a temporal integration method consists in integrating the variable $\hat{w}_k$ for one iteration and recovering $\hat{z}_k$ at the end by setting

$$\hat{z}_k = e^{-\beta N^2 \Delta t}\hat{w}_k. \tag{10}$$

The choice of the parameter β will depend on the specific temporal discretization. In what follows, we apply the modification above to the fourth order Runge–Kutta (RK4) scheme and give an estimate for the parameter β.

Let us consider (6) in the phase space:

$$\frac{d}{dt}\hat{z}_k = \lambda_k^0 \hat{z}_k. \tag{11}$$

The standard RK4 scheme is given by

$$\begin{aligned}
&L\hat{z}_k^n = \lambda_k^0 \hat{z}_k^n, && K_1 = L\hat{z}_k^n,\\
&\hat{z}_k^{n+1/4} = \left(\hat{z}_k^n + \frac{\Delta t}{2}K_1\right), && K_2 = L\hat{z}_k^{n+1/4},\\
&\hat{z}_k^{n+2/4} = \left(\hat{z}_k^n + \frac{\Delta t}{2}K_2\right), && K_3 = L\hat{z}_k^{n+2/4},\\
&\hat{z}_k^{n+3/4} = \left(\hat{z}_k^n + \Delta t K_3\right), && K_4 = L\hat{z}_k^{n+3/4},\\
&\hat{z}_k^{n+1} = \left[\hat{z}_k^n + \frac{\Delta t}{6}(K_1 + 2(K_2+K_3) + K_4)\right]
\end{aligned} \tag{12}$$

and may be written in the form

$$\hat{z}_k^{n+1} = G(\lambda_k^0 \Delta t)\hat{z}_k^n, \tag{13}$$

where the amplification factor $G(\lambda_k^0 \Delta t)$ is given by

$$G(\lambda_k^0 \Delta t) = 1 + \lambda_k^0 \Delta t + \frac{(\lambda_k^0 \Delta t)^2}{2} + \frac{(\lambda_k^0 \Delta t)^3}{6} + \frac{(\lambda_k^0 \Delta t)^4}{24}. \tag{14}$$

If we apply the eigenvalues shifting to RK4 (ES–RK), we obtain

$$\tilde{z}_k^{n+1} = \tilde{G}(\lambda_k^\beta \Delta t)\tilde{z}_k^n, \tag{15}$$

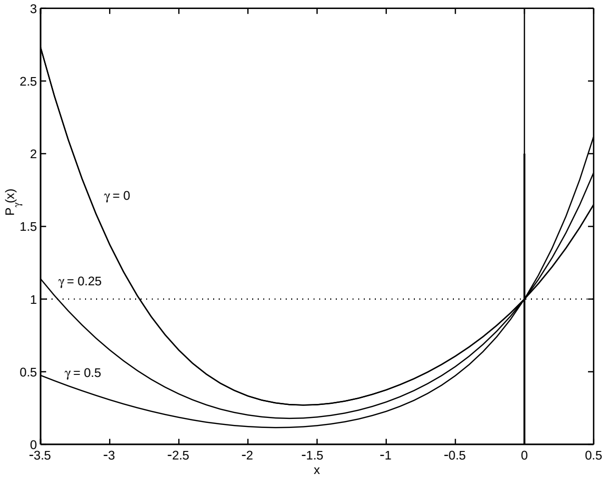


FIG. 1. *Graph of $P_\gamma(x)$.*

where the amplification factor $\tilde{G}(\lambda_k^\beta \Delta t)$ is given by

$$\tilde{G}(\lambda_k^\beta \Delta t) = \left[1 + \lambda_k^\beta \Delta t + \frac{(\lambda_k^\beta \Delta t)^2}{2} + \frac{(\lambda_k^\beta \Delta t)^3}{6} + \frac{(\lambda_k^\beta \Delta t)^4}{24}\right] e^{-\beta N^2 \Delta t}. \tag{16}$$

2.1. Stability analysis of ES–RK. In order to incorporate both methods in a common framework, we define the following family of functions:

$$\begin{cases} P(x) & = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24}, \\ P_\gamma(x) & = P(x)e^{\gamma x}. \end{cases} \tag{17}$$

Let us define the parameter

$$\gamma_k^\beta = \frac{-\beta N^2}{\lambda_k^\beta}. \tag{18}$$

Note that $\gamma_k^0 = 0$ and that γ_k^β increases when β increases.

Setting $x_k^\beta = \lambda_k^\beta \Delta t$, we can write the amplification factor of both methods as

$$G(\lambda_k^0 \Delta t) = P_0(x_k^0) = P_{\gamma_k^0}(x_k^0), \qquad \tilde{G}(\lambda_k^\beta \Delta t) = P_{\gamma_k^\beta}(x_k^\beta). \tag{19}$$

Figure 1 depicts the graphs of $P_\gamma(x)$ for increasing values of γ. The stability condition is determined by the first negative value of x such that $P_\gamma(x) = 1$. Note that increasing the value of β and therefore of γ increases the maximum absolute value of x_k^β that yields stability and therefore increases Δt. In the following we show that there is a value of β that makes the scheme unconditionally stable.

LEMMA 2.1. *Let the function $P_\gamma(x)$ be defined as in* (17). *There exist two real numbers $\gamma_b > 0$ and $\gamma_a < 0$ such that if $\gamma > \gamma_b$, then $P_\gamma(x) < 1$ for $x < 0$, and if $\gamma < \gamma_a$, then $P_\gamma(x) < 1$ for $x > 0$.*

Proof. The proof follows from the fact that $P(x) \leq e^{|x|}$. Therefore,

$$P_\gamma(x) = P(x)e^{\gamma x} \leq e^{\gamma x} e^{|x|},$$

yielding

$$\gamma \geq 1 = \gamma_b, \quad x \leq 0, \qquad \gamma \leq -1 = \gamma_a, \quad x \geq 0. \qquad \square$$

In light of the previous results, if we can find a value of β for which $\gamma_k^\beta \notin (\gamma_a, \gamma_b)$ for all k, the method (15) becomes unconditionally stable.

THEOREM 2.2. *For $\beta = \frac{3}{2}$, $\gamma_k^\beta \notin (\gamma_a, \gamma_b)$ for all $k = 1, \ldots, N$, and therefore the modified method is unconditionally stable for the high modes equation.*

Proof. Let us rewrite γ_k^β as

$$\gamma_k^\beta = \frac{\beta N^2}{(1-\beta)N^2 + 2kN + k^2}. \tag{20}$$

For a fixed positive β there exists $1 \leq k_0 \leq N$ such that

$$\begin{aligned} k < k_0 \quad &\Rightarrow \quad \gamma_k^\beta < 0, \\ k > k_0 \quad &\Rightarrow \quad \gamma_k^\beta > 0. \end{aligned}$$

Moreover, γ_k^β is increasing and decreasing in modulus for $k < k_0$ and $k > k_0$, respectively. Therefore, we just need to check that γ_1^β and γ_N^β are not in (γ_a, γ_b). The value $\beta = \frac{3}{2}$ is the one corresponding to $\gamma_N^\beta = \gamma_b$, and for this value of β, $\gamma_1^\beta \approx -2 < \gamma_a$. $\square$

REMARK 2.1. *The proposed modification shifts the upper part of the spectrum and causes the stability condition to be determined by the low modes. However, when considering the high mode solution $\hat{z}_k^n$ after n iterations*

$$\hat{z}_k^n = \left[\left(1 + \lambda_k^\beta \Delta t + \frac{(\lambda_k^\beta \Delta t)^2}{2} + \frac{(\lambda_k^\beta \Delta t)^3}{6} + \frac{(\lambda_k^\beta \Delta t)^4}{24}\right) e^{-\beta N^2 \Delta t}\right]^n \hat{z}_k^0, \tag{21}$$

one can easily see that the mode $\hat{z}_k^n$ converges exponentially to 0. *The scheme will therefore be inconsistent whenever the high modes of the exact solution do not go to* 0 *exponentially. This suggests that the exponential correction to retrieve the original variable z at the end of each iteration may be too strong. In the following section we propose a different correction that will eliminate this inconvenience.*

Further evidence of the inconsistency of this scheme is the fact that the steady state solution generated by ES–RK does not converge to the steady state solution of the original problem (3)

$$\hat{z}_k^{st} = -\frac{\hat{f}_k}{\lambda_k^0}. \tag{22}$$

For the sake of simplicity, consider the eigenvalues shifting for the forward Euler method:

$$\hat{z}_k^{n+1} = ((1 + \lambda_k^0 \Delta t + \beta N^2 \Delta t)\hat{z}_k^n + \Delta t \hat{f}_k) e^{-\beta N^2 \Delta t}. \tag{23}$$

The steady state solution for this method is

$$\hat{z}_k^{st} = \frac{\Delta t \hat{f}_k}{e^{\beta N^2 \Delta t} - (1 + \beta N^2 \Delta t) - \lambda_k^0 \Delta t}. \tag{24}$$

Note that, due to the desired stability condition, $\beta N^2 \Delta t = O(1)$; therefore, (24) is not a good approximation to (22).

3. The implicit correction technique. As pointed out in the previous section, the exponential correction to the Runge–Kutta scheme for the modified (8) is too strong and yields a different steady state solution from the original problem. This is due to the fact that applying an RK4 corresponds to applying a fourth order polynomial approximation of the exponential which we then counterbalance using a negative exponential at the end of each iteration.

The idea is to use a milder polynomial correction instead of an exponential one. In this section we illustrate the details of this idea first on the forward Euler scheme and later we extend it to higher order Runge–Kutta schemes.

3.1. Nonconsistent implicit correction. Let us consider the first iteration of the standard forward Euler scheme for the modified (8):

$$\hat{w}_k^1 = (1 + \lambda_k \Delta t + \beta N^2 \Delta t)\tilde{z}_k^0. \tag{25}$$

Note that the forward Euler scheme for the original equation can be retrieved by setting

$$\hat{z}^1 = \hat{w}_k^1 - \beta N^2 \Delta t \tilde{z}_k^0. \tag{26}$$

Instead, we propose the implicit correction

$$\tilde{z}_k^1 = \hat{w}_k^1 - \beta N^2 \Delta t \tilde{z}_k^1, \tag{27}$$

which leads to the modified forward Euler scheme for the nonhomogeneous equation:

$$\tilde{z}_k^{n+1} = \frac{1 + \lambda_k \Delta t + \beta N^2 \Delta t}{1 + \beta N^2 \Delta t} \tilde{z}_k^n + \frac{\Delta t}{1 + \beta N^2 \Delta t} \hat{f}_k. \tag{28}$$

Remark 3.1. *The scheme above, although obtained from the ES idea presented in the last section, can be classified in the more general class of rational Runge–Kutta schemes, which have been previously considered by ODE analysts in the study of stiff systems (see* [17], [18], *and* [24]*). Nevertheless, here we conjugate scheme* (28) *with the multilevel splitting method by applying it only to the high modes equation.*

The implicit correction technique differs from the ES in the substitution of the exponential term $e^{-\beta N^2 \Delta t}$ for the polynomial correction $(1 + \beta N^2 \Delta t)^{-1}$, which can be seen as a formal truncation in the Taylor series approximation to $e^{-\beta N^2 \Delta t}$. It is easy to check that the steady state solution of the modified Euler scheme coincides with the steady state solution of the original nonhomogeneous problem.

The following theorem shows that for certain values of the parameter β this polynomial correction is enough to improve the stability of the method or even make it unconditionally stable.

Theorem 3.1. *The modified forward Euler scheme* (28) *applied to the high modes equation in* (4) *with $\beta = 3/2$ is stable for*

$$\Delta t \leq \frac{2}{N^2}, \tag{29}$$

which is the same stability condition of the original forward Euler scheme applied to the low modes. Moreover, if $\beta = 2$, the scheme is unconditionally stable.

Proof. The scheme (28) is stable if

$$\frac{|1 + \lambda_k \Delta t + \beta N^2 \Delta t|}{|1 + \beta N^2 \Delta t|} \leq 1.$$

Knowing that $\lambda_k = -k^2$ and $\beta \geq 0$, we obtain the condition

$$\Delta t \leq \frac{2}{k^2 - 2\beta N^2}.$$

Considering the largest eigenvalue λ_{2N}, we have

$$\Delta t \leq \frac{2}{4N^2 - 2\beta N^2} = \frac{2}{(4-2\beta)N^2}.$$

Therefore, the choice of $\beta = 3/2$ yields (29), and if $\beta = 2$, the denominator vanishes, yielding unconditional stability. $\square$

REMARK 3.2. *When choosing $\Delta t = \frac{1}{N^2}$, as required by the stability condition for the low modes equation, it can be shown that the homogeneous version of the scheme is a discretization of the modified equation*

$$\frac{d}{dt}\tilde{z}_k = \frac{\lambda_k}{1+\beta}\tilde{z}_k, \tag{30}$$

which, in the physical space, corresponds to solving

$$(1+\beta)z_t = z_{xx}. \tag{31}$$

It is necessary to point out that adding the term βz_t in (31) *is the correct way of decreasing the viscosity without changing the steady state solution of* (6)*. However, this modification introduces an inconsistency and affects the order of the truncation error of the temporal scheme. In the next section we will consider a consistent implicit correction.*

In the following we compare the error committed by the standard forward Euler scheme,

$$\hat{u}_k^n = (1+\lambda_k\Delta t)^n \hat{u}_k^0 - \frac{1-(1+\lambda_k\Delta t)^n}{\lambda_k}\hat{f}_k, \tag{32}$$

with that committed by the modified forward Euler (28):

$$\tilde{u}_k^n = \left(1 + \frac{\lambda_k \Delta t}{1+\beta N^2\Delta t}\right)^n \hat{u}_k^0 - \frac{1-(1+\frac{\lambda_k\Delta t}{1+\beta N^2\Delta t})^n}{\lambda_k}\hat{f}_k. \tag{33}$$

Defining $\hat{\epsilon}_k^n = \tilde{u}_k^n - \hat{u}_k^n$ and considering that $\lambda_k = -k^2$, we obtain

$$\begin{aligned} |\hat{\epsilon}_k^n| &= \left| \left[\left(1 - \frac{k^2\Delta t}{1+\beta N^2 \Delta t}\right)^n - (1-k^2\Delta t)^n\right]\left(\hat{u}_k^0 - \frac{\hat{f}_k}{k^2}\right)\right| \\ &\leq \left|\left[k^2 n\Delta t\left(1 - \frac{1}{1+\beta N^2\Delta t}\right) + \cdots\right]\right| \left(|\hat{u}_k^0| + \frac{|\hat{f}_k|}{k^2}\right). \end{aligned} \tag{34}$$

Since $N^2\Delta t = O(1) = n\Delta t$, the term $(1 - \frac{1}{1+\beta N^2\Delta t}) = O(1)$, and we obtain

$$|\hat{\epsilon}_k^n| = O(k^2)|\hat{u}_k^0| + |\hat{f}_k|. \tag{35}$$

Let us consider, for example, f to be a continuous differentiable function, whose Fourier modes $\hat{f}_k$ decay as $O(\frac{1}{k^2})$, and an initial data u_0, whose Fourier modes decay as $\hat{u}_k^0 = O(\frac{1}{k^4})$. The contribution to the error in the low and high modes, respectively, is

$$(36)\left\{\begin{array}{l} O(\hat{u}_k^0) = O(f_k) = O(k^2) = O(1) \qquad \text{for } k \leq N, \\ O(\hat{u}_k^0) = O(\Delta t^2), O(\hat{f}_k) = O(\Delta t), \text{ and } O(k^2) = O(\Delta t^{-1}) \qquad \text{for } k > N. \end{array}\right.$$

Substituting this into (35), we get

$$|\epsilon_k^n| = \left\{\begin{array}{ll} O(1) & \text{for } k \leq N, \\ O(\Delta t) & \text{for } k > N. \end{array}\right. \tag{37}$$

This shows that for f with the required regularity, we will have an error of the first order in time if we apply the modification to the high modes equation and no convergence if we apply it to the low modes equation.

Scheme (28) is unconditionally stable even when applied to the nonsplit equation (3). However, (37) shows that no time accuracy is obtained when including the low modes in the modified scheme.

3.2. The nonconsistent implicit correction of RK2 and RK4. The extension of the method to a multistage scheme like a higher order Runge–Kutta is done following the guidelines of Remark 3.2, where the addition of the term βz_t is enforced at each stage. Thus, the nonconsistent implicit correction of the second order Runge–Kutta (NCIC2) for the nonhomogeneous version of (11) is the following:

$$\begin{array}{ll} \tilde{L}\tilde{z}_k^n = (\lambda_k^0 + \beta N^2)\tilde{z}_k^n, & \tilde{K}_1 = \tilde{L}\tilde{z}_k^n + \hat{f}_k, \\ \tilde{z}_k^{n+\frac{1}{2}} = \left(\tilde{z}_k^n + \dfrac{\Delta t}{2}\tilde{K}_1\right) / \left(1 + \beta N^2 \dfrac{\Delta t}{2}\right), & \tilde{K}_2 = \tilde{L}\tilde{z}_k^{n+\frac{1}{2}} + \hat{f}_k, \\ \tilde{z}_k^{n+1} = \left(\tilde{z}_k^n + \Delta t \tilde{K}_2\right) / (1 + \beta N^2 \Delta t). & \end{array} \tag{38}$$

As before, the steady state solution of scheme (38) converges to the original steady state solution, and the value $\beta = 1/2$ allows the use of the same time step of the low modes equation.

Finally, we introduce the nonconsistent implicit correction of the fourth order Runge–Kutta scheme (NCIC4):

$$\begin{array}{ll} \tilde{L}\tilde{z}_k^n = (\lambda_k^0 + \beta N^2)\tilde{z}_k^n, & \tilde{K}_1 = \tilde{L}\tilde{z}_k^n + \hat{f}_k, \\ \tilde{z}_k^{n+\frac{1}{4}} = \left(\tilde{z}_k^n + \dfrac{\Delta t}{2}\tilde{K}_1\right) / \left(1 + \beta N^2 \dfrac{\Delta t}{2}\right), & \tilde{K}_2 = \tilde{L}\tilde{z}_k^{n+\frac{1}{4}} + \hat{f}_k, \\ \tilde{z}_k^{n+\frac{2}{4}} = \left(\tilde{z}_k^n + \dfrac{\Delta t}{2}\tilde{K}_2\right) / \left(1 + \beta N^2 \dfrac{\Delta t}{2}\right), & \tilde{K}_3 = \tilde{L}\tilde{z}_k^{n+\frac{2}{4}} + \hat{f}_k, \\ \tilde{z}_k^{n+\frac{3}{4}} = \left(\tilde{z}_k^n + \Delta t \tilde{K}_3 +\right) / (1 + \beta N^2 \Delta t), & \tilde{K}_4 = \tilde{L}\tilde{z}_k^{n+\frac{3}{4}} + \hat{f}_k, \\ \tilde{z}_k^{n+1} = \left[\tilde{z}_k^n + \dfrac{\Delta t}{6}(\tilde{K}_1 + 2(\tilde{K}_2 + \tilde{K}_3) + \tilde{K}_4)\right] / (1 + \beta N^2 \Delta t). & \end{array} \tag{39}$$

The computation of the corresponding parameter β is more involved in this case; however, the discussion in section 2 hints at the correct range for β, since both methods apply the same idea. Numerical experiments indicate that for $\beta = 2$, unconditional stability is achieved for (39).

3.3. A consistent implicit correction. The idea that led to the NCIC can be improved to yield a consistent method. In this method we replace $\tilde{z}_k^0$ in (26) by $\frac{(\tilde{z}_k^1+\tilde{z}_k^{-1})}{2}$ and obtain

$$\tilde{z}_k^1 = \hat{w}_k^1 - \beta N^2 \Delta t \frac{(\tilde{z}_k^1 + \tilde{z}_k^{-1})}{2}. \tag{40}$$

This leads to the modified forward Euler scheme (CIC1)

$$\tilde{z}_k^{n+1} = (1+\lambda_k \Delta t)\tilde{z}_k^n - \frac{\beta N^2 \Delta t}{2}(\tilde{z}_k^{n+1} - 2\tilde{z}_k^n + \tilde{z}_k^{n-1}), \tag{41}$$

or, alternatively,

$$\tilde{z}_k^{n+1} = \frac{1+\lambda_k \Delta t}{1+\beta N^2 \frac{\Delta t}{2}}\tilde{z}_k^n + \frac{\beta N^2 \frac{\Delta t}{2}}{1+\beta N^2 \frac{\Delta t}{2}}(2\tilde{z}_k^n - \tilde{z}_k^{n-1}). \tag{42}$$

The stability analysis of the scheme above follows by defining a new variable $v^{n+1} = z^n$, transforming (42) into a system of equations whose amplification matrix G is given by

$$G(\lambda, \beta, \Delta t) = \begin{pmatrix} \frac{1+\lambda_k \Delta t + \beta N^2 \Delta t}{1+\beta N^2 \frac{\Delta t}{2}} & -\beta N^2 \frac{\Delta t}{2} \\ 1 & 0 \end{pmatrix}.$$

We will show that the eigenvalues of G are strictly less than 1 in magnitude for the high modes, and stability will follow. In order to simplify the calculations, we set $\lambda = \gamma N^2$, where $\gamma \in [1,2]$ and $\Delta t = \frac{2}{N^2}$, since we are only interested in stability for the set of high modes when using the low modes time step. Additionally, if, as before, we put $\beta = 2$, the matrix G above becomes

$$G = \begin{pmatrix} \frac{5-2\gamma^2}{3} & -\frac{2}{3} \\ 1 & 0 \end{pmatrix}$$

with its eigenvalues given by

$$\alpha_\gamma = \frac{a \pm \sqrt{a^2+4b}}{2},$$

where $a = \frac{5-2\gamma^2}{3}$ and $b = -\frac{2}{3}$. When $\gamma \in [1,2]$, it is easily proven that $a^2 + 4b < 0$; therefore,

$$|\alpha_\gamma|^2 < 1.$$

It is easily shown that the steady state solution of the CIC1 scheme (42) converges to the original steady state solution.

The truncation error of scheme (42) is given by

$$O(\Delta t) + O((\Delta t N)^2 z_{tt}). \tag{43}$$

The first term comes from the forward Euler scheme, and the second is the one associated with the implicit correction (40). Differently from scheme (28), scheme (42) possesses time accuracy independently of the particular set of modes being solved. If $\Delta t N^2$ is fixed (O(1)), then the second term in (43) is $O(\Delta t z_{tt})$.

REMARK 3.3. *Observe for* (43) *that z_{tt} is not small in the sup norm, but it is small in average. Hence there is here a new issue in numerical analysis on how to understand and define truncation error accuracy for high modes.*

REMARK 3.4. *The CIC scheme is similar to the Du Fort–Frankel scheme (see* [4], [15]*, and* [16]*)*

$$u^{n+1} = u^{n-1} + 2\Delta t \delta^2 u^n - \gamma N^2 \Delta t (u_j^{n+1} - 2u_j^n + u_j^{n-1}), \tag{44}$$

*where δ^2 represents the discrete second derivative operator and γ is a coefficient playing the same role as β for the previous schemes. The fundamental difference is in the way the first derivative in time is discretized. The Du Fort–Frankel scheme uses a central approximation resulting in a nondissipative scheme, not well suited for the parabolic equations under consideration, while in the CIC*1 *a forward approximation is used.*

By applying the consistent implicit correction at each stage of the fourth order Runge–Kutta we obtain the CIC scheme of order 4 (CIC4):

$$\begin{aligned}
\tilde{L}\tilde{z}_k^n &= (\lambda_k^0 + \beta N^2)\tilde{z}_k^n, & \tilde{K}_1 &= \tilde{L}\tilde{z}_k^n - \tfrac{\beta N^2}{2}\tilde{z}_k^{(n-1)+\frac{2}{4}} + \hat{f}_k,\\
\tilde{z}_k^{n+\frac{1}{4}} &= \left(\tilde{z}_k^n + \frac{\Delta t}{2}\tilde{K}_1\right) \Big/ \left(1 + \beta N^2 \frac{\Delta t}{2}\right), & \tilde{K}_2 &= \tilde{L}\tilde{z}_k^{n+\frac{1}{4}} - \frac{\beta N^2}{2}\tilde{z}_k^n + \hat{f}_k,\\
\tilde{z}_k^{n+\frac{2}{4}} &= \left(\tilde{z}_k^n + \frac{\Delta t}{2}\tilde{K}_2\right) \Big/ \left(1 + \beta N^2 \frac{\Delta t}{2}\right), & \tilde{K}_3 &= \tilde{L}\tilde{z}_k^{n+\frac{2}{4}} - \frac{\beta N^2}{2}\tilde{z}_k^n + \hat{f}_k,\\
\tilde{z}_k^{n+\frac{3}{4}} &= \left(\tilde{z}_k^n + \Delta t\tilde{K}_3\right) / \left(1 + \beta N^2 \Delta t\right), & \tilde{K}_4 &= \tilde{L}\tilde{z}_k^{n+\frac{3}{4}} - \frac{\beta N^2}{2}\tilde{z}_k^n + \hat{f}_k,\\
\tilde{z}_k^{n+1} &= \left[\tilde{z}_k^n + \frac{\Delta t}{6}(\tilde{K}_1 + 2(\tilde{K}_2 + \tilde{K}_3) + \tilde{K}_4)\right] / (1 + \beta N^2 \Delta t).
\end{aligned} \tag{45}$$

3.4. The nonlinear case. The analysis presented above is still valid for nonlinear problems where the stiffness of the linear dissipative operator determines the stability restriction on the time step, in particular, problems with a high value for the viscosity parameter ν. For those where the choice of the Δt is also influenced by the Courant, Friedrichs, and Lewy (CFL) condition, an extra modification in the implicit correction is necessary in order to control the numerical instability coming from the nonlinear advective terms. This is the subject of a forthcoming work. Here we limit ourselves to see the numerical advantages obtained when the above schemes relax the stability restrictions of the linear operator without recurring to implicit integration.

Thus, in the next section, when applying the implicit correction schemes to nonlinear problems, the nonlinear terms will be treated explicitly as in the standard Runge–Kutta schemes. All the gain in the size of the time step will come from the weaker stability restriction of the linear term obtained with the new schemes. The resulting scheme when applying the modified forward Euler scheme (28) to the nonlinear system (2) is given by

$$\left\{\begin{array}{l}
y^{n+1} = y^n + \Delta t(\nu y_{xx}^n + \frac{1}{2}J_N((y^n + z^n)^2) + J_N f^n),\\
z^{n+1} = \frac{\Delta t}{1+\beta N^2 \Delta t}(z^n + \nu z_{xx}^n + \beta N^2 \Delta t z^n) + \frac{\Delta t}{1+\beta N^2 \Delta t}(\frac{1}{2}G_M(y^n + z^n)_x^2 + G_M f^n).
\end{array}\right. \tag{46}$$

The second equation of the nonlinear system above is obtained from the linear scheme (28) by adjoining the nonlinear term to the forcing term. For the sake of completeness we include below the fourth order scheme corresponding to (39) when applied to the high modes equation of the nonlinear system (2):

$$
\begin{aligned}
&\tilde{L}\tilde{z}_k^n = (\lambda_k^0 + \beta N^2)\tilde{z}_k^n, && \tilde{K}_1 = \tilde{L}\tilde{z}_k^n + \hat{\mathrm{NL}}_k^{\,n} + \hat{f}_k,\\
&\tilde{z}_k^{n+\frac{1}{4}} = \left(\tilde{z}_k^n + \frac{\Delta t}{2}\tilde{K}_1\right) / \left(1 + \beta N^2 \frac{\Delta t}{2}\right), && \tilde{K}_2 = \tilde{L}\tilde{z}_k^{n+\frac{1}{4}} + \hat{\mathrm{NL}}_k^{\,n+\frac{1}{4}} + \hat{f}_k,\\
&\tilde{z}_k^{n+\frac{2}{4}} = \left(\tilde{z}_k^n + \frac{\Delta t}{2}\tilde{K}_2\right) / \left(1 + \beta N^2 \frac{\Delta t}{2}\right), && \tilde{K}_3 = \tilde{L}\tilde{z}_k^{n+\frac{2}{4}} + \hat{\mathrm{NL}}_k^{\,n+\frac{2}{4}} + \hat{f}_k,\\
&\tilde{z}_k^{n+\frac{3}{4}} = \left(\tilde{z}_k^n + \Delta t\tilde{K}_3 +\right) /(1 + \beta N^2 \Delta t), && \tilde{K}_4 = \tilde{L}\tilde{z}_k^{n+\frac{3}{4}} + \hat{\mathrm{NL}}_k^{\,n+\frac{3}{4}} + \hat{f}_k,\\
&\tilde{z}_k^{n+1} = \left[\tilde{z}_k^n + \frac{\Delta t}{6}(\tilde{K}_1 + 2(\tilde{K}_2 + \tilde{K}_3) + \tilde{K}_4)\right] /(1 + \beta N^2 \Delta t), &&
\end{aligned}
\tag{47}
$$

where

$$NL^n = \left(\frac{1}{2}G_M(y^n + z^n)_x^2\right).$$

One obtains the higher order schemes for the nonlinear case corresponding to (38) and (45) in an analogous way.

4. Numerical experiments.

In what follows we will refer to the SCM with the standard fourth order Runge–Kutta as SCM4. The collocation splitting coupled with the nonconsistent and consistent implicit corrections of the fourth order Runge–Kutta schemes ((39) and (45)) will be denoted by NCIC4 and CIC4, respectively.

The parameter β is set to 2 in order to yield unconditional stability for the high modes. The time step is written in the form

$$\Delta t = \frac{C}{\nu M^2}, \tag{48}$$

where the constant C is the parameter determining the size of Δt.

It is shown in [2] that the computational cost per iteration is the same when solving (3) with the SCM or the collocation splitting. Since the modification in the Runge–Kutta methods does not cause any relevant increase in the number of flops with respect to the original Runge–Kutta method, the constant C, and therefore the number of iterations to achieve a final time T, can also be used as a measure of the computational effort for comparison purposes. In this paper, we are interested only in measuring the gain in computational time by the use of a bigger Δt. It is shown in [2] that by just applying the splitting, CPU effort reduction can be achieved; however, this might also depend on the particular equation being solved.

Remark 4.1. *As was mentioned in the previous section, the following numerical experiments deal only with problems where the linear stability restriction on the time step is stronger than the CFL condition arising from the nonlinear advective terms. More precisely, since the nonlinear terms in scheme* (46) *are treated explicitly, the quantity*

$$\frac{\Delta x_{\min}}{||u||_\infty} \tag{49}$$

must be larger than (48) *at all times. For the dissipative problems below, it is sufficient that the initial data satisfies the above condition. However, depending on the strength of the nonlinearity, the time step bound* (49) *must also be checked during temporal integration to avoid blow-up of solutions.*

4.1. The one dimensional case. We start by solving the linear equation

$$\left\{ \begin{array}{l} u_t - \nu u_{xx} = f, \quad x \in (0, 2\pi), \quad t > 0, \\ u(0,t) = u(2\pi, t), \quad t > 0, \end{array} \right. \tag{50}$$

with a right-hand side of the form

$$f(x,t) = \cos(\nu\alpha t) \sum_{k=1}^{M} \frac{\cos(kx)}{k^2}. \tag{51}$$

The exact solution is

$$u(x,t) = \sum_{k=1}^{M} \left[c_k e^{-k^2 t} + \frac{\nu \cos(\alpha t)}{\nu(\alpha^2 + k^4)} + \frac{\alpha \sin(\nu \alpha t)}{\nu k^2 (\alpha^2 + k^4)} \right] \cos(kx), \tag{52}$$

where $c_k = O(\frac{1}{k^4})$. We set $\alpha = 0.01$ (providing a small variation in time) and $M = 33$.

Figures 2 and 3 show the L^2 error generated by SCM4, NCIC4, and CIC4 for $\nu = 0.1$ and $\nu = 0.001$, respectively. We took $C = 2$ in (48) when using SCM4. (The maximum value of C for SCM4 is 2.8; see section 2.) However, when using the modified methods NCIC4 and CIC4, C was taken equal to 8, a time step four times larger. Since we are considering 33 modes in (52), a minimum of 66 points is necessary to completely represent low, $k \leq 16$, and high modes, $k > 17$. Note that the multilevel methods generate intermediate solutions between SCM4 with 34 and 66 points, but at a lower computational cost than this last one since they use a bigger Δt.

Another interesting aspect of Figures 2 and 3 is the different behavior of NCIC4 and CIC4 with regard to the temporal stage of the solution. We see that NCIC4 presents a smaller error during the transient phase. This can be understood by analyzing the special form of the solution (52). At the beginning of the temporal evolution, the dominating term in (52) is the negative exponential, which represents the transient part of $u(x,t)$. Therefore, for small t we can write that

$$z_{tt} = O(N^2 z_t).$$

This makes both NCIC4 and CIC4 of the same order whenever a strong dissipation is dominating the evolution of z (see Remark 3.2 and (43)). Figure 4 shows the graph of the numerical values of z_t and $\Delta t z_{tt}$ for $\nu = 0.001$ and confirms the preceding analysis.

In the next example we solve the full Burgers equation

$$\left\{ \begin{array}{l} u_t - \nu u_{xx} + \frac{1}{2}(u^2)_x = f, \quad x \in (0, 2\pi), \quad t > 0, \\ u(0,t) = u(2\pi, t), \quad t > 0, \end{array} \right. \tag{53}$$

with an exact solution of the form

$$u(x,t) = \sum_{k=1}^{16} (1 - e^{-\nu k^2 t}) \frac{\cos(kx)}{k^4}. \tag{54}$$

We applied SCM4 with two grids containing 34 and 66 points. Due to the nonlinearity, only this last one solves the problem without aliasing. The solution was computed up to $t = 25$ with $\nu = 0.1$ and $C = 2$. For the modified methods, NCIC4 and CIC4, we used the 66 points grid with $C = 8$.

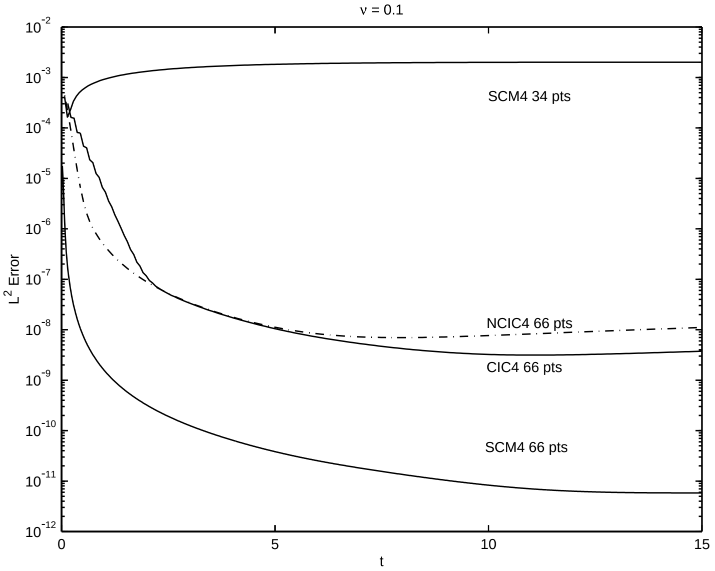


FIG. 2. *Comparison between* SCM4, NCIC4, *and* CIC4 *for* $\nu = 0.1$.

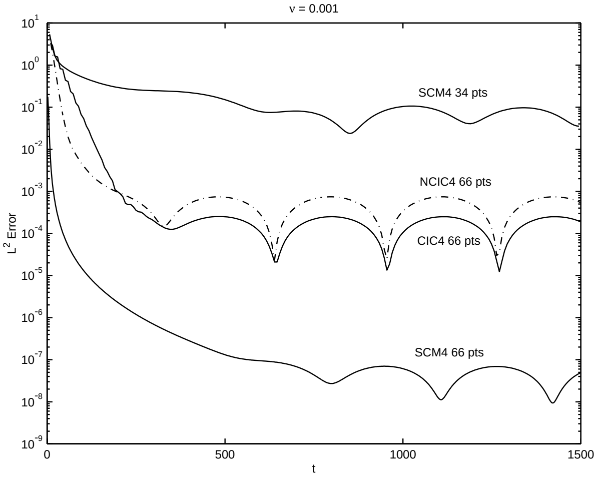


FIG. 3. *Comparison between* SCM4, NCIC4, *and* CIC4 *for* $\nu = 0.001$.

Figure 5 shows the L^2 error results for these experiments. Note that the high modes interactions, which are not captured by the 34 points grid, are relevant for the correct solution at the steady state. NCIC4 and CIC4 achieved the same steady state with the bigger time step. Now, CIC4 presents a better result than NCIC4 at the transient stage due to its consistency.

Finally, Figure 6 shows that although the modified schemes achieved the steady state at a later value of t, they did so at a smaller computational cost than SCM4 due to the utilization of the bigger Δt. The little discrepancy observed at the steady state of the 66 points grid is due to the better roundoff error presented by the modified methods (see [2, section 4]).

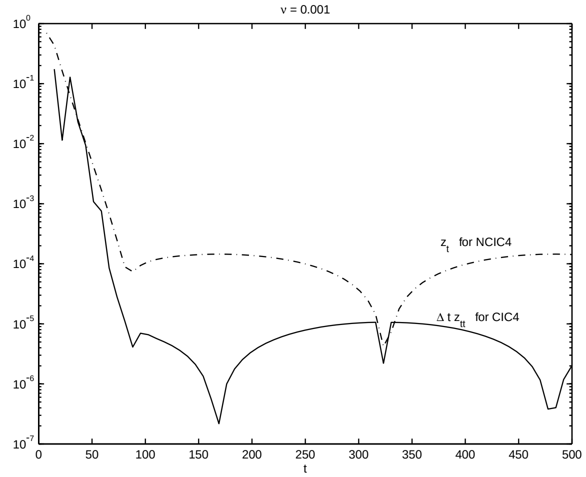


FIG. 4. *Comparison between the numerical values of z_t when using* NCIC4 *and $\Delta t z_{tt}$ when using* CIC4: *(z_t) dashed line; ($\Delta t z_{tt}$) continuous line.*

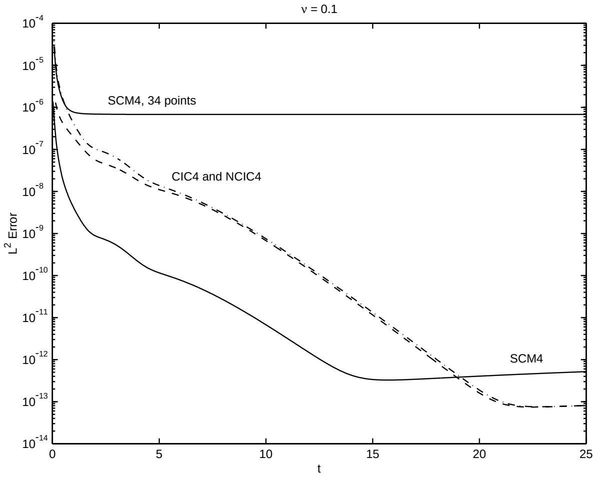


FIG. 5. *L^2 error for the Burgers equation.* (SCM4) *continuous line;* (CIC4) *dashed line;* (NCIC4) *dash–dotted line.*

4.2. The two dimensional case. In the next example we solve the periodic Burgers equation in two dimensions:

$$\begin{cases} U_t - \nu \Delta U + (U \cdot \nabla) U = 0, & x \in [0, 2\pi]^2, \\ U(0,t) = U(2\pi, t), \quad t > 0, \end{cases} \tag{55}$$

where $U = (u, v)$ with the initial conditions

$$\begin{cases} u(0,x,y) = \sin(x)\cos(3y), \\ v(0,x,y) = \cos(3x)\cos(y). \end{cases} \tag{56}$$

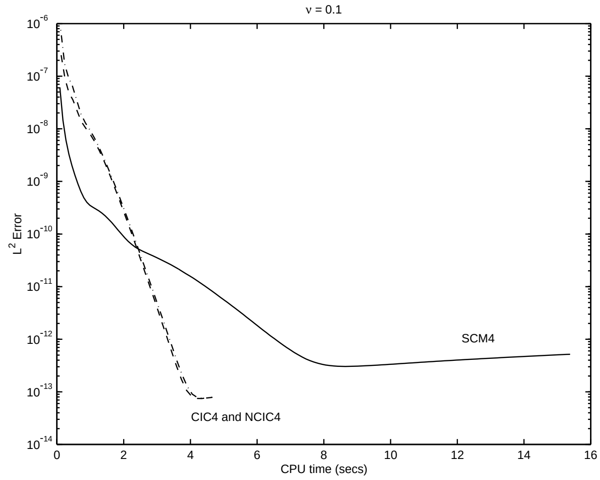


FIG. 6. *CPU time for the Burgers equation.* (SCM4) *continuous line;* (CIC4) *dashed line;* (NCIC4) *dash–dotted line.*

As shown in [2], in the two dimensional case each component of the unknown U is split in four quantities w, z_1, z_2, and z_3, depending on the mode $\vec{k} = (k_1, k_2)$:

$$\begin{array}{lll}
\text{low modes } w, & |k_1| \le N, & |k_2| \le N, \\
\text{mixed modes } z_1, & N < |k_1| \le M, & |k_2| \le N, \\
\text{mixed modes } z_2, & |k_1| \le N, & N < |k_2| \le M, \\
\text{pure high modes } z_3, & N < |k_1| \le M, & N < |k_2| \le M.
\end{array}$$

In this case, we apply the modified methods above to the three resulting equations involving the z variables. The determination of the parameter β_i for each set of modes follows along the same lines as in the one dimensional case.

In the first example we consider a high viscosity problem, $\nu = 0.5$, and compare the SCM coupled with a forward Euler scheme for the time integration (SCM1) with the multilevel method coupled with the nonconsistent (NCIC1) and the consistent (CIC1) implicit corrections. Figure 7 shows the SCM1 solution at $t = 1$ when using ten Fourier modes in each direction for the solution representation. The time step in (48) was taken with $C = 1$. Figure 8 shows the NCIC1 and CIC1 solutions at the same time and same number of modes, but now using $C = 4$, i.e., we use a time step four times bigger than the SCM. At $t = 1$ the problem is still in its transient stage, and we notice the difference between the solutions in Figure 8. The parameters β_i were taken to be $\beta_i = 4$, $i = 1, 2, 3$, for both NCIC1 and CIC1.

Figure 9 shows the solutions at $t = 4$ for all methods above. Here NCIC1 and CIC1 display the same graphic solutions. On the other hand, since these last two schemes used a bigger time step than SCM1, their computational costs are much lower. Figure 10 presents the CPU time results for the three methods when integrating in time up to $t = 10$.

In the case of a lower viscosity, the stability constraint is influenced by the advective term of (55). Therefore, increasing the time step requires a modification also in the nonlinear part of the equation. Since this is out of the scope of this article, we want to show instead that the splitting of modes can also reduce computational costs by generating solutions with fewer modes than the minimum resolution required by

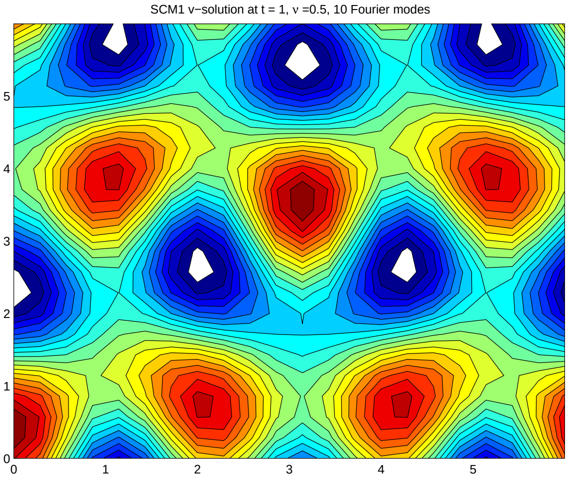


FIG. 7. SCM1 *v-solution for* (55) *at* $t = 1$ *and* $\nu = 0.5$.

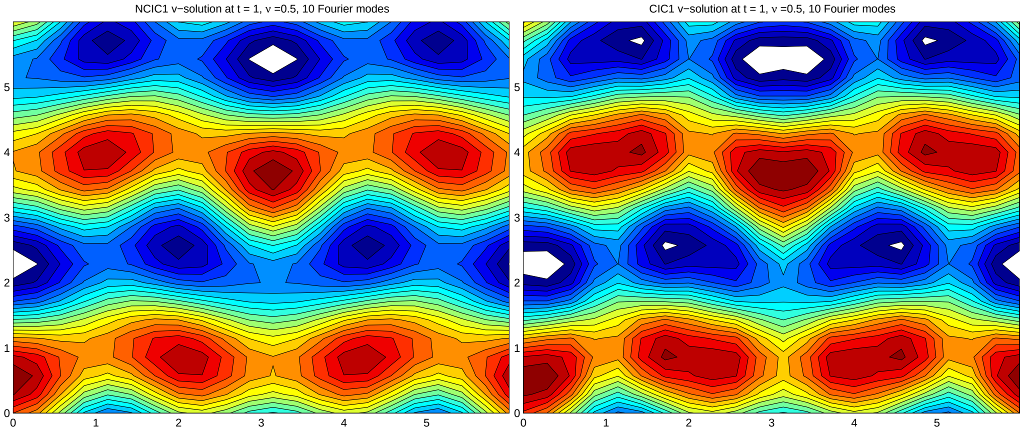


FIG. 8. NCIC1 *and* CIC1 *v-solutions for* (55) *at* $t = 1$ *and* $\nu = 0.5$.

the standard collocation in order to avoid blow-up due to the accumulation of energy in the high modes.

Thus, in this second example, we compare the solutions obtained by SCM1 and NCIC1 when solving problem (55), (56) with a value of $\nu = 0.01$. Due to the low viscosity, the solution goes through a critical high gradient phase before being smoothed out by the dissipative term. When using SCM1, a minimum of 80 Fourier modes in each direction is necessary to pass the high gradient stage, which occurs at $t = 1$, without the blow-up of the solutions. The results are shown in Figure 11 for $t = 1$ and $t = 6$. On the other hand, when using NCIC1, only 40 modes are necessary to overcome the high gradients and continue the temporal integration achieving the results shown in Figure 12 for $t = 1$ and $t = 6$.

5. Conclusions. In this article we introduced new time marching techniques that make use of distinguished treatments of low and high modes to improve the stability condition of explicit time integration schemes, allowing the use of larger time

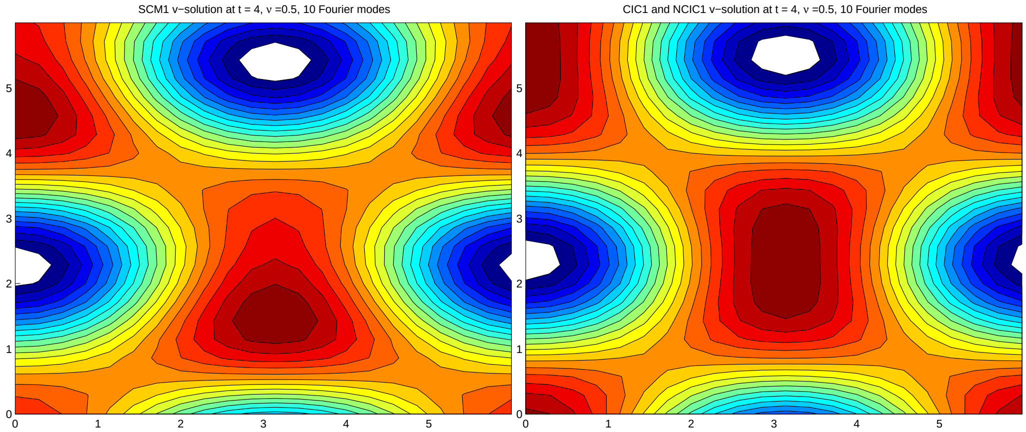


FIG. 9. SCM1, NCIC1, *and* CIC1 *solutions for* (55) *at $t = 4$ and $\nu = 0.5$.*

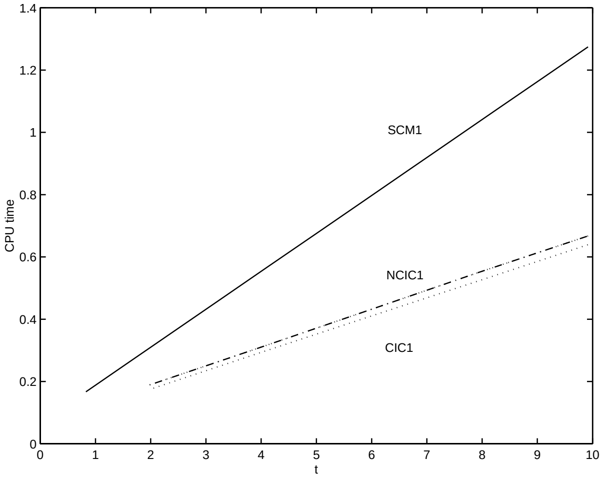


FIG. 10. *CPU time for* (55).

steps. These techniques consist of applying an unconditionally stable explicit scheme to the high modes equation in order to use the larger time step determined by the low modes equation. They are based on a shifting of the upper part of the spectrum of the dissipative operator and they bear similarities with well-known explicit numerical schemes like the Du Fort–Frankel and rational Runge–Kutta methods.

The numerical experiments showed that association of the new techniques with spatial discretizations of the collocation type can substantially reduce the computational effort in numerically approximating the solution of partial differential equations due to the utilization of a larger step to march in time. In this article, we altered only the stability condition related to the linear dissipative operator. In a forthcoming work we intend to treat the nonlinear hyperbolic case where the numerical stability is strongly influenced by the CFL condition when dealing with low viscosity problems.

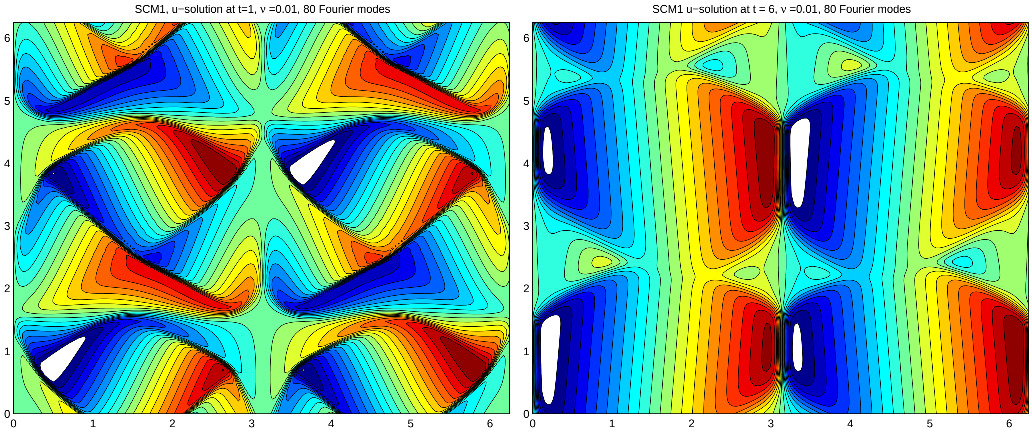


FIG. 11. SCM1 *solution for* (55) *at* $t = 1$ *and* $t = 6$ *with* $\nu = 0.01$ *and* 80 *Fourier modes in each direction.*

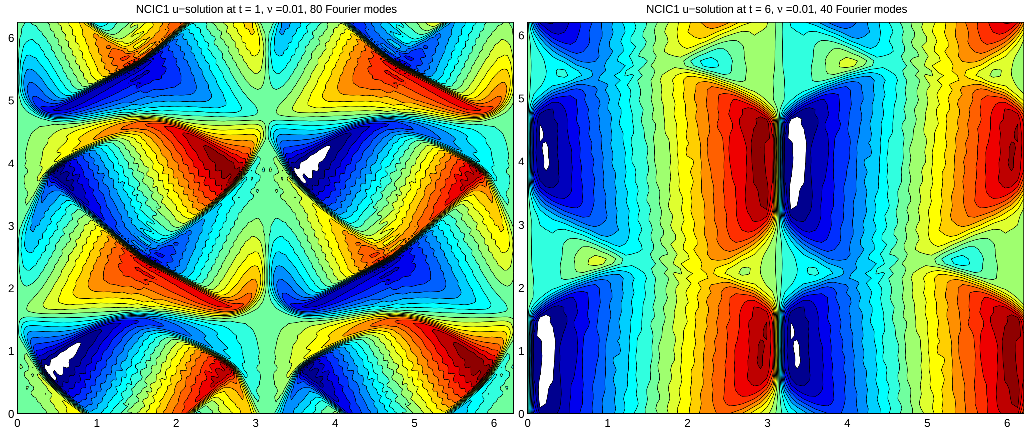


FIG. 12. NCIC1 *solution for* (55) *at* $t = 1$ *and* $t = 6$ *with* $\nu = 0.01$ *and* 40 *Fourier modes in each direction.*

REFERENCES

[1] C. CANUTO, M. Y. HUSSAINI, A. QUARTERONI, AND T. A. ZANG, *Spectral Methods in Fluid Dynamics*, Springer-Verlag, New York, 1988.

[2] B. COSTA AND L. DETTORI, *Fourier collocation splittings for the solution of partial differential equations*, J. Comput. Phys., 142 (1998), pp. 562–580.

[3] D. R. DURRAM, *Numerical Methods for Wave Equations in Geophysical Fluid Dynamics*, Texts Appl. Math. 32, Springer-Verlag, New York, 1999.

[4] E. C. DU FORT AND S. P. FRANKEL, *Stability conditions in the numerical treatment of parabolic differential equations*, Math. Tables and Other Aids to Computation, 7 (1953), pp. 135–152.

[5] P. J. DAVIS AND P. RABINOVITZ, *Methods of Numerical Integration*, Academic Press, New York, 1975.

[6] L. DETTORI, D. GOTTLIEB, AND R. TEMAM, *A nonlinear Galerkin Method: The two-level Fourier-collocation case*, J. Sci. Comput., 10 (1995), pp. 371–389.

[7] L. DETTORI, D. GOTTLIEB, AND R. TEMAM, *A nonlinear Galerkin Method: The Two-Level Chebyshev-Collocation Case*, A. V. Ilin and L. R. Scott, eds., special issue of Houston J. Math., 1996, pp. 75–83.

[8] T. Dubois, F. Jauberteau, and R. Temam, *Solution of the incompressible Navier-Stokes equations by the nonlinear Galerkin method*, J. Sci. Comput., 8 (1993), pp. 167–194.

[9] T. Dubois, F. Jauberteau, and R. Temam, *Dynamic Multilevel Methods and the Numerical Simulation of Turbulence*, Cambridge University Press, Cambridge, UK, 1998.

[10] K. Dekker and J. G. Verwer, *Stability of Runge-Kutta Methods for Stiff Nonlinear Differential Equations*, North-Holland, Amsterdam, 1984.

[11] C. W. Gear, *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice Hall, Englewood Cliffs, NJ, 1971.

[12] D. Gottlieb, M. Y. Hussaini, and S. A. Orzag, *Theory and applications of spectral methods*, in Spectral Methods for Partial Differential Equations, R. Voigt, D. Gottlieb, and M. Y. Hussaini, eds., SIAM, Philadelphia, 1984, pp. 1–94.

[13] B. García-Archilla, J. Novo, and E. S. Titi, *Postprocessing the Galerkin method: A novel approach to approximate inertial manifolds*, SIAM J. Numer. Anal., 35 (1998), pp. 941–972.

[14] B. García-Archilla and J. de Frutos, *Time integration of the non-linear Galerkin method*, IMA J. Numer. Anal., 15 (1995), pp. 221–244.

[15] D. Gottlieb and B. Gustafsson, *Generalized Du Fort–Frankel methods for parabolic initial-boundary value problems*, SIAM J. Numer. Anal., 13 (1976), pp. 129–144.

[16] D. Gottlieb and L. Lustman, *The DuFort-Frankel Chebyshev Method for Parabolic Initial Boundary Value Problems*, ICASE Report 81-42, ICASE, Hampton, VA, 1981.

[17] E. Hairer, *Unconditionally stable methods for second order differential equations*, Numer. Math., 32 (1979), pp. 373–379.

[18] E. Hairer, *Unconditionally stable methods for parabolic differential equations*, Numer. Math., 35 (1980), pp. 57–68.

[19] F. Jauberteau, *Résolution numérique des équations de Navier-Stokes instationnaires por méthods spectrales. Méthode de Galerkin non linéaire*, Thesis, Université de Paris-Sud, Paris, France, 1990.

[20] F. Jauberteau, C. Rosier, and R. Temam, *The nonlinear Galerkin method in computational fluid dynamics*, Appl. Numer. Math., 6 (1989/90), pp. 361–370.

[21] M. Marion and R. Temam, *Nonlinear Galerkin methods*, SIAM J. Numer. Anal., 26 (1989), pp. 1139–1157.

[22] L. R. Petzold, L. O. Jay, and J. Yen, *Numerical solution of highly oscillatory ordinary differential equations*, in Acta Numerica 1997, Acta Numer. 6, Cambridge University Press, Cambridge, UK, 1997, pp. 437–483.

[23] R. Temam, *Multilevel methods for the simulation of turbulence*, J. Comput. Phys., 127 (1996), pp. 309–315.

[24] A. Wambecq, *Rational Runge-Kutta method for solving systems of ordinary differential equations*, Computing, 20 (1978), pp. 333–342.

SIAM J. Sci. Comput.
Vol. 23, No. 1, pp. 66–80

A NEW TRIANGULAR FINITE-ELEMENT WITH OPTIMUM CONSTRAINT RATIO FOR COMPRESSIBLE FLUIDS[*]

DANIEL Y. LE ROUX[†]

Abstract. The discretization of the shallow-water equations using the finite-element method is a delicate problem. Apart from the possible occurrence of pressure and/or velocity modes, other spurious modes may appear that are essentially a consequence of having more momentum than continuity discretized equations, contrary to the continuum case. In this paper a new triangular finite-element pair is proposed which overcomes this imbalance problem. The new pair is shown to improve on results obtained with existing pairs in representing the propagation of fast gravity and slow Rossby waves by discretizing the linear shallow-water equations.

Key words. finite elements, compressible fluids, gravity waves, Rossby waves

AMS subject classifications. 65N30, 76N10, 76B15, 76C20

PII. S1064827500367403

1. Introduction.

In the case of the compressible Navier–Stokes and shallow-water equations, the contribution of the time derivative in the continuity equations leads, at least theoretically, to a problem that is always stable without an inf-sup (or LBB) [2] condition. An important result obtained in [11, 12], in the context of the incompressible Navier–Stokes equations, is, however, generalized in [14] to include the shallow-water and the compressible Navier–Stokes equations by examining their numerical dispersion relations. It is found that spurious solutions may arise from the coupling of the momentum and continuity equations, and that their existence and behavior depend upon the placement of the variables on a mesh and upon the choice of appropriate basis functions for finite-element formulations. These spurious modes are small-scale artifacts introduced by the spatial discretization scheme which do not propagate but are trapped within the model grid. In [14] two basic sets of such spurious modes are described. For the first set the velocity field is zero and nonconstant pressure functions lie in the null space of the discrete gradient operator. Solution uniqueness is then lost since any multiple of a spurious mode can be added to any solution of the discrete equations and still satisfy them. The second set of possible modes are those for which the pressure is zero and the velocity field is in the null space of the discrete divergence operator. Having noted their possible existence, little else is said about them in [14]; attention is focused almost exclusively on the spurious pressure modes of zero velocity, since these are argued to be the most troublesome. The occurrence of such spurious pressure and/or velocity modes has been observed in a variety of finite-difference [15] and finite-element [14] approximations to the shallow-water equations.

Another difficulty comes from the so-called constraint ratio (CR), defined to be the ratio of the number of continuity equations to the number of vector momentum equations. In the continuum, at each point in the fluid, one vector momentum equation is balanced with one continuity equation; hence CR = 1. A desirable goal of the

[*] Received by the editors January 3, 2000; accepted for publication (in revised form) December 18, 2000; published electronically June 5, 2001.
http://www.siam.org/journals/sisc/23-1/36740.html

[†] National Center for Atmospheric Research, Advanced Study Program, P.O. Box 3000, Boulder, CO 80307-3000 (dleroux@ucar.edu). NCAR is sponsored by the National Science Foundation.

discrete approximation would be to achieve the same balance. For most of the finite-difference schemes the balance is obtained and CR $\simeq 1$. A valuable example, however, is the C-D finite-difference grid [1] leading to CR $\simeq 1/2$. The dispersion relation obtained in [1] by discretizing the shallow-water equations on the C-D grid reveals the presence of spurious frequencies; they are essentially a consequence of having twice as many momentum equations as pressure equations. The use of the finite-element method in discretizing the shallow-water equations is even more problematic because for many popular finite-element pairs, e.g., the $P_2 - P_1$, $P_1 \,\mathrm{iso}\, P_2 - P_1$, and $Q_2 - Q_1$ pairs, CR $\simeq 1/4$ [9]; hence spurious frequencies are highly expected.

The purpose of this paper is to find new triangular finite-element pairs which are able to improve upon the discretization of the shallow-water equations; such pairs need to have no pressure modes and preferably satisfy CR $\simeq 1$. All the aforementioned problems occur in the context of linear formulations; hence solving *inviscid linear* equations is sufficient for our purpose. The shallow-water equations are of considerable importance for a variety of problems of coastal and environmental engineering, including oceanic, atmospheric, and groundwater flows. For many of these flows the boundary conditions require the calculation of the normal at the boundary. The latter is generally not unique at boundary vertices, and hence velocity nodes should be located elsewhere in order to exactly satisfy the boundary conditions.

Element pairs satisfying CR $\simeq 1$ and avoiding the placement of velocity nodes at vertices are found in section 2. The ability of these pairs to generate pressure modes and to solve two basic equations embedded in the shallow-water equations is examined in sections 3 and 4, respectively. This leads to the choice of a new pair. In section 5 the linear inviscid shallow-water equations are discretized using this pair. The results of environmental flow experiments, namely, propagation of fast gravity modes and slow Rossby modes, are presented and discussed in section 6. Conclusions are summarized in section 7.

2. Finite elements candidates.

In the two-dimensional case, for one connected component, let V be the total number of vertices of a given domain, C the number of cells or triangles, TF the total number of faces of the triangulation, and IF the number of interior faces. Euler's relations may then be expressed as

$$TF + IF = 3\,C\,, \tag{2.1}$$

$$V + IF = 2\,C + 1\,. \tag{2.2}$$

Assuming $TF \simeq IF$ we deduce from (2.1) and (2.2) that the number of triangles and midpoint nodes is approximately $2\,V$ and $3\,V$, respectively. This estimate provides an easy tool for finding velocity and pressure node locations leading to CR $\simeq 1$. An obvious choice—and the subject of this section—is to locate one variable at midpoints and the other at both vertices and barycenters. A second possibility offers three possible locations for the variables: at vertices, barycenters, and midpoints (P_2^+ [2]) or twice at midpoints (P_2^{NC} [3]) or at three internal nodes (P_{0-3} [9]). However, because the P_2^{NC} element is not directly usable [5] and the P_2^+ has nodes at the vertices, this second possibility is not investigated here. Further, it seems difficult to obtain other tractable combinations.

In the following let the subscripts $_A$, $_B$, and $_C$ distinguish new elements from existing ones, and let $N^{\mathbf{u}}$ and N^p be, respectively, the number of velocity and pressure nodes of the domain. In all figures the symbol • indicates the location of nodes.

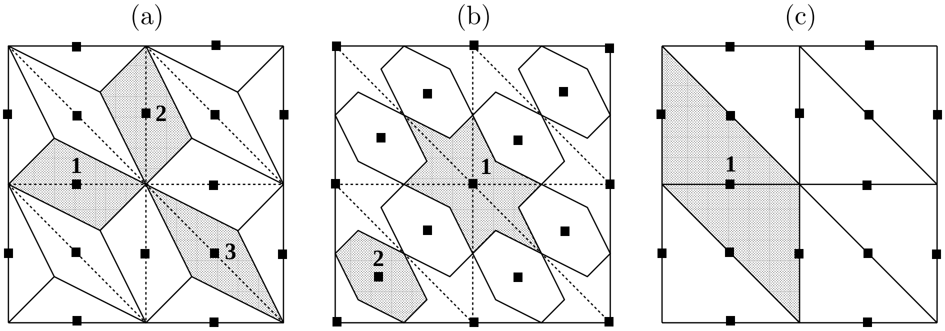


FIG. 2.1. *The elements* (a) P_{0-A}*,* (b) P_{0-B}*,* (c) P_1^{NC}*. The compact support of the basis function at selected nodes is shaded.*

2.1. Constant and linear elements. Examples of P_{0-A} and P_{0-B} elements are shown in Figures 2.1(a) and (b), respectively, on a square domain made up of eight biased dotted triangles (except for the boundaries). The element P_{0-A} is obtained in Figure 2.1(a) by joining the barycenters of biased triangles (dotted lines) with their vertices. The nodes are defined at midpoints and the basis functions are constant over the resulting quadrangles (solid lines), or triangles for boundary nodes. The element P_{0-B} shown in Figure 2.1(b) has nodes at both vertices and barycenters, and the basis functions are constant over the hexagonal and "star" areas defined by solid lines. The P_1^{NC} element [3, 7], shown in Figure 2.1(c), has nodes at triangle midpoints. The nonconforming linear basis functions are continuous only across triangle boundaries at midpoint nodes and are discontinuous everywhere else around a triangle boundary. Four pairs are considered: the $P_{0-A} - P_{0-B}$, the $P_{0-A} - P_1^{NC}$, the $P_1^{NC} - P_{0-A}$, and the $P_1^{NC} - P_{0-B}$ pairs. The first of these pairs is treated as a finite volume.

2.2. Linear–linear element pairs.

2.2.1. The P_1^{NC}–cross-grid P_1 element pair. The cross-grid P_1 element shown in Figure 2.2(a) has nodes at the triangle vertices and barycenters. The basis functions are linear upon each one of the three subtriangles sharing the center of gravity of the element as a common vertex and vanishing on the element boundary. We have $N^p = V + C$ and $N^{\mathbf{u}} = TF$, and thus $N^p - N^{\mathbf{u}} = V + C - TF$. From (2.1) and (2.2) we have $TF - V = C - 1$, and thus $N^p - N^{\mathbf{u}} = 1$ and CR $\simeq 1$.

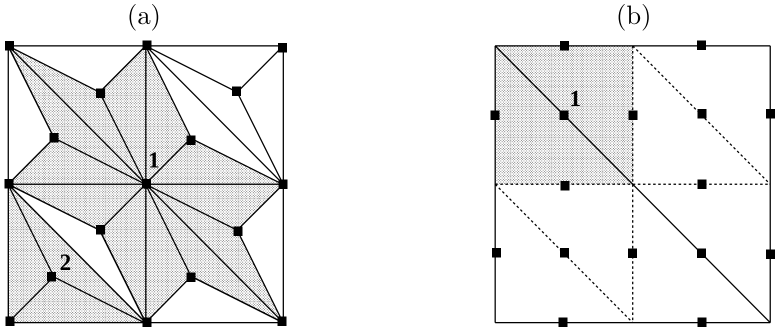


FIG. 2.2. *The elements* (a) *cross-grid* P_1*,* (b) P_1^{NC} *iso* P_2*. The compact support of the basis function at selected nodes is shaded.*

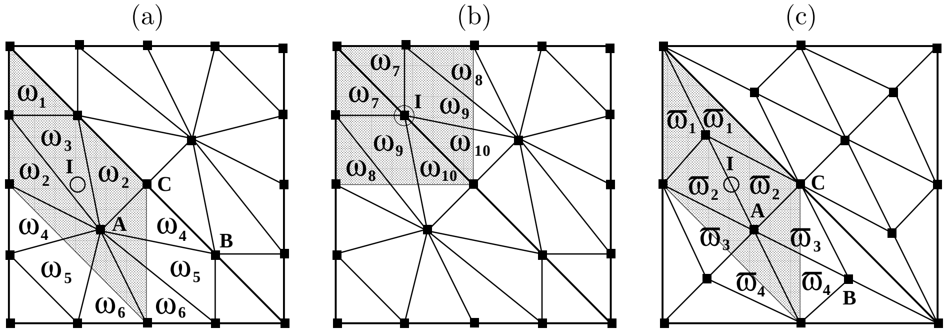


FIG. 2.3. (a) *and* (b)*: The* P_{1-A} *element.* (c) *The* P_{1-B} *element. At node I,* (a) *and* (c) *on a horizontal face,* (b) *on a diagonal face; the compact support of a* P_1^{NC} *iso* P_2 *basis function is shaded.*

TABLE 2.1
Definition of the weights used in the calculation of the gradient matrix of a scalar field for the P_1^{NC} *iso* $P_2 - P_{1-A}$ *and* P_1^{NC} *iso* $P_2 - P_{1-B}$ *pairs.*

P_1^{NC} iso $P_2 - P_{1-A}$					P_1^{NC} iso $P_2 - P_{1-B}$	
$\omega_1 = -\dfrac{1}{6}$	$\omega_3 = \dfrac{109}{450}$	$\omega_5 = \dfrac{2}{225}$	$\omega_7 = \dfrac{1}{3}$	$\omega_9 = \dfrac{3}{25}$	$\varpi_1 = \dfrac{1}{18}$	$\varpi_3 = \dfrac{1}{9}$
$\omega_2 = \dfrac{16}{45}$	$\omega_4 = \dfrac{23}{225}$	$\omega_6 = -\dfrac{13}{225}$	$\omega_8 = -\dfrac{3}{25}$	$\omega_{10} = \dfrac{11}{50}$	$\varpi_2 = \dfrac{7}{18}$	$\varpi_4 = -\dfrac{1}{18}$

2.2.2. The P_1^{NC} iso $P_2 - P_{1-A}$ and P_1^{NC} iso $P_2 - P_{1-B}$ element pairs. The P_1^{NC} iso P_2 element is obtained by dividing each biased triangle of Figure 2.2(b) (solid lines) into four subtriangles (dotted lines) using the midpoints of the triangle sides. On each subtriangle, the nodes are defined at triangle midpoints and nonconforming linear basis functions (P_1^{NC}) are used over the refined triangulation. The P_1^{NC} iso P_2 element is chosen to approximate the velocity variable, and so there are 9 velocity nodes per unrefined triangle or $12\,V$ over the domain. To obtain CR $= 1$ we need to find $12\,V$ pressure nodes. If we suppose that the pressure variables are at least located at the V vertices, we search for two positive integers n_1 and n_2, respectively, the number of nodes per face and per cell of the unrefined triangulation, such that

$$(3\,V)\,n_1 + (2\,V)\,n_2 = 12\,V - V \qquad \text{or} \qquad 3\,n_1 + 2\,n_2 = 11\,. \tag{2.3}$$

Equation (2.3) has only two solutions, $(n_1, n_2) = (3, 1)$ and $(n_1, n_2) = (1, 4)$, which define the nodal positions for the P_{1-A} and P_{1-B} pairs, as shown in Figure 2.3 for two biased triangles. In both cases 12 subtriangles are defined per unrefined triangle and conforming linear basis functions are used over the refined triangulation. We have $N_A^p = V + 3\,TF + C$, $N_B^p = V + TF + 4\,C$, and $N_{A,B}^{\mathbf{u}}(u) = 2\,TF + 3\,C$. Let BF to be the number of boundary faces, with $TF = IF + BF$. From (2.1)–(2.2) we deduce $N_A^p - N_{A,B}^{\mathbf{u}} = BF + 1$ and $N_B^p - N_{A,B}^{\mathbf{u}} = 1$; thus CR $\simeq 1$ for both elements.

The weights ω_i $(i = 1, 10)$ and ϖ_i $(i = 1, 4)$ given in Table 2.1 are used to calculate the discrete gradient matrix of a scalar field (e.g., pressure) for any original triangulation, even an unstructured one. For example, in Figure 2.3(a) let I (denoted by the symbol ○) be a velocity node and (x_B, y_B) and (x_C, y_C) denote the coordinates of the nodes B and C. The entries of the gradient matrix at line I and column A (for the contribution of triangle (A, B, C) only) are $\omega_4\,(y_B - y_C)$ and $\omega_4\,(x_C - x_B)$ for the

x- and y-derivatives, respectively. The other entries are obtained in the same manner by cyclic permutations. For the element P_{1-A} only, two different sets of weights are defined depending on whether I is an interior or a boundary element node.

2.2.3. The P_1^{NC}–P_{1-C} element pair. The cross-grid P_1 element is shown in Figure 2.4(a) over a biased triangulation intentionally represented with dotted faces, except for the bottom and right boundaries. The P_{1-C} element is derived from the cross-grid P_1 in the following manner. Each dotted face (Figure 2.4(a)) is suppressed and replaced by a new face (Figure 2.4(b)) joining the barycenters of the two adjacent triangles (except for the boundary faces). For example, faces 1-2, 2-3, and 3-1 are suppressed in Figure 2.4(a) and faces 4-5, 4-6, and 4-7 are created in Figure 2.4(b). Except for the boundary triangles, such a transformation preserves equilateral triangles. As for the P_1^{NC}–cross-grid P_1 pair, $N^p - N^{\mathbf{u}} = 1$; thus CR $\simeq 1$.

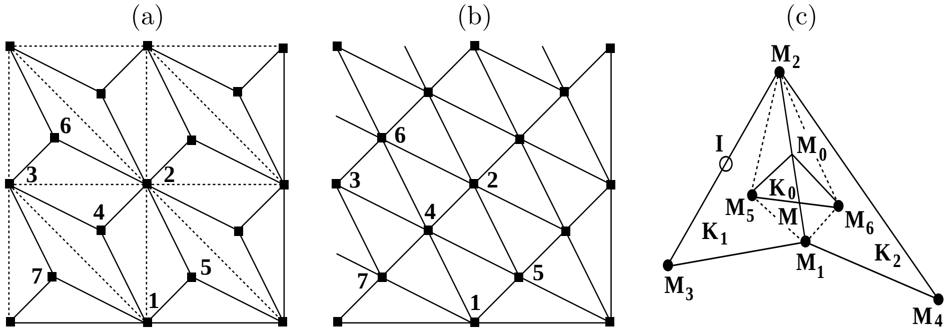


FIG. 2.4. *The elements* (a) *cross-grid P_1,* (b) *P_{1-C},* (c) *reconnection of faces for nonregular geometries.*

The determination of the weights used to calculate the discrete gradient matrix of a scalar field in the case of the P_1^{NC}–P_{1-C} pair now follows. The computation is more difficult than in section 2.2.2 because the P_{1-C} element results from a redefinition of the triangulation by swapping the faces. Except for regular geometries (e.g., biased triangles of Figure 2.4(b)), it is no longer guaranteed that the midpoints of old and new faces coincide. The problem is illustrated in Figure 2.4(c). Let M_0 and M be the middle points of (M_1, M_2) and (M_5, M_6), respectively, where M_5 and M_6 are the respective barycenters of the triangles K_1 (M_1, M_2, M_3) and K_2 (M_1, M_4, M_2). Let (x_i, y_i) and (x, y) be the coordinates of M_i $(i = 0, 6)$ and M, respectively, and let K_0 be the triangle (M_0, M_5, M_6) with

$$\text{Area}(K_0) = \frac{\varepsilon}{2}\,[\,(x_5 - x_0)\,(y_6 - y_0) - (y_5 - y_0)\,(x_6 - x_0)\,]\,, \tag{2.4}$$

where $\varepsilon = +1$ if M_0, M_5, M_6 are numbered counterclockwise (as in Figure 2.4 (c)) and $\varepsilon = -1$ otherwise.

PROPOSITION 2.1. *We have*

$$\overrightarrow{MM_0} = \frac{3\,\varepsilon\,Area(K_0)}{Area(K_1) + Area(K_2)}\;\overrightarrow{M_1M_2}\;. \tag{2.5}$$

Proof. Since $\overrightarrow{MM_0}$ and $\overrightarrow{M_1M_2}$ are parallel,

$$(x_0 - x)\,(y_2 - y_1) + (y_0 - y)\,(x_1 - x_2) = 0\,. \tag{2.6}$$

Then, noting that M belongs to (M_5, M_6), we obtain

$$x\,(y_6 - y_5) + y\,(x_5 - x_6) = x_5\,y_6 - x_6\,y_5\,. \tag{2.7}$$

By using the property $\overrightarrow{M_3M_4} = 3\ \overrightarrow{M_5M_6}$ and (2.4), equation (2.7) reduces to

$$(x_0 - x)\,(y_3 - y_4) + (y_0 - y)\,(x_4 - x_3) = 6\,\varepsilon\,\text{Area}(K_0)\,. \tag{2.8}$$

Some elementary calculations show that

$$\begin{vmatrix} y_2 - y_1 & x_1 - x_2 \\ y_3 - y_4 & x_4 - x_3 \end{vmatrix} = 2\,[\,\text{Area}(K_1) + \text{Area}(K_2)\,]\,, \tag{2.9}$$

and the solution of (2.6) and (2.8) then gives

$$x_0 - x = \frac{3\,\varepsilon\,\text{Area}(K_0)}{\text{Area}(K_1) + \text{Area}(K_2)}\,(x_2 - x_1)\,, \tag{2.10}$$

$$y_0 - y = \frac{3\,\varepsilon\,\text{Area}(K_0)}{\text{Area}(K_1) + \text{Area}(K_2)}\,(y_2 - y_1)\,. \tag{2.11}$$

Thus, we obtain (2.5), and the proof is completed. □

Let I be a velocity node located at the middle point of (M_2, M_3) as shown in Figure 2.4(c). Typical weights associated with I over the triangles (M_1, M_6, M_5), (M_2, M_5, M_6), and (M_2, M_3, M_5) of the P_{1-C} triangulation are

$$\omega_{(M_1,M_6,M_5)} = \left(-\frac{1}{9} + \frac{\varepsilon}{3}\,\frac{\|\overrightarrow{MM_0}\|}{\|\overrightarrow{M_1M_2}\|}\right)\,\frac{\text{Area}(K_1)}{\text{Area}(K_1) + \text{Area}(K_2)}\,, \tag{2.12}$$

$$\omega_{(M_2,M_5,M_6)} = \left(\frac{2}{9} + \frac{\varepsilon}{3}\,\frac{\|\overrightarrow{MM_0}\|}{\|\overrightarrow{M_1M_2}\|}\right)\,\frac{\text{Area}(K_1)}{\text{Area}(K_1) + \text{Area}(K_2)}\,, \tag{2.13}$$

$$\omega_{(M_2,M_3,M_5)} = \frac{7}{18} \quad \text{whether } (M_2, M_3) \text{ is a boundary face or not}\,, \tag{2.14}$$

where $\|\overrightarrow{MM_0}\|$ is obtained from (2.5). If (M_1, M_2) is a boundary face, we have $\omega_{(M_1,M_2,M_5)} = 1/18$. All the weights are derived by analogy with (2.12)–(2.14) and they are used in the same manner as in section 2.2.2 to calculate the entries of the gradient matrix of a scalar field at line I.

3. Examination of the pressure modes. To determine if the element pairs considered in section 2 have pressure modes, their corresponding discrete gradient matrices are computed on Grid 1, as shown in Figure 3.1(a), and then decomposed in singular values (SV). For all pairs at least 3 SV are zero, implying that the kernel of the discrete gradient operator is more than one-dimensional and hence that pressure modes exist. On Grid 1 two sides of a triangle may coincide with the boundary. By requiring triangulation into corners, as shown on Grid 2 in Figure 3.1(b) at the upper right and lower left corners, better results are obtained as shown in Table 3.1 for different types of boundary conditions.

For the $P_{0-A} - P_1^{NC}$ and the $P_1^{NC} - P_{1-C}$ pairs only 1 SV is zero; the null space of their discrete gradient operator is one-dimensional and these pairs have no pressure modes. This result reflects the fact that pressure is only determined to within an

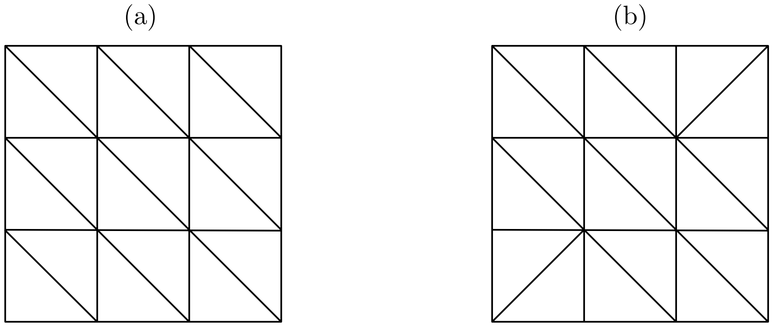


FIG. 3.1. (a) *Grid* 1*, a* 4×4 *mesh;* (b) *Grid* 2*, as for* (a) *with triangulation into corners.*

TABLE 3.1
Number of zero SV of the discrete gradient matrix computed on Grid 2 *for several element pairs.*

Element pair	Dirichlet boundary conditions	Normal velocity specified	Tangential velocity specified
P_{0-A}–P_{0-B}	2	2	1
P_{0-A}–P_1^{NC}	1	1	1
P_1^{NC}–cross-grid P_1	2	2	1
P_1^{NC} iso P_2–P_{1-A}	2	2	1
P_1^{NC} iso P_2–P_{1-B}	2	2	1
P_1^{NC}–P_{1-C}	1	1	1

arbitrary additive constant by fixing the pressure reference level. The requirement to triangulate into corners is a minor constraint, even sometimes desirable to impose boundary conditions. The other pairs in Table 3.1 have 2 zero SV and thus have pressure modes. Finally, the $P_1^{NC} - P_{0-A}$ and the $P_1^{NC} - P_{0-B}$ pairs (not included in Table 3.1) have many pressure modes whatever the grid.

The results shown in Table 3.1 have been found to be identical on meshes $n \times n$ with $n \geq 5$ and do not depend on whether the number of nodes in each direction is even or odd. It has been also verified that when only 1 SV is zero no other SV converges to zero as the mesh parameter becomes small.

At this stage of the argument, the P_{0-A}–P_1^{NC} and P_1^{NC}–P_{1-C} pairs are favorable candidates for solving the coupled momentum-continuity equations.

4. Representing basic equations. Two basic equations embedded in the shallow-water formulation are now considered:

$$\mathbf{u} = \alpha \nabla p \,, \tag{4.1}$$

$$p = \gamma \nabla \cdot \mathbf{u} \,, \tag{4.2}$$

where $\mathbf{u} = (u, v)$ and p are, respectively, the velocity field and the pressure, and α and γ are constant scaling factors.

In [9] it is shown that pairs having a low CR perform poorly in solving (4.1), while smooth results are obtained for pairs with a CR larger than 1. A similar situation is expected to arise, in the opposite sense, when computing p from $\mathbf{u}$ in (4.2); this has been verified for several pairs (results not shown). As a preliminary before solving

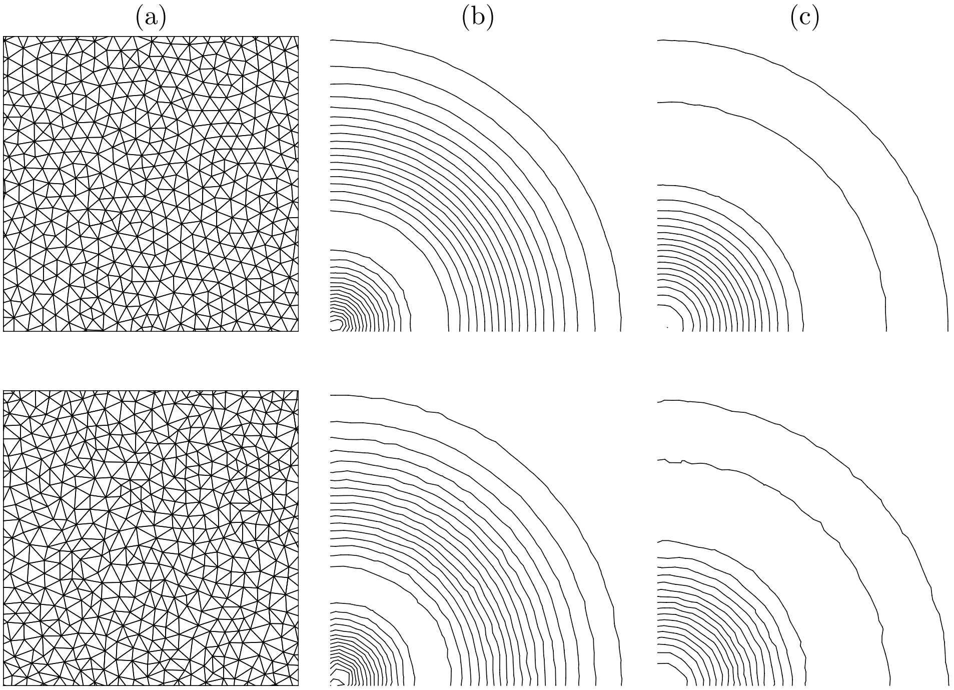


FIG. 4.1. (a) *A window of Mesh* 1 *made up of unstructured triangles with smoothing (row* 1*), a window of Mesh* 2 *made up of unstructured triangles without smoothing (row* 2*);* (b) *isolines (upper right quarter) of the simulated flow speed field corresponding to* (4.1) *on Mesh* 1 *(row* 1*) and Mesh* 2 *(row* 2*) for the* P_1^{NC}*–*P_{1-C} *pair;* (c) *isolines (upper right quarter) of the simulated pressure corresponding to* (4.2) *on Mesh* 1 *(row* 1*) and Mesh* 2 *(row* 2*) for the* P_1^{NC}*–*P_{1-C} *pair.*

the shallow-water equations, smooth results should be obtained for $\mathbf{u}$ and p in (4.1) and (4.2). These two basic equations are now solved for the pairs in Table 3.1.

The pressure in (4.1) is a specified Gaussian distribution with an e-folding radius that is resolved by about 15 velocity nodes on Meshes 1 and 2, shown in Figure 4.1(a), and $\mathbf{u}$ is to be determined from the finite-element discretization of (4.1). The velocity field in (4.2) is the exact gradient of the specified Gaussian distribution used for p in (4.1) and p is to be determined from the finite-element discretization of (4.2). The exact solutions for the flow-speed field $(u^2+v^2)^{1/2}$ and p are shown in Figure 4.2. The numerical solutions shown here are all obtained using the visualization environment VU [10].

For the P_1^{NC}–P_{1-C} pair smooth solutions, shown in Figures 4.1(b) and (c), are obtained on Mesh 1. Good results are also found on Mesh 2 considering the unusually high level of mesh distortion. The other pairs of Table 3.1 also give smooth results for both solutions, due to the fact that CR $\simeq 1$. There is, however, an exception for the P_{0-A}–P_1^{NC} pair, which gives a very noisy representation of p in (4.2). The problem arises from a lack of contribution of u and v to the discretization of the divergence term at horizontal and vertical nodes, respectively. This is consistent with the lack of equations for u and v on horizontal and vertical faces, respectively, in solving (4.1) with the P_1^{NC}–P_{0-A} pair (and also the P_1^{NC}–P_0 pair [9, Figure 3(a)]).

By a process of elimination, the P_1^{NC}–P_{1-C} pair has been identified as a promising choice. The discretization of the linear inviscid shallow-water equations using this pair now follows.

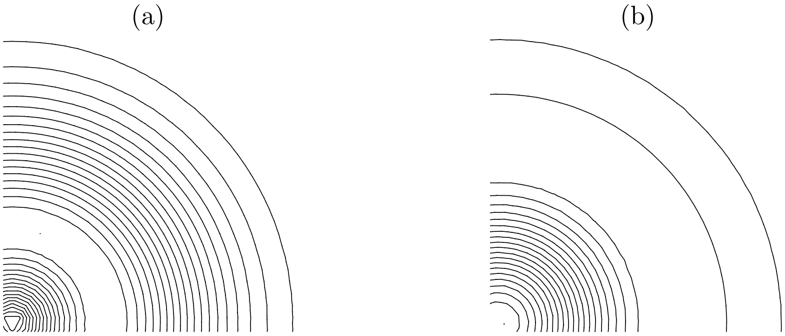


FIG. 4.2. (a) *Isolines (upper right quarter) of the exact solution for the flow-speed field corresponding to* (4.1)*, where the pressure is a specified Gaussian;* (b) *isolines (upper right quarter) of the exact solution for pressure corresponding to* (4.2)*, where the velocity field is the exact gradient of a specified Gaussian.*

5. Discretization of the inviscid linear shallow-water equations. Let Ω be the model domain with boundary Γ. The inviscid linear shallow-water equations are expressed in Cartesian coordinates [8] as

$$\mathbf{u}_t + f\,\mathbf{k}\times\mathbf{u} + g\,\nabla\eta = 0\,, \tag{5.1}$$

$$\eta_t + H\,\nabla\cdot\mathbf{u} = 0\,, \tag{5.2}$$

where η is the surface elevation with respect to the reference level $z = 0$, f and g are the Coriolis parameter and the gravitational acceleration, respectively, $\mathbf{k}$ is a unit vector in the vertical, and the mean depth H is constant. Note that η plays the role that pressure plays in the Navier–Stokes equations. For a contained flow, (5.1) and (5.2) are solved subject to the no-normal flow boundary condition

$$\mathbf{u}\cdot\mathbf{n} = 0 \quad \text{on } \Gamma\,, \tag{5.3}$$

where $\mathbf{n}$ is the outward pointing normal at the boundary.

An implicit Crank–Nicolson time discretization of (5.1) and (5.2) gives

$$\mathbf{u} + \frac{f\,\Delta t}{2}\,\mathbf{k}\times\mathbf{u} + \frac{g\,\Delta t}{2}\,\nabla\eta = \left[\mathbf{u} - \frac{f\,\Delta t}{2}\,\mathbf{k}\times\mathbf{u} - \frac{g\,\Delta t}{2}\,\nabla\eta\right]_{t-\Delta t} \equiv \mathbf{R}^{\mathbf{u}}\,, \tag{5.4}$$

$$\eta + \frac{H\,\Delta t}{2}\,\nabla\cdot\mathbf{u} = \left[\eta - \frac{H\,\Delta t}{2}\,\nabla\cdot\mathbf{u}\right]_{t-\Delta t} \equiv R^{\eta}\,, \tag{5.5}$$

where $[\,.\,]_{t-\Delta t}$ denotes evaluation at the previous timestep.

The Sobolev space $H^1(\Omega)$ is the space of functions in the square-integrable space $L^2(\Omega)$, whose first derivatives belong to $L^2(\Omega)$. Let η be in a subspace V of $H^1(\Omega)$ and let each component of $\mathbf{u}$ be a sufficiently regular scalar function such that $\mathbf{u}\cdot\mathbf{n} = 0$ on Γ. The weak formulation of (5.4) and (5.5) requires the test functions $\boldsymbol{\varphi}$ (whose x- or y-component is formally denoted by φ) and ψ to belong, respectively, to the same function space as $\mathbf{u}$ and η, such that

$$\int_\Omega \mathbf{u}\cdot\boldsymbol{\varphi}\,d\Omega + \frac{\Delta t}{2}\int_\Omega f\,(\mathbf{k}\times\mathbf{u})\cdot\boldsymbol{\varphi}\,d\Omega + \frac{g\,\Delta t}{2}\int_\Omega \nabla\eta\cdot\boldsymbol{\varphi}\,d\Omega = \int_\Omega \mathbf{R}^{\mathbf{u}}\cdot\boldsymbol{\varphi}\,d\Omega\,, \tag{5.6}$$

$$\int_\Omega \eta\,\psi\,d\Omega + \frac{H\,\Delta t}{2}\int_\Omega \nabla\cdot\mathbf{u}\,\psi\,d\Omega = \int_\Omega R^{\eta}\,\psi\,d\Omega\,, \tag{5.7}$$

where $d\Omega$ is the areal element.

On both sides of (5.7) the term $\nabla\cdot\mathbf{u}$ is integrated by parts using Green's theorem. The boundary integrals vanish by applying (5.3); thus (5.7) can be rewritten as

$$\int_\Omega \eta\,\psi\,d\Omega - \frac{H\,\Delta t}{2}\int_\Omega \mathbf{u}\cdot\nabla\,\psi\,d\Omega = \int_\Omega R^\eta\,\psi\,d\Omega\,, \tag{5.8}$$

where

$$\int_\Omega R^\eta\,\psi\,d\Omega = \int_\Omega [\,\eta\,]_{t-\Delta\,t}\,\psi\,d\Omega + \frac{H\,\Delta t}{2}\int_\Omega [\,\mathbf{u}\,]_{t-\Delta\,t}\cdot\nabla\,\psi\,d\Omega\,. \tag{5.9}$$

The Galerkin finite-element method then approximates the solution of (5.6) and (5.8) in finite-dimensional subspaces. Consider two triangulations $\mathcal{T}_h^{\mathbf{u}}$ and $\mathcal{T}_h^{\eta}$ of the polygonal domain Ω, defined in Figures 2.1(c) and 2.4(b), respectively, where h is a discretization parameter tending to zero. For triangles $K^{\mathbf{u}}\in\mathcal{T}_h^{\mathbf{u}}$ and $K^\eta\in\mathcal{T}_h^\eta$, let $P_1(K^{\mathbf{u}})$ and $P_1(K^\eta)$ denote the space of linear polynomials on $K^{\mathbf{u}}$ and K^η, respectively.

The discrete solution $\mathbf{u}_h$ sought belongs to a finite-dimensional space $\mathbf{W}_h$ defined to be the set of functions $\mathbf{u}_h$ whose restriction on $K^{\mathbf{u}}$ belongs to $P_1(K^{\mathbf{u}})\times P_1(K^{\mathbf{u}})$, with $\mathbf{u}_h$ being continuous only at the midpoints of each face of $\mathcal{T}_h^{\mathbf{u}}$, and $\mathbf{u}_h\cdot\mathbf{n}=0$ on Γ. The discrete solution η_h is sought in a finite-dimensional subspace V_h of V, where V_h is defined to be the set of functions η_h whose restriction on K^η belongs to $P_1(K^\eta)$, with η_h being continuous at each vertex of $\mathcal{T}_h^\eta$.

By applying the Galerkin procedure the problem can be summarized as seeking solutions $\mathbf{u}_h$ and η_h, respectively, in $\mathbf{W}_h$ and V_h such that

$$\begin{aligned}\sum_{K^{\mathbf{u}}\in\mathcal{T}_h^{\mathbf{u}}}\int_{K^{\mathbf{u}}}\mathbf{u}_h\cdot\boldsymbol{\varphi}_i\,d\Omega &+ \frac{\Delta\,t}{2}\sum_{K^{\mathbf{u}}\in\mathcal{T}_h^{\mathbf{u}}}\int_{K^{\mathbf{u}}} f\,(\mathbf{k}\times\mathbf{u}_h)\cdot\boldsymbol{\varphi}_i\,d\Omega\\ &+\frac{g\,\Delta\,t}{2}\sum_{K^{\mathbf{u}}\in\mathcal{T}_h^{\mathbf{u}}}\int_{K^{\mathbf{u}}}\nabla\eta_h\cdot\boldsymbol{\varphi}_i\,d\Omega = \sum_{K^{\mathbf{u}}\in\mathcal{T}_h^{\mathbf{u}}}\int_{K^{\mathbf{u}}}\mathbf{R}^{\mathbf{u}}\cdot\boldsymbol{\varphi}_i\,d\Omega\,,\end{aligned} \tag{5.10}$$

$$\sum_{K^\eta\in\mathcal{T}_h^\eta}\int_{K^\eta}\eta_h\,\psi_j\,d\Omega - \frac{H\,\Delta\,t}{2}\sum_{K^\eta\in\mathcal{T}_h^\eta}\int_{K^\eta}\mathbf{u}_h\cdot\nabla\psi_j\,d\Omega = \sum_{K^\eta\in\mathcal{T}_h^\eta}\int_{K^\eta}R^\eta\,\psi_j\,d\Omega \tag{5.11}$$

for all basis functions $\boldsymbol{\varphi}_i$ and ψ_j (defined in [3]) belonging to $\mathbf{W}_h$ and V_h, respectively, where i and j are typical nodes of $\mathcal{T}_h^{\mathbf{u}}$ and $\mathcal{T}_h^\eta$, respectively.

After insertion of the expansions $\mathbf{u}_h=\sum_{k=1}^3\mathbf{u}_k\,\varphi_k$ over $K^{\mathbf{u}}$ and $\eta_h=\sum_{l=1}^3\eta_l\,\psi_l$ over K^η into (5.10) and (5.11), where $\mathbf{u}_k$ and η_l are the values of $\mathbf{u}_h$ and η_h at nodes k and l of $\mathcal{T}_h^{\mathbf{u}}$ and $\mathcal{T}_h^\eta$, respectively, we obtain a set of linear equations of the form

$$\mathsf{M}^{\mathbf{u}}\,\mathbf{u} + \frac{g\,\Delta\,t}{2}\,\mathsf{G}\,\eta = \mathsf{R}^{\mathbf{u}}\,, \tag{5.12}$$

$$\mathsf{M}^\eta\,\eta - \frac{H\,\Delta\,t}{2}\,\mathsf{G}^t\,\mathbf{u} = \mathsf{R}^\eta\,, \tag{5.13}$$

where $\mathsf{M}^{\mathbf{u}}$ and M^η denote the velocity and surface-elevation mass matrices, respectively, G is the gradient matrix, and $\mathsf{R}^{\mathbf{u}}$ and R^η are the right-hand sides.

Since the velocity nodes are located at triangle midpoints, the normal direction along the boundary is defined uniquely, a property which is not true in general for boundary vertices. Following [4], the x-y momentum equations corresponding to a boundary node in (5.12) are transformed into tangential and normal equations, the local x-y coordinate system at this node is rotated to coincide with the tangential and normal directions, and the boundary condition (5.3) is then applied.

Due to the orthogonality property of the velocity basis functions [13], $\mathsf{M}^{\mathbf{u}}$ is a diagonal matrix with 2×2 blocks along the diagonal. Equation (5.12) is rewritten as

$$\mathbf{u} = -\frac{g\,\Delta t}{2}\,(\mathsf{M}^{\mathbf{u}})^{-1}\,\mathsf{G}\,\eta + (\mathsf{M}^{\mathbf{u}})^{-1}\,\mathsf{R}^{\mathbf{u}}\,, \tag{5.14}$$

and (5.14) is then used to eliminate $\mathbf{u}$ from (5.13). The substitution results in

$$\left(\mathsf{M}^{\eta} + g\,H\,\frac{(\Delta\,t)^2}{4}\,\mathsf{G}^t\,(\mathsf{M}^{\mathbf{u}})^{-1}\,\mathsf{G}\right)\,\eta = \mathsf{R}^{\eta} + H\,\frac{\Delta\,t}{2}\,\mathsf{G}^t\,(\mathsf{M}^{\mathbf{u}})^{-1}\,\mathsf{R}^{\mathbf{u}}\,. \tag{5.15}$$

The elimination of $\mathbf{u}$ thus leads to a linear system for the η only, which greatly enhances computational efficiency. The matrix on the left-hand side of (5.15) is a very sparse matrix, with an average of 31 nonzero elements per row. Finally, once η is obtained from (5.15), $\mathbf{u}$ is computed explicitly from (5.14).

6. Numerical results. A linear stability analysis of (5.1) and (5.2) reveals that there are two basic kinds of associated motion: small-amplitude fast-moving gravitational oscillations and slow-moving Rossby modes [6]. To determine how the P_1^{NC}–P_{1-C} pair approximates these two types of modes, two tests are proposed. The first test examines the propagation and dispersion of fast surface gravity waves in a circular basin and their reflection at the lateral boundary. In the second test, the slowly propagating Rossby modes are simulated in the case of the evolution of a typical anticyclonic vortex at midlatitudes.

For both tests, the linear inviscid shallow-water equations are solved with a Gaussian distribution of the surface elevation prescribed at initial time, i.e.,

$$\eta\,(r,0) = a\,e^{-b\,r^2}\,, \tag{6.1}$$

where r is the distance from the Gaussian's center, and a and b are prescribed.

The second, more stringent test was applied in [9] and the $P_1\,\mathrm{iso}\,P_2$–P_{0-3} pair (CR = 3/2) was shown to give much better results than already existing pairs. Hence, this pair is chosen for comparison with the P_1^{NC}–P_{1-C} pair.

6.1. Gravity wave propagation and dispersion. For many applications very little energy is carried by the small-amplitude fast-moving surface gravity waves, and this justifies slowing them down via a semi-implicit time discretization. Nevertheless, the numerical solution of (5.1)–(5.3) at small values of the gravitational Courant number $C_g \equiv c\,\Delta\,t\,/\,h_0$, where $c \equiv \sqrt{g\,H}$ is the phase speed of the surface gravity waves and h_0 is the smallest distance between two nodes of $\mathcal{T}_h^{\mathbf{u}}$ or $\mathcal{T}_h^{\eta}$, should be expected to reasonably well approximate the analytical one when using the semi-implicit scheme.

The analytical solution of the problem is obtained from (5.1) and (5.2) by exploiting the circular symmetry. In order to do so, the Coriolis term is set to zero, leading to a second-order wave equation in polar coordinates for η, viz.

$$\eta_{tt} - c^2\,\frac{1}{r}\,(\,r\,\eta_r\,)_r = 0 \tag{6.2}$$

subject to the boundary conditions $\eta_r|_{r=R} = 0$, and no singularity at the origin. The initial conditions are $\eta\,(r,0) = a\,e^{-b\,r^2}$ and $\eta_t\,(r,0) = 0$; the latter condition follows from setting the initial velocity to zero. The exact solution of (6.2) is the Bessel-function expansion

$$\eta\,(r,t) = \sum_{n=1}^{\infty} \mu_n\, J_0\,(\,\sqrt{\lambda_n}\, r\,)\,\cos c\,\sqrt{\lambda_n}\, t\,,$$

where $\mu_n = \frac{\int_0^R \eta\,(r,0)\, J_0\,(\,\sqrt{\lambda_n}\, r\,)\, r\, dr}{\int_0^R J_0^2\,(\,\sqrt{\lambda_n}\, r\,)\, r\, dr}$ and λ_n are the roots of $J_1\,(\,\sqrt{\lambda_n}\, R\,) = 0$.

The circular domain has a radius of $R = 1000\,\mathrm{km}$ and is discretized using a 40 km vertex node spacing unstructured triangulation $\mathcal{T}_h^{\mathbf{u}}$ with smoothing ($h_0 = 20\,\mathrm{km}$). A flat bottom with mean depth $H = 2000\,\mathrm{m}$ is assumed, leading to $c \approx 140\,\mathrm{m\,s^{-1}}$. By taking a timestep of 20 s, C_g is approximately 0.15. The Gaussian distribution parameters that define η at initial time are set to $a = 100\,\mathrm{m}$ and $b = 6.4 \times 10^{-11}\,\mathrm{m}^{-2}$.

Time sequences for η are shown in Figure 6.1. The surface elevation is first shown at stage 1, after a single timestep. At stage 3 it is being reflected by the basin wall, and by stages 5, 6, 7, and 8 it has returned one, two, three, and four times, respectively, to its starting point. Comparing panels 1 and 5 of Figure 6.1 we can see that very little dispersion has occurred after a single cycle, but after two and three cycles the dispersion effect is quite obvious. The dispersion is due to the individual Bessel modes of the exact solution propagating with different phase speeds.

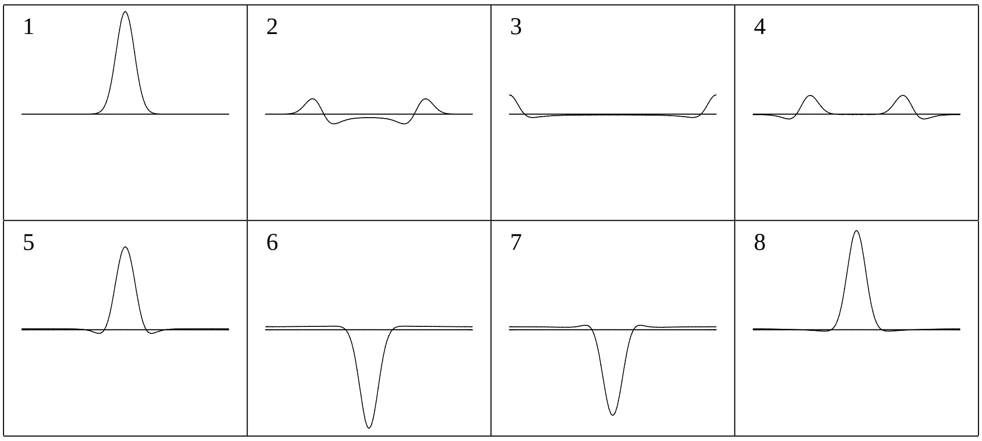


FIG. 6.1. *Vertical cross sections in the x, z plane of the surface elevation η at different stages of gravity wave propagation and dispersion. The initial Gaussian distribution is shown in panel* 1. *Reflection occurs at the boundary in panel* 3, *and in panels* 5, 6, 7, *and* 8 *the disturbance has returned to its starting point one, two, three, and four times, respectively.*

Good agreement is obtained between the analytical and the computed solutions for the P_1^{NC}–P_{1-C} pair as shown in Table 6.1. Further, it has been observed that the radial symmetry of the exact solution is very well reproduced by the numerical one. For the $P_1\,\mathrm{iso}\,P_2 - P_{0-3}$ pair the discrepancy with the analytical solution after stage 4 may arise from the constant approximation for η and/or from an inaccurate calculation of the normal at the boundary vertices. The test has been done on a Power Challenge XL machine with MIPS R8000 processor chips. Note that for the P_1^{NC}–P_{1-C} pair the computational cost and the memory requirement are reduced by a factor of $\delta_1 = 3.74$ and $\delta_2 = 1.84$, respectively, compared to the $P_1\,\mathrm{iso}\,P_2$–P_{0-3} pair. A conjugate gradient method is used to solve for (5.15) and residuals are found

TABLE 6.1

Maximum and minimum values of the analytical and computed surface-elevation field, for the P_1^{NC}*–*P_{1-C} *and* P_1 *iso* P_2*–*P_{0-3} *pairs, at the stages corresponding to the plots of Figure* 6.1.

Stage	No. of timesteps	Analytical	P_1^{NC}–P_{1-C}	P_1 iso P_2–P_{0-3}
1	1	99.55 0.00	99.55 0.00	99.77 0.00
2	179	14.84 -9.47	14.91 -9.53	15.05 -9.58
3	357	18.53 -3.33	18.64 -3.43	18.13 -3.68
4	536	18.12 -4.66	18.25 -4.79	18.58 -4.79
5	683	80.76 -3.58	80.40 -3.69	83.36 -3.57
6	1430	3.40 -95.57	3.52 -95.60	4.10 -90.86
7	2113	4.26 -83.92	4.46 -83.03	4.36 -86.10
8	2860	95.91 -1.33	96.13 -1.48	82.02 -3.03
	Number of velocity nodes		8176	10 958
	Number of pressure nodes		8177	16 185
	Computational cost		875 s	3271 s
	Memory requirement		2.5 MB	4.6 MB

to decrease by a factor of 10^{-6} within 10 iterations for both pairs using a diagonal preconditioner.

6.2. Eddy propagation. In the second experiment the domain is an idealized 1200 km × 1200 km square basin discretized using a 10 km vertex node spacing unstructured triangulation $\mathcal{T}_h^{\mathbf{u}}$ with smoothing ($h_0 = 5\,\mathrm{km}$). The constant depth $H = 1.63\,\mathrm{m}$ results in a phase speed for gravity waves of approximately $4\,\mathrm{m\,s^{-1}}$. Such a small equivalent depth is pertinent for the adjustment under gravity of a density-stratified fluid [6]. The initial Gaussian surface-elevation distribution is centered on a point 600 km from both the south and west walls. By setting $b = 5.92 \times 10^{-11}\,\mathrm{m^{-2}}$ the e-folding radius is 130 km. The β-plane approximation, $f = f_0 + \beta\, y$, is used, where f_0 and β are evaluated at 25^o N ($f_0 = 6.16 \times 10^{-5}\,\mathrm{s^{-1}}$ and $\beta = 2.07 \times 10^{-11}\,\mathrm{m^{-1}\,s^{-1}}$). The radius of deformation at midbasin is thus $R_d \equiv \sqrt{g\,H}/f_0 \approx 65\,\mathrm{km}$.

The initial symmetric anticyclonic velocity field is taken to be in exact geostrophic balance $f\,\mathbf{k} \times \mathbf{u} = -g\,\nabla\,\eta$, and thus

$$u\,(x, y, 0) = 2\,\frac{g}{f}\,a\,b\,y\,e^{-b\,(x^2 + y^2)}\,, \tag{6.3}$$

$$v\,(x, y, 0) = -2\,\frac{g}{f}\,a\,b\,x\,e^{-b\,(x^2 + y^2)}\,. \tag{6.4}$$

By setting $a = 0.95\,\mathrm{m}$, the initial maximum surface azimuthal velocity is $1\,\mathrm{m\,s^{-1}}$. The timestep is 30 minutes and thus C_g is approximately 1.5; the results are relatively insensitive to the precise choice of the timestep.

During the first inertial period ($2\,\pi/f_0 \simeq 28\,\mathrm{h}\,22\,\mathrm{mn}$) the initial condition adjusts to the β-plane balance of the model. After this initial adjustment, the anticyclonic vortex evolves purely westward at an average translation speed of $\beta\,{R_d}^2 \simeq 7.5\,\mathrm{km\,day^{-1}}$. The evolution of the flow-speed field and surface elevation is shown in Figure 6.2

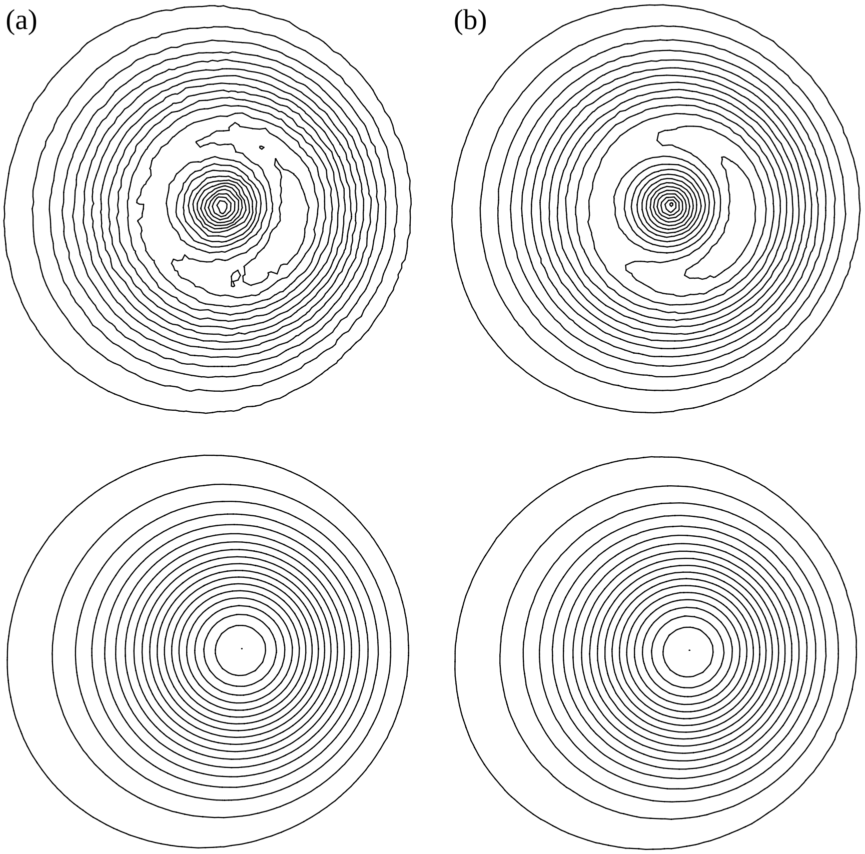


FIG. 6.2. *Isolines of the flow-speed field (row* 1*) and surface elevation (row* 2*) after one week of simulation for the* (a) P_1 *iso* P_2–P_{0-3}*,* (b) P_1^{NC}–P_{1-C} *pairs. The contour interval is* $0.05\,m\,s^{-1}$ *for the flow-speed field and* 0.05 *m for the surface elevation.*

for the P_1^{NC}–P_{1-C} and P_1 iso P_2–P_{0-3} element pairs after one week of simulation. Smooth surface elevations are obtained for both pairs. For the P_1 iso P_2–P_{0-3} pair, noise gradually develops in the flow-speed field during the week of simulation while results obtained with the P_1^{NC}–P_{1-C} pair are much smoother.

For this experiment δ_1 and δ_2 are similar to the first test. A GMRES iterative method is used to solve for (5.15) and residuals are found to decrease by a factor of 10^{-6} within 21 and 32 iterations for the P_1^{NC}–P_{1-C} and P_1 iso P_2–P_{0-3} pairs, respectively, using a diagonal preconditioner.

The P_1^{NC}–P_{1-C} element pair is shown to give favorable results not only for the propagation and dispersion of gravity waves, but also for the simulation of the slowly propagating Rossby modes.

7. Conclusions. Most popular finite-element pairs result in having more vector momentum than continuity discretized equations. Consequently, relevant problems such as wave propagation may be poorly approximated. In this paper several new pairs are proposed which lead to a good balance. Of these, one pair is shown to be preferable; it has no pressure modes (assuming a triangulation into corners) and gives smooth results for two basic equations embedded in the inviscid linear shallow-water equations. The latter set of equations is then solved using this new pair. Two tests are performed: simulation of propagation of gravity waves in a circular basin and simulation of evolution of an anticyclonic velocity field. In both experiments the new pair gives better results than those obtained previously, and at much lower computational and memory requirement costs. It is concluded that this pair is a very promising choice for the discretization of the nonlinear shallow-water equations with forcing and varying depth.

Acknowledgments. The author gratefully acknowledges Dr. Andrew Staniforth for his continued interest and support of this work, and both Dr. Andrew Staniforth and Professor Roger Pierre for reviewing a draft of the manuscript. Special thanks are due to Dr. Léonard Rivier and Dr. Isla Gilmour for critical proofreading and comments. Professor Charles Lin and CERCA (Centre de Recherche en Calcul Appliqué) are also acknowledged for providing computing facilities.

REFERENCES

[1] A. J. Adcroft, C. N. Hill, and J. C. Marshall, *A new treatment of the Coriolis terms in C-grid models at both high and low resolutions*, Monthly Weather Review, 127 (1999), pp. 1928–1936.

[2] F. Brezzi and M. Fortin, *Mixed and Hybrid Finite Element Methods*, Springer Ser. Comput. Math. 15, Springer-Verlag, Berlin, 1991.

[3] M. Crouzeix and P. A. Raviart, *Conforming and non-conforming finite-element methods for solving the stationary Stokes equations*, RAIRO Anal. Numér., 7 (1973), pp. 33–76.

[4] M. S. Engelman, R. L. Sani, and P. M. Gresho, *The implementation of normal and/or tangential boundary conditions in finite-element codes for incompressible fluid flow*, Internat. J. Numer. Methods Fluids, 2 (1982), pp. 225–238.

[5] M. Fortin and M. Soulié, *A non-conforming piecewise quadratic finite element on triangles*, Internat. J. Numer. Methods Engrg., 19 (1983), pp. 505–520.

[6] A. E. Gill, *Atmosphere-Ocean Dynamics*, Internat. Geophysics Series 31, Academic Press, San Diego, 1982.

[7] B. L. Hua and F. Thomasset, *A noise-free finite-element scheme for the two-layer shallow-water equations*, Tellus, 36A (1984), pp. 157–165.

[8] P. H. LeBlond and L. A. Mysak, *Waves in the Ocean*, Elsevier, Amsterdam, 1978.

[9] D. Y. Le Roux, A. Staniforth, and C. A. Lin, *Finite elements for shallow-water equation ocean models*, Monthly Weather Review, 126 (1998), pp. 1931–1951.

[10] B. Ozell, R. Camarero, A. Garon, and F. Guibault, *Analysis and visualization tools in CFD, part* I: *A configurable data extraction environment*, Finite Elem. Anal. Des., 19 (1995), pp. 295–307.

[11] R. L. Sani, P. M. Gresho, R. L. Lee, and D. Griffiths, *The cause and cure (?) of the spurious pressures generated by certain FEM solutions of the incompressible Navier–Stokes equations.* I, Internat. J. Numer. Methods Fluids, 1 (1981), pp. 17–43.

[12] R. L. Sani, P. M. Gresho, R. L. Lee, D. Griffiths, and M. Engelman, *The cause and cure (!) of the spurious pressures generated by certain FEM solutions of the incompressible Navier–Stokes equations.* II, Internat. J. Numer. Methods Fluids, 1 (1981), pp. 171–204.

[13] F. Thomasset, *Implementation of Finite Element Methods for Navier–Stokes Equations*, Springer-Verlag, Berlin, 1981.

[14] R. A. Walters and G. F. Carey, *Analysis of spurious oscillation modes for the shallow-water and Navier–Stokes equations*, Comput. Fluids, 11 (1983), pp. 51–68.

[15] R. A. Walters and G. F. Carey, *Numerical noise in ocean and estuarine models*, Advances in Water Resources, 7 (1984), pp. 5–20.

SIAM J. SCI. COMPUT.
Vol. 23, No. 1, pp. 81–91

EVALUATION OF SINGLE LAYER POTENTIALS OVER CURVED SURFACES[*]

LEANDRO FARINA[†]

Abstract. The evaluation of nearly singular single layer potentials encountered in boundary element methods is treated by a new approach. The potential is expressed as a sum of a one-dimensional integral and a correction term that vanishes for planar surfaces. The small variance of the second term's integrand allows the use of a quasi-Monte Carlo quadrature. Numerical results show that a significant reduction in computational time is obtained over algorithms employing domain subdivisions.

Key words. numerical integration, potential theory, boundary integral method, Monte Carlo method

AMS subject classifications. 65D30, 65R20, 31C20, 65C05

PII. S1064827599363393

1. Introduction.

In boundary element methods, the solution of a boundary value problem is expressed in terms of an integral over the boundary S of the original domain. For potential problems, a solution ϕ is obtained by solving the Fredholm integral equation

$$\lambda(p)\phi(p) + \int_S \phi(q)G(p,q)\ d\sigma = f(p), \quad p \in S, \tag{1.1}$$

where $G = \frac{1}{4\pi}\frac{1}{|p-q|}$. $p = (\xi, \eta, \zeta) \in \mathbb{R}^3$ is called the field point.

Discretization of (1.1) poses the task of evaluating

$$U = \int_S g(q)\ G(p,q)\ d\sigma_q, \tag{1.2}$$

called the single layer potential with density g. A number of methods has been described [4], [10] to deal with integrals where the integrand behaves like the fundamental solution of Laplace's equation. However, usually the assumption that S is of a particular form or has a certain parametrization is made, and/or the case where the integral is nearly singular is ill treated. In this paper, we will relax the usual conditions over the parametrization on S, and we also treat the case where U assumes a moderate to a highly near singular character, as explained below. This is a topic of interest in the boundary element method, in particular, in higher-order methods.

Let S be defined by a mapping $T : \mathcal{P} = [a,b] \times [c,d] \longrightarrow S$ with the property

$$T(\partial\mathcal{P}) = \partial S. \tag{1.3}$$

[*] Received by the editors November 5, 1999; accepted for publication (in revised form) November 27, 2000; published electronically June 5, 2001. This work was supported by the Industry Consortium *Numerical Analysis for Wave Loads on Offshore Structures*, Cambridge, MA, 1997–1999, Mobil, Statoil, DNV Software, Shell, OTRC, Petrobras, NorskHydro, Exxon, Chevron, SAGA, NSWC.

http://www.siam.org/journals/sisc/23-1/36339.html

[†] Research Laboratory of Electronics, MIT, Cambridge, MA 02139. Current address: Centro de Previsão de Tempo e Estudos Climáticos (CPTEC), Instituto Nacional de Pesquisas Espaciais (INPE), Cachoeira Paulista, SP 12630-000, Brazil (farina@cptec.inpe.br).

It is useful to classify U according with the order of magnitude of the distance dist(p,S) between the field point p and the surface S. Thus we say

$$\text{if } \operatorname{dist}(p,S) = O(1), \quad U \text{ is regular,} \tag{1.4}$$

$$\text{if } 0 < \operatorname{dist}(p,S) \le o(1), \quad U \text{ is nearly singular, and} \tag{1.5}$$

$$\text{if } \operatorname{dist}(p,S) = 0, \quad U \text{ is singular.} \tag{1.6}$$

We are mainly interested in the case in (1.5). In this case, the integral U above is regular. However, as the integrand is near a singularity, standard quadrature formulae are not appropriate. This case appears in the boundary element method, typically in applications involving bodies with thin components where one part of S is close to another part of this surface. Moreover, the occurrence of nearly singular integrals is also associated with more general types of domains, depending on how the surface is discretized or panelized. A large difference in size between two panels may create the condition described in (1.5). Recently, it was shown by Luo, Liu, and Berger [7] that conventional boundary integral methods will not degenerate even when applied to thin structures with the thickness to length ratio in the micro (10^{-6}) or nano (10^{-9}) scales. This is true as long as numerical difficulties, such as the calculation of the nearly singular integrals, are addressed.

2. Semianalytical approach. Integrals of type (1.2), where g is a polynomial, can be evaluated in closed form when the surface S is a flat polygon (see Newman [8]). However, for an arbitrary surface S, numerical integration becomes mandatory in the evaluation of U. Our approach here is to use analytical evaluation to a maximum degree in this ultimately numerical task. This will sustain the accuracy and efficiency of an analytical evaluation into the method.

Thus we decompose the surface integral U as

$$U = U_o + U_c, \tag{2.1}$$

where U_o is a planar approximation to U in the sense that it is given as an integral over a flat domain in $\mathbb{R}^3$ and it coincides with U when S is flat. U_c is the correction due to this approximation. As we will see, the inner integral in U_o will be evaluated analytically.

A precise expression for U_o and U_c will be given later. Let us first describe the change of variables defining the auxiliary flat domain.

3. New coordinate systems. The function G presents a weak singularity at $q = (x, y, z) = p$, where it is unbounded. The fact that this function's integral is finite can be easily proved by using polar coordinates, since the jacobian will cancel the singularity. This also suggests a way of evaluating U numerically as a regular integral. Consider a new coordinate system given by

$$(\tilde{x}, \tilde{y}, \tilde{z}) = M^{-1} \begin{pmatrix} x \\ y \\ z \end{pmatrix},$$

where the field point p is the origin lying on the $\tilde{x}\tilde{y}$-plane, denoted by D. M^{-1} is a 3×3 matrix. Using the polar coordinates $\rho = \sqrt{\tilde{x}^2 + \tilde{y}^2}, \theta = \arctan(\tilde{y}/\tilde{x})$ gives

$$U = \frac{1}{4\pi} \int_0^{2\pi} \int_0^{\mathcal{R}(\theta)} g(\rho,\theta) \frac{\rho}{\sqrt{\rho^2 + h^2}} \, \tilde{J} \; d\rho d\theta, \tag{3.1}$$

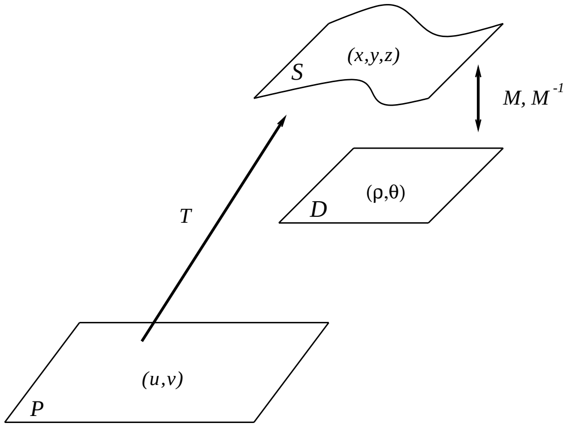


FIG. 1. *The domains $\mathcal{P}, S, D$ and the transformations $T, M,$ and M^{-1}.*

where $h = \tilde{z}(\rho, \theta)$ and $\mathcal{R}(\theta)$ is the value of ρ, as a function of θ, at the boundary of S. $\tilde{J}$ is the jacobian of the transformation M^{-1}, and it is given by $\tilde{J} = \frac{1}{n_D \cdot n_S}$, where n_D and n_S are the respective normal vectors to D and S at q. Because D is flat, $\tilde{J} = n3$, where $n3$ is the third component of n_S.

With this new setting, we see that as q approaches p, the integrand remains bounded and approaches the value of $g\tilde{J}$ at that point. Thus, in our approach, the integral defining U is represented in the physical three-dimensional space.

There are three domains of interest in our problem: the parameter domain

$$\mathcal{P} = [a, b] \times [c, d],$$

the three-dimensional space $\mathbb{R}^3$ (where S is embedded), spanned by the canonical basis

$$\mathcal{C} = \{x, y, z\},$$

or by the alternative basis

$$\mathcal{B} = \{\tilde{x}, \tilde{y}, \tilde{z}\},$$

and the flat domain

$$D = \operatorname{span}\{\tilde{x}, \tilde{y}\} = \operatorname{span}\{\rho, \theta\}$$

where the integrand is regularized.

The main difficulty associated with evaluating (3.1) is that it is not possible, in general, to evaluate $\mathcal{R}, h,$ and $\tilde{J}$ as functions of (ρ, θ). This is a result of the lack of restriction on S. Indeed, we assume S can be *any* parametrized surface with the property (1.3). Then, the fact that the transformation $T : \mathcal{P} = [a, b] \times [c, d] \longrightarrow S$ is not, in general, invertible prevents us from evaluating $\mathcal{R}, h,$ and $\tilde{J}$ directly as functions of variables in S or D (see Figure 1) and therefore from using a quadrature formula where the location of the integration points are predefined. In other words, the control over the integration points is lost; the relative locations of these points in $\mathcal{P}$ will be altered by the transformation T. Thus our approach is to use a interpolation quadrature [2] based on linear functions or on splines for computing U_o and a Monte Carlo method for U_c. We will describe and give details about this approach in the following sections.

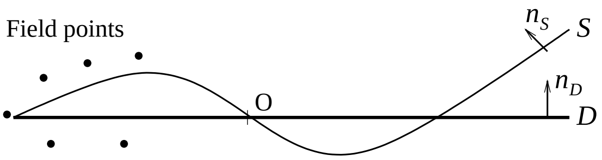


FIG. 2. *The domain D and the surface S.*

4. Independence on the field point. We saw that the change of variables described in the last section, above formula (3.1), provides a regular and simple integrand. However, the new variables are dependent on the field point p. This feature is inconvenient from the computational point of view if the objective is to evaluate U for several, say, L field points, as is usual in boundary element methods for numerical solution of integral equations. Therefore, we will use a similar change of variables, but independent on the field point. In this way, the matrix of the linear transformation between $\mathcal{C}$ and $\mathcal{B}$ is determined once for all computations associated with the surface S. Moreover, the integration points and certain parts of the integrands are computed only once. Then consider $\mathcal{B} = \{\tilde{x}, \tilde{y}, \tilde{z}\}$, as defined in section 3, but with a fixed origin $O = (O_1, O_2, O_3) \in S$, which will be a free parameter specified as one finds appropriate (see Figure 2). The optimum location of O is found by minimizing

$$\max\{|\tilde{z}| : ((\tilde{x}, \tilde{y}, \tilde{z}) - O) \in S\}.$$

Let us assume g may be expanded in powers of $x-\xi$, $y-\eta$, and $z-\zeta$. In view of this, for the evaluation of U, it is sufficient to consider $U^{\mu\nu\upsilon} := \int_S P_{\mu\nu\upsilon}(p-q)\; G(p,q)\; d\sigma_q$, where $P_{\mu\nu\upsilon}(p-q) = (x-\xi)^{\mu}(y-\eta)^{n}u(z-\zeta)^{\upsilon}$.

Now define $U_o^{\mu\nu\upsilon}$ as

$$U_o^{\mu\nu\upsilon} = \frac{1}{4\pi}\int_0^{2\pi}\int_0^{\mathcal{R}(\theta)} P_{\mu\nu\upsilon}(p-q)\frac{\rho}{\sqrt{\rho^2 + b\rho + a}}\; d\rho d\theta, \tag{4.1}$$

where $a = \xi^2 + \eta^2 + (\overline{z} - \zeta)^2$ and $b(\theta) = -2(\xi\cos\theta + \eta\sin\theta)$. The value $\overline{z}$ represents an average value of $\tilde{z}$ and can be taken as $\overline{z} = O_3$.

We now have

$$U^{\mu\nu\upsilon} = U_o^{\mu\nu\upsilon} + U_c^{\mu\nu\upsilon},$$

where

$$U_c^{\mu\nu\upsilon} = \frac{1}{4\pi}\int_0^{2\pi}\int_0^{\mathcal{R}(\theta)} P_{\mu\nu\upsilon}(p-q)\left\{\frac{\rho}{r}\frac{1}{n3} - \frac{\rho}{\overline{r}}\right\}\; d\rho d\theta, \tag{4.2}$$

with $r = |p-q|$ and $\overline{r} = \sqrt{(\tilde{x}-\xi)^2 + (\tilde{y}-\eta)^2 + (\overline{z}-\zeta)^2} = \sqrt{\rho^2 + b\rho + a}$.

5. Quadrature of $\boldsymbol{U_o}$. The objective in this section is to express (4.1) in the form

$$U_o^{nm} = \frac{1}{4\pi}\int_0^{2\pi} F(\mathcal{R}(\theta), \theta)\; d\theta, \tag{5.1}$$

where F is an exact closed form expression for the inner integral in (4.1), and to subsequently apply a one-dimensional quadrature to (5.1). This quadrature will be based on nodes (θ_i, F_i) obtained from a sample of points (u_i, v_i) selected in $\partial\mathcal{P} := \mathcal{P}_1 \cup \mathcal{P}_2 \cup \mathcal{P}_3 \cup \mathcal{P}_4$, where

$$\begin{aligned} \mathcal{P}_1 &= \{(u,v) : v = c\}, \quad \mathcal{P}_2 = \{(u,v) : u = b\}, \\ \mathcal{P}_3 &= \{(u,v) : v = d\}, \quad \mathcal{P}_4 = \{(u,v) : u = a\}. \end{aligned}$$

Condition (1.3) assures that $\mathcal{R}$ is in fact evaluated at the boundary of S. This is the only place where (1.3) is used. Thus the quadrature nodes (θ_i, F_i) are obtained without the knowledge of their exact location. For smooth integrands F, a quadrature based on interpolating cubic splines or Hermite functions [5] can be used. Alternatively, noninterpolating methods could be used, such as the locally corrected quadrature proposed by Strain [9], where singularities are allowed in the integrand F.

In order to make $F(\mathcal{R}(\theta), \theta)$ explicit, note that

$$(x, y, z) = M \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{pmatrix} - O,$$

where $M = \{M_{ij}\}$ is a 3×3 matrix. It follows that on D we have

$$\begin{cases} x = M_{11}\ \rho\cos\theta + M_{12}\ \rho\sin\theta - O_1, \\ y = M_{21}\ \rho\cos\theta + M_{22}\ \rho\sin\theta - O_2, \\ z = M_{31}\ \rho\cos\theta + M_{32}\ \rho\sin\theta - O_3. \end{cases}$$

Therefore,

$$F(\mathcal{R}(\theta), \theta) = \int_0^{\mathcal{R}(\theta)} (c\rho + d)^\mu (e\rho + f)^\nu (h\rho + l)^\upsilon \frac{\rho}{\sqrt{\rho^2 + b\rho + a}}\, d\rho, \tag{5.2}$$

where $d = -\xi - O_1$, $f = -\eta - O_2$, $l = -\zeta - O_3$, $c(\theta) = M_{11}\cos\theta + M_{12}\sin\theta$, $e(\theta) = M_{21}\cos\theta + M_{22}\sin\theta$, and $h(\theta) = M_{31}\cos\theta + M_{32}\sin\theta$. Expanding the binomials in (5.2), we get

$$\begin{aligned} F(\mathcal{R}(\theta), \theta) = &\sum_{i=0}^{\mu}\sum_{j=0}^{\nu}\sum_{k=0}^{\upsilon} \binom{\mu\nu\upsilon}{ijk} d^i f^j l^k c^{\mu-i}(\theta) e^{\nu-j}(\theta) h^{\upsilon-k}(\theta) \\ &\times H_{\mu+\nu+\upsilon-i-j-k+1}(\theta), \end{aligned}$$

where $\binom{\mu\nu\upsilon}{ijk} := \binom{\mu}{i}\binom{\nu}{j}\binom{\upsilon}{k}$ and

$$H_t(\theta) = \int_0^{\mathcal{R}(\theta)} \frac{\rho^t}{\sqrt{\rho^2 + b\rho + a}}\, d\rho. \tag{5.3}$$

The integral (5.3) can be recursively evaluated using [3]

$$H_t(\theta) = \frac{\rho^{t-1}}{t}\sqrt{\rho^2 + b\rho + a} - \frac{(2t-1)b}{2t} H_{t-1}(\theta) - \frac{(t-1)a}{t} H_{t-2}(\theta),$$

with the starting functions

$$H_0(\theta) = \begin{cases} \log|2\rho + b + 2\sqrt{\rho^2 + b\rho + a}|\Big|_{\rho=0}^{\rho=\mathcal{R}(\theta)}, & 4a \neq b^2, \\ \log|b + 2\rho|\Big|_{\rho=0}^{\rho=\mathcal{R}(\theta)}, & 4a = b^2, \end{cases} \tag{5.4}$$

$$H_1(\theta) = \sqrt{\rho^2 + b\rho + a}\Big|_{\rho=0}^{\rho=\mathcal{R}(\theta)} - \frac{b}{2}H_0(\theta). \tag{5.5}$$

6. Quadrature of U_c by quasi-Monte Carlo. The evaluation of expression (4.2) is more complex than the numerical integration of (4.1). In the case of U_o, it was possible to separate the variables ρ and θ in order to evaluate the inner integral analytically. Because $n3$ is an arbitrary function associated with the given surface S, the correction component U_c does not permit a similar variable separation. Thus U_c has to be integrated numerically as a two-dimensional integral. Since U_c presents the same discretization condition as U_o, namely, the location of the nodes in the (ρ, θ) variables is arbitrary, this makes the problem more delicate. The apparent solution[1] seems to use interpolation in two variables, imitating the procedure for evaluating (5.1). However, we believe that interpolation provides neither more efficiency nor more accuracy than a Monte Carlo quadrature. Traditionally, since its convergence rate is independent of the problem dimension, Monte Carlo quadrature has been used in high-dimensional integrals as an efficient and robust alternative to grid-based methods. In what follows we will outline the reasons for using a Monte Carlo type of quadrature in a low-dimensional (i.e., two-dimensional) integral and describe the specific approach employed.

Let $(\rho_i, \theta_i)_{i=\{1,\ldots,N\}}$ be a sequence of points in D. To evaluate (4.2) we use an integration formula Q_N of the form

$$Q_N(U_c) = A(D)\mu_N(K), \tag{6.1}$$

where

$$A(D) = \frac{1}{4\pi}\int_0^{2\pi} \mathcal{R}(\theta)\, d\theta,$$

$$K = P_{\mu\nu v}(p - q)\left\{\frac{\rho}{r}\frac{1}{n3} - \frac{\rho}{r}\right\},$$

and μ_N denotes the sample mean of K on D, given as

$$\mu_N(K) = \frac{1}{N}\sum_{i=1}^{N} K(\rho_i, \theta_i).$$

Note that the evaluation of $A(D)$ is intimately related with the numerical integration of U_o. From (5.1), we see that $A(D)$ is a special case, where $F(\mathcal{R}(\theta), \theta) \equiv \mathcal{R}(\theta)$.

[1] Strain's approach [9] does not seem applicable here because of its restriction that the domain of integration must be a hypercube.

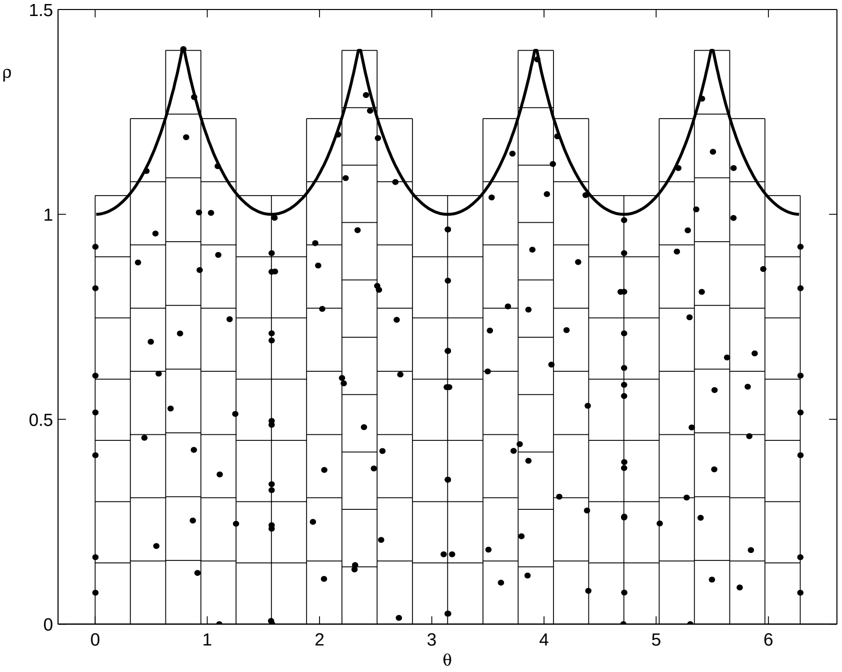


FIG. 3. *The effect of the partitioning algorithm represented by the location of the integration nodes in the $\theta - \rho$ plane bounded by $\mathcal{R}(\theta)$. Here $N_c = 1$, and S is a curved quadrilateral.*

As is well known in the theory of Monte Carlo methods, there are basically two resources for improving the convergence rate of our approximation (6.1) to U_c. One is to force the nodes to be as uniform as possible, and the other is to reduce somehow the variance

$$\sigma^2(K) := \int_D (K - \overline{K})^2 \; d\rho d\theta,$$

where $\overline{K}$ is the mean of the integrand K.

We will use both means to obtain a suitable algorithm for the computation of U_c. The particular procedure to achieve this goal in the correction integral will be described next.

The control variates [1], [6, Chapter 6] form of variance reduction is in fact used from the beginning in our approach to evaluate the single layer potential as in the decomposition (2.1). The term

$$\left\{ \frac{\rho}{r} \frac{1}{n3} - \frac{\rho}{\overline{r}} \right\}$$

in K vanishes where the S is flat and, for moderate curvatures in S, presents small variances.

The other resource that we will use to improve convergence rates is the *uniformization* of the nodes distribution over the domain of integration.

The sequence of the nodes will fill the integration domain more uniformly and less uncorrelatedly than random nodes characterizing a quasi-Monte Carlo formula. For

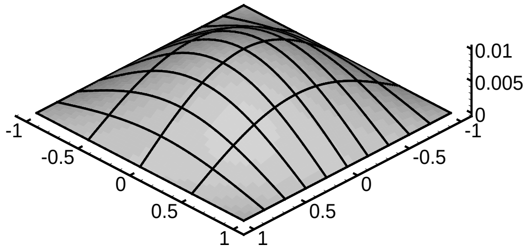


FIG. 4. *The surface S_c.*

our case, the transformations T and M would void any deterministic choice of points in $\mathcal{P}$. In order to obtain low-discrepancy nodes, we combine the *stratification* and the *acceptance-rejection* methods [6] into a partitioning algorithm. Thus we divide the $\rho-\theta$ domain in approximately equal sized *cells*, or subregions, and we accept or reject the pseudorandom generated points for the formula (6.1) until an equal number of subnodes N_c are present in each cell. A typical example of the effect of this algorithm can be seen in Figure 3, where the distribution of the integration nodes in the $\theta-\rho$ plane is represented. In this figure, the bounding curve is $\mathcal{R}(\theta)$, given for a curved quadrilateral with equal, straight sides, and the cells are rectangles with one subnode.

7. Numerical results. In this section we present numerical results obtained from the Monte Carlo-based method described above and compare them with a standard quadrature employing the Gauss–Legendre formula combined with subdivision of the integration domain. This subdivision takes place whenever $\operatorname{dist}(p, S)$ becomes small compared with the area of S. All results in this section are believed to have relative error less than 0.3%.

Let S_c be a curved quadrilateral given by

$$S_c : (x(u,v), y(u,v), z(u,v)) = T(u,v) = (u, v, 0.01\sin u \sin v + 0.01)$$

and represented in Figure 4. In Figure 5, the dotted line represents a subdivision method, and the solid line indicates quasi-Monte Carlo. The computing time (on a UNIX workstation) is plotted as a function of the number L of single layer potentials computed, each layer potential corresponding to a field point. Thus up to 1000 field points p_i were chosen satisfying the condition $\operatorname{dist}(p_i, S) \geq 10^{-2}$. For $L \approx 28$, quasi-Monte Carlo requires less computational time than a standard subdivision method. The graph shows monotonically increasing linear functions, and for a large number of evaluations (1000 field points), we see a factor of 3 difference between the two methods. This difference is due to the fact that in our field point independent approach, information such as the location of the integration nodes and parts of the integrands are reused in all the integrals evaluated. In particular, calls to a subroutine or function defining the transformation T are made only for the first integral. Furthermore, the value of $A(D)$ in (6.1) is fixed. Hence the effort to evaluate successive integrals becomes minimal with the increase of L.

Let us now comment on another example. Suppose now that the surfaces over which the integration is done are a family of planar quadrilaterals given as

$$S_\epsilon : (x(u,v), y(u,v), z(u,v)) = T(u,v) = (u, v, \epsilon),$$

where $10^{-7} \leq \epsilon \leq 0.1$. In this case, $U_c = 0$, and the Monte Carlo method is not

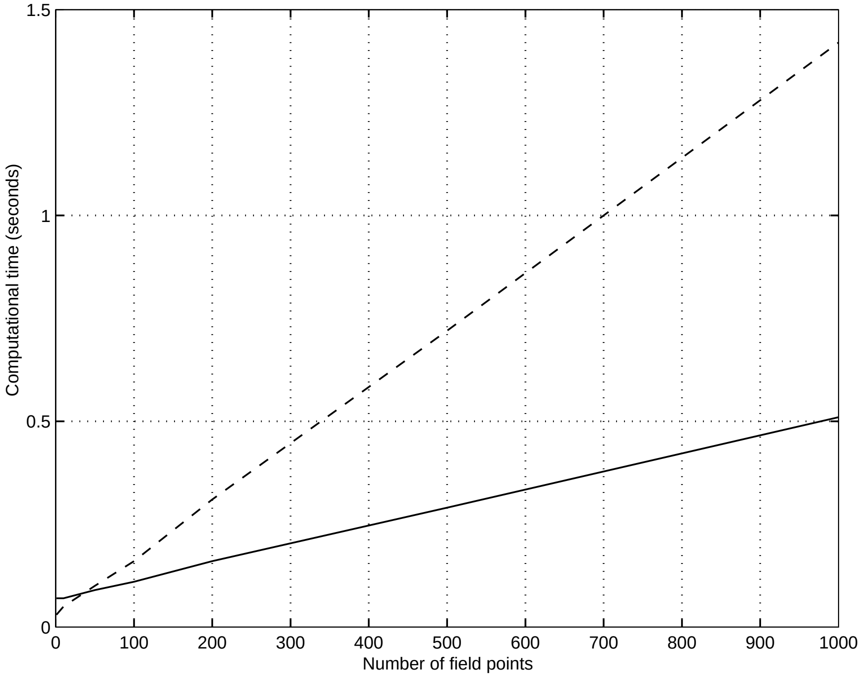


FIG. 5. *The results for the surface S_c and for dist$(p_i, S) \geq 10^{-2}$. The dotted line represents the subdivision method, and the solid line indicates the quasi-Monte Carlo method.*

necessary to use. The integrals are evaluated by one-dimensional quadrature since

$$U = U_o.$$

In Figure 6, with the same representations for the dotted and solid line, we see the computational time as a function of $\mathrm{dist}(p_i, s) = \epsilon$, in the range of ϵ specified above. Here $L = 500$, and the field points are distributed uniformly over the x-axis. As can be seen, the computation of U_o is not affected by the degree of the near singularity, represented by ϵ. On the other hand, the use of standard quadratures, even with domain subdivision, suffers from the necessity of an excessive number of nodes and/or domain subdivisions.

It should be noted that under certain conditions the algorithm presented here will not be appropriate in its present form. Although it will provide good results for the case (1.4) because the integrand will be smooth and well approximated by polynomials, its efficiency will not be superior to a regular quadrature with a small number of nodes. Also, for surfaces with large curvature, the contribution from U_c will be large compared with U_o, and a large number of nodes may be required to sustain accuracy.

8. Conclusion. A method for computing nearly singular single layer potentials has been introduced. Restrictions on the parametrization of the integration domain are relaxed, and the original integral is decomposed in a sum two terms, U_o and U_c, where the first is an approximation which coincides with the single layer potential when the surface is planar. The term U_c provides the correction when the surface loses its planar character. This decomposition not only allows U_o to be evaluated in terms

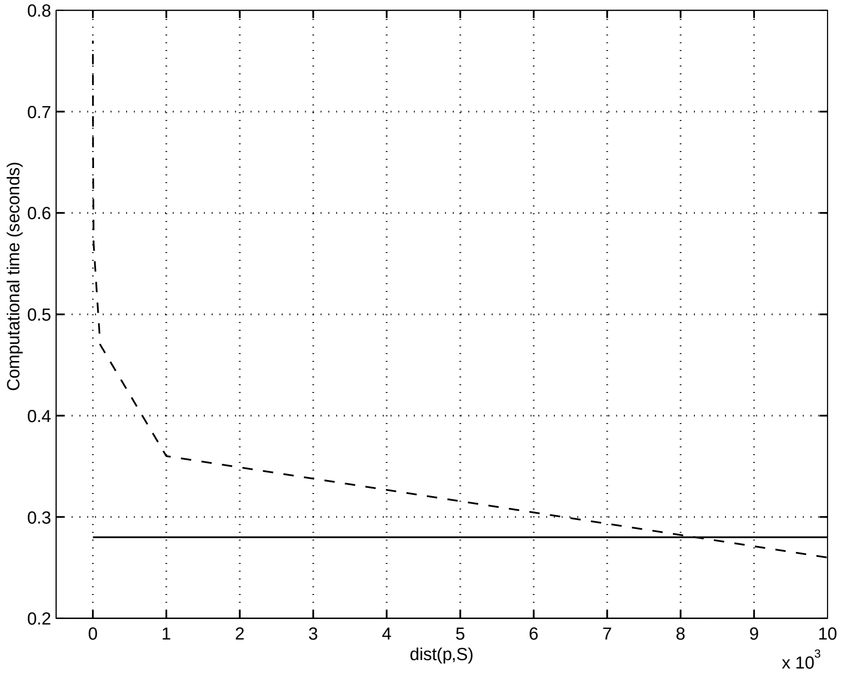


FIG. 6. *The results for S_ϵ and for $L = 500$. The dotted line represents the subdivision method, and the solid line indicates the quasi-Monte Carlo method.*

of a one-dimensional regular quadrature but also reduces the integrand variance in U_c, making it viable to apply a two-dimensional quadrature based on pseudorandom nodes. This latter quadrature also employs a uniformization algorithm to the nodes distribution in order to improve the formula convergence rate. Also, because the quadrature of U_c is not grid-based, adaptiveness can be easily incorporated.

Usually, in boundary element methods, integrals like U have to be computed several times according with different field points. The method described above explores this fact by reusing information from previous computations. Numerical results clearly show an economy in computational costs for this approach when compared with standard quadrature employing domain subdivisions. For surfaces with moderate curvature, previously prohibitive nearly singular integrals where the ratio distance from the field point to the surface average length reaches 10^{-7} or less can be dealt with using the present algorithm without significant increase in the computational cost.

It may be possible to extend this approach to other types of integrands, such as the double layer potential. In order to achieve this successfully, one needs to efficiently integrate the strong near singularity that will be present in U.

Acknowledgments. The author thanks the people at the Computational Hydrodynamics Facility at MIT for helpful discussions about this work.

REFERENCES

[1] R. E. CAFLISCH, *Monte Carlo and quasi-Monte Carlo methods*, in Acta Numerica, 1998, Acta Numer. 7, Cambridge University Press, Cambridge, UK, 1998, pp. 1–49.

[2] P. J. Davis and P. Rabinowitz, *Methods of Numerical Integration*, 2nd ed., Academic Press, Orlando, FL, 1984.

[3] I. S. Gradshteyn and I. M. Ryshik, *Table of Integrals, Series, and Products*, 5th ed., Academic Press, Boston, 1994.

[4] W. Hackbusch and S. A. Sauter, *On numerical cubatures of nearly singular surface integrals arising in BEM collocation*, Computing, 52 (1994), pp. 139–159.

[5] D. Kahaner, C. B. Moler, and S. Nash, *Numerical Methods and Software*, Prentice Hall, Englewood Cliffs, NJ, 1989.

[6] A. R. Krommer and C. W. Ueberhuber, *Computational Integration*, SIAM, Philadelphia, 1998.

[7] J. Luo, Y. Liu, and E. Berger, *Analysis of two-dimensional thin structures (from micro- to nano-scales) using the boundary element method*, Comput. Mech., 22 (1998), pp. 404–412.

[8] J. N. Newman, *Distributions of sources and normal dipoles over a quadrilateral panel*, J. Engrg. Math., 20 (1986), pp. 113–126.

[9] J. Strain, *Locally corrected multidimensional quadrature rules for singular functions*, SIAM J. Sci. Comput., 16 (1995), pp. 992–1017.

[10] J. C. F. Telles, *A self-adaptive co-ordinate transformation for efficient numerical evaluation of general boundary element integrals*, Internat. J. Numer. Methods Engrg., 24 (1987), pp. 959–973.

SIAM J. SCI. COMPUT.
Vol. 23, No. 1, pp. 92–108

A FETI PRECONDITIONER FOR TWO DIMENSIONAL EDGE ELEMENT APPROXIMATIONS OF MAXWELL'S EQUATIONS ON NONMATCHING GRIDS[*]

FRANCESCA RAPETTI[†] AND ANDREA TOSELLI[‡]

Abstract. A class of FETI methods for the mortar approximation of a vector field problem in two dimensions is proposed. Edge element discretizations of lowest degree are considered. The method proposed can be employed with geometrically conforming and nonconforming partitions. Our numerical results show that its condition number increases only with the number of unknowns in each subdomain and is independent of the number of subdomains and the size of the problem.

Key words. edge elements, Maxwell's equations, domain decomposition, FETI, preconditioners, nonmatching grids

AMS subject classifications. 65F10, 65N22, 65N30, 65N55

PII. S1064827500366999

1. Introduction.

In this paper, we consider the boundary value problem

$$\begin{aligned} L\mathbf{u} := \mathbf{curl}\,(a\,\mathrm{curl}\,\mathbf{u}) + A\,\mathbf{u} &= \mathbf{f} \quad \text{in } \Omega, \\ \mathbf{u}\cdot\mathbf{t} &= 0 \quad \text{on } \partial\Omega, \end{aligned} \tag{1}$$

with Ω a bounded polygonal domain in $\mathbb{R}^2$. Here

$$\mathbf{curl}\,v := \begin{bmatrix} \dfrac{\partial v}{\partial x_2} \\ -\dfrac{\partial v}{\partial x_1} \end{bmatrix}, \quad \mathrm{curl}\,\mathbf{u} := \frac{\partial u_2}{\partial x_1} - \frac{\partial u_1}{\partial x_2};$$

see, e.g., [17]. The coefficient matrix A is a symmetric, uniformly positive definite matrix-valued function with entries $A_{ij} \in L^\infty(\Omega)$, $1 \le i,j \le 2$, and $a \in L^\infty(\Omega)$ is a positive function bounded away from zero. The domain Ω has unit diameter, and $\mathbf{t}$ is the unit tangent to its boundary.

The weak formulation of problem (1) requires the introduction of the Hilbert space $H(\mathrm{curl}\,;\Omega)$, defined by

$$H(\mathrm{curl}\,;\Omega) := \left\{\mathbf{v} \in (L^2(\Omega))^2 | \quad \mathrm{curl}\,\mathbf{v} \in L^2(\Omega)\right\}.$$

The space $H(\mathrm{curl}\,;\Omega)$ is equipped with the following inner product and graph norm:

$$(\mathbf{u},\mathbf{v})_{\mathrm{curl}} := \int_\Omega \mathbf{u}\cdot\mathbf{v}\,d\mathbf{x} + \int_\Omega \mathrm{curl}\,\mathbf{u}\,\mathrm{curl}\,\mathbf{v}\,d\mathbf{x}, \qquad \|\mathbf{u}\|^2_{\mathrm{curl}} := (\mathbf{u},\mathbf{u})_{\mathrm{curl}}.$$

[*] Received by the editors February 4, 2000; accepted for publication (in revised form) November 30, 2000; published electronically June 5, 2001.

http://www.siam.org/journals/sisc/23-1/36699.html

[†] ASCI–UPR 9029 CNRS, Paris Sud University, Building 506, 91403 Orsay Cedex, France (rapetti@asci.fr, http://www.asci.fr/Francesca.Rapetti). The work of this author was supported by the European Community under contract TMR ERB4001GT965424 and by the Direction des Rélations Internationnelles du CNRS, France, under the fund PICS 478.

[‡] Seminar for Applied Mathematics, ETH Zürich, Rämistrasse 101, CH-8092 Zürich, Switzerland (toselli@sam.math.ethz.ch). The work of this author was carried out and completed while the author was affiliated to the Courant Institute of Mathematical Sciences, New York, and was supported in part by the Applied Mathematical Sciences Program of the U.S. Department of Energy under contract DEFGO288ER25053.

The tangential component $\mathbf{u} \cdot \mathbf{t}$, of a vector $\mathbf{u} \in H(\mathrm{curl}\,;\Omega)$ on the boundary $\partial\Omega$, belongs to the space $H^{-\frac{1}{2}}(\partial\Omega)$; see [17, 8]. The subspace of vectors in $H(\mathrm{curl}\,;\Omega)$ with a vanishing tangential component on $\partial\Omega$ is denoted by $H_0(\mathrm{curl}\,;\Omega)$.

For any $\mathcal{D} \subset \Omega$, we define the bilinear form

$$a_{\mathcal{D}}(\mathbf{u}, \mathbf{v}) := \int_{\mathcal{D}} (a \,\mathrm{curl}\,\mathbf{u}\,\mathrm{curl}\,\mathbf{v} + A\,\mathbf{u} \cdot \mathbf{v})\, d\mathbf{x}, \quad \mathbf{u}, \mathbf{v} \in H(\mathrm{curl}\,;\Omega). \tag{2}$$

The variational formulation of (1) is as follows.

Find $\mathbf{u} \in H_0(\mathrm{curl}\,;\Omega)$ such that

$$a_{\Omega}(\mathbf{u}, \mathbf{v}) = \int_{\Omega} \mathbf{f} \cdot \mathbf{v}\, d\mathbf{x}, \quad \mathbf{v} \in H_0(\mathrm{curl}\,;\Omega). \tag{3}$$

We discretize this problem using edge elements, also known as Nédélec elements; see [24]. These are vector-valued finite elements that ensure only the continuity of the tangential component across the common side of adjacent mesh triangles, as is physically required for the electric and magnetic fields, which are solutions of Maxwell's equations.

In this paper, we consider a mortar approximation of this problem. The computational domain is partitioned into a family of nonoverlapping subdomains, and independent triangulations are introduced in each subdomain. The weak continuity of the tangential component of the solution is then enforced by using suitable integral conditions that require that the jumps across the subdomain inner boundaries are perpendicular to suitable finite element spaces defined on the edges of the partition. The mortar method was originally introduced in [9] for finite element approximations in H^1. Mortar approximations for edge element approximations in two and three dimensions have been studied in [3] and [4], respectively. There has also been additional recent work for the case of sliding meshes for the study of electromagnetic fields in electrical engines; see, e.g., [25, 26].

The applications that we have in mind are mainly problems arising from static and quasi-static Maxwell equations (eddy current problems); see, e.g., [6, 5]. In this paper, we consider only the model problem (3), where the dependency on the time variable or on the frequency has been eliminated, and we generically refer to it as Maxwell's equations. A good preconditioner for this model problem is the first step for the efficient solution of linear systems arising from the edge element approximation of static problems, and of time- or frequency-dependent problems arising from the quasi-static approximation of Maxwell's equations.

The aim of this paper is to build an iterative method of finite element tearing and interconnecting (FETI) type for a mortar edge element approximation of problem (1). FETI methods were first introduced for the solution of conforming approximations of elasticity problems in [15]. In this approach, the original domain Ω is decomposed into nonoverlapping subdomains Ω_i, $i = 1, \ldots, N$. On each subdomain Ω_i a local stiffness matrix is obtained from the finite element discretization of $a_{\Omega_i}(\cdot, \cdot)$. Analogously, a set of right-hand sides is built. The continuity of the solution corresponding to the primal variables is then enforced by using Lagrange multipliers across the interface defined by the subdomain inner boundaries. In the original FETI algorithm, the primal variables are then eliminated by solving local Neumann problems, and an equation in the Lagrange multipliers is obtained. Several preconditioners have been proposed and studied for its solution; see, e.g., [14, 16, 23, 13, 30, 27, 19, 33].

Many iterative methods for the solution of linear systems arising from mortar approximations have been proposed. We cite, in particular, [22, 1, 20, 2, 10, 11, 7, 12, 21, 34] and refer to the references therein for a more detailed discussion.

To our knowledge, the application of FETI-type preconditioners to mortar approximations was first explored in [21, 18] and then tested more systematically in [29]. The idea is fairly simple and relies on the observation that mortar approximations with Lagrange multipliers and FETI formulations, where the pointwise continuity across the substructures is enforced by using Lagrange multipliers, give rise to indefinite linear systems that have the same form. FETI preconditioners can then be devised for mortar approximations in a straightforward way; see [29]. In this paper, we apply the FETI preconditioner introduced in [19] for the case of nonredundant Lagrange multipliers to the mortar approximation originally studied in [3]. Our work generalizes our previous study of FETI preconditioners for two dimensional conforming edge element approximations in [33]. As opposed to the H^1 case, the generalization of FETI preconditioners to mortar approximations requires some modifications in $H(\mathrm{curl}\,)$. More precisely, the coarse components of the preconditioners need to be modified here in order to obtain a scalable method, and a suitable scaling matrix Q has to be introduced; see section 5. As shown in [29], no modification appears to be necessary for nodal finite elements in H^1. Finally, we note that in this paper we consider only problems without jumps of the coefficients. For conforming approximations, FETI methods that are robust with respect to large variations of jumps of the coefficients have been developed and studied (see [27, 19]), but the case of nodal or edge element approximations on nonmatching grids still needs to be explored and is left for a future work.

The outline of the remainder of this paper is as follows. In section 2, we introduce a partition of the domain Ω and local finite element spaces. In section 3, we consider the mortar condition, and in section 4, we present our FETI method, in terms of a projection onto a low dimensional subspace and a local preconditioner. The expressions for the projection and the preconditioner are then given in section 5, and some numerical results for geometrically conforming and nonconforming partitions are presented in section 6.

2. Finite element spaces. We first consider a nonoverlapping partition of the domain Ω,

$$\mathcal{F}_H = \left\{ \Omega_i,\ i=1,\dots,N \;\middle|\; \bigcup_{i=1}^{N} \overline{\Omega_i} = \overline{\Omega} \quad ; \quad \Omega_k \cap \Omega_l = \emptyset \quad , \quad 1 \le k < l \le N \right\},$$

such that each subdomain Ω_i is a connected polygonal open set in $\mathbb{R}^2$. We remark that $\mathcal{F}_H$ does not need to be geometrically conforming. We denote the diameter of Ω_i by H_i and the maximum of the diameters of the subdomains by H:

$$H := \max_{1 \le i \le N} \{H_i\}.$$

The elements of $\mathcal{F}_H$ are also called *substructures.* Let $\mathbf{t}_i$ be the unit tangent to $\partial\Omega_i$, chosen so that, following the direction of $\mathbf{t}_i$, Ω_i is on the left.

For every subdomain Ω_i, we define the set of its open edges that do not lie on $\partial\Omega$ by $\{\Gamma^{i,j} |\, j \in \mathcal{I}_i\}$. We then define the interface Γ, also called the “skeleton” of the decomposition, as the union of the edges of $\mathcal{F}_H$ that do not lie on $\partial\Omega$:

$$\Gamma := \bigcup_{i=1}^{N} \partial\Omega_i \setminus \partial\Omega = \bigcup_{i=1}^{N} \bigcup_{j\in\mathcal{I}_i} \Gamma^{i,j}.$$

We also define the local spaces of restrictions of vectors in $H_0(\mathrm{curl}\,;\Omega)$ to Ω_i:

$$H_\star(\mathrm{curl}\,;\Omega_i) := \{\mathbf{u}_i \in H(\mathrm{curl}\,;\Omega_i)|\ \mathbf{u}_i \cdot \mathbf{t} = 0 \text{ on } \partial\Omega \cap \partial\Omega_i\}.$$

For every substructure Ω_i, we consider a triangulation $\mathcal{T}_{i,h}$ made of triangles or rectangles. Let $\mathcal{E}_{i,h}$ be the set of edges of $\mathcal{T}_{i,h}$. For every edge $e \in \mathcal{E}_{i,h}$, we fix a direction, given by a unit vector $\mathbf{t}_e$. The length of the edge e is denoted by $|e|$. The local triangulations are assumed to be shape-regular and quasi-uniform, and they do not need to match across the inner boundaries of the subdomains. We define h as the maximum of the mesh-sizes of the triangulations.

We next consider the lowest-order Nédélec finite element spaces, originally introduced in [24], defined on each subdomain Ω_i as

$$X_h(\Omega_i) = X_i := \{\mathbf{u}_i \in H_\star(\mathrm{curl}\,;\Omega_i)|\ \mathbf{u}_{i|_t} \in \mathcal{R}(t),\ t \in \mathcal{T}_{i,h}\},$$

where, in the case of triangular meshes, we have

$$\mathcal{R}(t) := \left\{ \left[\begin{array}{c} \alpha_1 + \alpha_3 x_2 \\ \alpha_2 - \alpha_3 x_1 \end{array} \right] \mid \alpha_k \in \mathbb{R} \right\}.$$

We recall that the tangential component of a vector $\mathbf{u}_i \in X_i$ is constant on the edges of the triangulation $\mathcal{T}_{i,h}$ and that the degrees of freedom can be chosen as the values of the tangential component on the edges

$$\lambda_{e_k}(\mathbf{u}_i) = u_k^{(i)} := \mathbf{u}_i \cdot \mathbf{t}_{e_k|_{e_k}}, \quad e_k \in \mathcal{E}_{i,h}. \tag{4}$$

We next introduce the product space

$$X_h(\Omega) = X := \prod_{i=1}^{N} X_i \subset \prod_{i=1}^{N} H_\star(\mathrm{curl}\,;\Omega_i),$$

the spaces of tangential vectors

$$W_h(\partial\Omega_i) = W_i := \{(\mathbf{u}_i \cdot \mathbf{t}_i)\,\mathbf{t}_i \text{ restricted to } \partial\Omega_i \setminus \partial\Omega\ \mid \mathbf{u}_i \in X_i\},$$

and the product space

$$W_h(\Gamma) = W := \prod_{i=1}^{N} W_i.$$

We note that we have chosen a different definition of the trace spaces than that employed in [33]. Here, the spaces W_i consist of piecewise constant tangential *vectors* on $\partial\Omega_i \setminus \partial\Omega$.

Throughout this paper, we will use the following conventions. We will use the same notation for the vectors in X_i and tangential vectors in W_i. We denote a generic vector function in X_i using a bold letter with the subscript i, e.g., $\mathbf{u}_i$, and

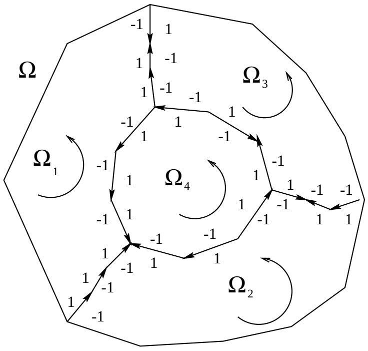


FIG. 1. *Example of a partition of the domain Ω. We show the directions of the subdomain boundaries, given by the unit vectors $\{\mathbf{t}_i\}$, those of the fine edges on the interface Γ, and the corresponding values of the degrees of freedom $t^{(i)}$.*

the column vector of its degrees of freedom, defined in (4), using the same letter with the superscript (i), e.g., $u^{(i)}$. Its kth degree of freedom corresponding to the edge e_k, defined in (4), is $u_k^{(i)}$. A generic vector in the product space X (or W) is also denoted by a bold letter, e.g., $\mathbf{u}$, and the corresponding vector of degrees of freedom by the same letter, e.g., u. We will use the same notation for the spaces of functions X_i and W_i and the corresponding spaces of degrees of freedom.

Given the unit vectors $\mathbf{t}_i$, the column vectors $t^{(i)}$ are defined by

$$t_k^{(i)} := \mathbf{t}_i \cdot \mathbf{t}_{e_k}, \quad e_k \subset \partial\Omega_i \setminus \partial\Omega, \quad e_k \in \mathcal{E}_{i,h}.$$

We will need these tangential vectors in the definition of our FETI method; see section 5. We remark that in case all the edges e_k on $\partial\Omega_i$ have the same direction of the boundary $\partial\Omega_i$, the entries of the vector $t^{(i)}$ are equal to one. Figure 1 shows an example of a partition, with the directions of the subdomain boundaries and of the fine edges on the interface Γ, and the corresponding degrees of freedom $t^{(i)}$.

Finally, for $i = 1, \dots, N$, we define the discrete harmonic extensions with respect to the bilinear forms $a_{\Omega_i}(\cdot,\cdot)$ into the interior of Ω_i:

$$\mathcal{H}_i : W_i \longrightarrow X_i.$$

We recall that $\mathcal{H}_i\mathbf{u}_i$ minimizes the energy $a_{\Omega_i}(\mathcal{H}_i\mathbf{u}_i, \mathcal{H}_i\mathbf{u}_i)$ among all the vectors of X_i with tangential component equal to $\mathbf{u}_i$ on $\partial\Omega_i \setminus \partial\Omega$.

3. A mortar condition. The mortar method presented in this section was originally developed and studied in [3]. We consider the skeleton Γ and choose a splitting of Γ as the disjoint union of some edges $\{\overline{\Gamma}^{k,j}\}$, which we call *mortars*. We note that this partition is not in general unique; see Figure 2 (left) for an example of decomposition.

A unique set of indices corresponds to this choice, and we denote it by

$$\mathcal{I}_M := \{m = (k,j) \text{ such that } \Gamma^{k,j} \text{ is a mortar }\}.$$

To simplify the notation, we denote the mortars by $\{\Gamma^m |\ m \in \mathcal{I}_M\}$. We have

$$\overline{\Gamma} := \bigcup_{i=1}^{M} \overline{\Gamma}^m, \quad \Gamma^m \cap \Gamma^n = \emptyset, \qquad \text{if } m \neq n \text{ and } n, m \in \mathcal{I}_M.$$

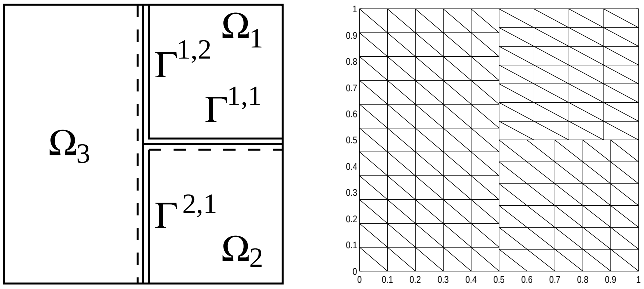


Fig. 2. *An example where the domain is decomposed into three (rectangular) nonoverlapping subdomains. The skeleton has two partitions (figure on the left): one in terms of mortars (solid dark line) and the other in terms of nonmortars (long dashed lines). On the right is an example of discretization of the subdomains by means of triangular grids that do not match at the interfaces between adjacent subdomains.*

For any m in $\mathcal{I}_M$, we denote by W^m the space $W^{k,j}$ $(m = (k, j) \in \mathcal{I}_M)$ given by

$$W^{k,j} := \{(\mathbf{u}_k \cdot \mathbf{t}_k)\, \mathbf{t}_k \text{ restricted to } \Gamma^{k,j} \ \mid \mathbf{u}_k \in X_k\}.$$

We note that the vectors in $W^{k,j}$ are also the restrictions of vectors in W_k to $\Gamma^{k,j}$. Before introducing the mortar space, we need to fix a last point. Let Γ^m be a mortar edge with $m = (k, j)$ and $\mathbf{u} \in X_h$: for almost every $\mathbf{x} \in \Gamma^m$, there exists an index l $(1 \leq l \leq N)$, $l \neq k$ such that $\mathbf{x} \in \Gamma^m \cap \partial\Omega_l$. At this point $\mathbf{x}$ we have two fields, namely, $\mathbf{u}_k$ and $\mathbf{u}_l$. Since the domain decomposition is in general nonconforming, the value of l depends on $\mathbf{x}$, and we denote by $\mathcal{J}_m$ the set of indices l $(1 \leq l \leq N)$ such that $\Gamma^m \cap \partial\Omega_l \neq \emptyset$. We then define

$$\mathbf{u}_{-k}(\mathbf{x}) := \mathbf{u}_l(\mathbf{x}), \quad \mathbf{x} \in \Gamma^m \cap \partial\Omega_l, \quad l \in \mathcal{J}_m.$$

The function $\mathbf{u}_{-k}$ is defined for almost all $\mathbf{x} \in \Gamma^m$. In general, it is not the tangential component at Γ^m of a field $\mathbf{u} \in H_\star(\mathrm{curl}\,; \Omega_i)$: it can indeed correspond to tangential components from different subdomains which share a subset of Γ^m and live on different grids.

The equality between $\mathbf{u}_{-k} \cdot \mathbf{t}_k$ and $\mathbf{u}_k \cdot \mathbf{t}_k$ at Γ^m becomes too stringent a condition since the two fields are in general defined on different and nonmatching grids. As is usually done in nonconforming mortar domain decomposition methods, we impose these constraints in a weak form by means of suitable Lagrange multipliers. Here, the Lagrange multiplier space consists of the tangential components of the shape functions at the mortar edges; see [3].

Remark 3.1. The definition of the mortar space for the edge elements is simpler than for nodal finite elements. In the nodal case, the space of Lagrange multipliers cannot be chosen as a space of traces on the mortar edges but only as a suitable subspace of it. In the edge case, it is not necessary to decrease the dimension of the multiplier space since the information is associated to edges and not to nodes; see [9] for more details.

The Lagrange multiplier (mortar) space is now defined by

$$\{\mathbf{v} \in L^2(\Gamma) \mid \mathbf{v}_{|\Gamma^m} \in W^m, \quad m \in \mathcal{I}_M\}.$$

We remark that this is a space of tangential vectors on Γ. The transmission conditions at the interface between adjacent subdomains are then weakly imposed by means of these Lagrange multipliers. A solution $\mathbf{u} \in W$ is required to satisfy the constraints

$$(5) \qquad \int_{\Gamma^m} (\mathbf{u}_k \cdot \mathbf{t}_k - \mathbf{u}_{-k} \cdot \mathbf{t}_k\,)\, \mathbf{v} \cdot \mathbf{t}_k \, ds = 0, \quad \mathbf{v} \in W^m, \quad m = (k,j) \in \mathcal{I}_M.$$

The set of transmission conditions can be expressed in matrix form in the following way.

Let $\mathbf{w}_i^k$ be the basis function associated to the kth mesh edge of $\partial\Omega_i \setminus \partial\Omega$. We introduce two matrices C and D by

$$C_{kl} := \int_{\Gamma^m} (\mathbf{w}_i^k \cdot \mathbf{t}_i)(\mathbf{w}_i^l \cdot \mathbf{t}_i)\, ds = \delta_{kl}\, |e_k|, \quad e_k, e_l \subset \Gamma^m,$$

$$D_{kn} := \int_{\Gamma^m} (\mathbf{w}_i^k \cdot \mathbf{t}_i)(\mathbf{w}_r^n \cdot \mathbf{t}_i)\, ds, \quad e_k \subset \Gamma^m, \quad e_n \subset \partial\Omega_r, \quad r \in \mathcal{J}_m,$$

where $\Gamma^m = \Gamma^{i,j}$. Then the matching conditions (5) have the form

$$Bu = 0, \qquad \text{where} \qquad B = C - D.$$

We remark that the entries of C and D depend on the particular choice of degrees of freedom defined in (4).

The matrix B can also be written as

$$B = \left[B^{(1)} \; B^{(2)} \; \cdots \; B^{(N)} \right],$$

where the local matrices $B^{(i)}$ act on vectors in W_i. The entries of B do not belong in general to $\{0, 1, -1\}$ as in the conforming case described in [33], and, since we are working with nonmatching grids, they take into account the edge intersections at the interfaces.

We conclude this section by recalling an a priori estimate of the approximation error for the mortar edge element method in two dimensions (see [3] for a proof).

Theorem 3.1. *Assume that the exact solution* $\mathbf{u}$ *of* (1) *is such that* $\mathbf{u}_i \in H^1(\Omega_i)^2$ *and* $\operatorname{curl} \mathbf{u}_i \in H^1(\Omega_i)$*, and that the data* $\mathbf{f}$ *is such that* $\mathbf{f}_i \in H^1(\Omega_i)^2$*. Then the following estimate holds:*

$$||\mathbf{u} - \mathbf{u}_h||_{*,\Omega} \le C \sum_{i=1}^{N} h_i (||\mathbf{u}_i||_{H^1(\Omega_i)^2} + ||\operatorname{curl} \mathbf{u}_i||_{H^1(\Omega_i)^2} + ||\mathbf{f}_i||_{H^1(\Omega_i)^2}),$$

where

$$||\mathbf{u} - \mathbf{u}_h||_{*,\Omega} = \left(\sum_{i=1}^{N} ||\mathbf{u} - \mathbf{u}_h||^2_{L^2(\Omega_i)^2} + ||\operatorname{curl}(\mathbf{u} - \mathbf{u}_h)||^2_{L^2(\Omega_i)} \right)^{\frac{1}{2}}.$$

4. A FETI method. In this section, we introduce a FETI method for the solution of the linear system arising from the mortar edge element approximation of problem (3).

We first assemble the local stiffness matrices, relative to the bilinear forms $a_{\Omega_i}(\cdot,\cdot)$, and the local load vectors. The degrees of freedom that are not on the interface Γ belong only to one substructure and can be eliminated in parallel by block Gaussian elimination. Let $f^{(i)}$ be the resulting right-hand sides, and let $S^{(i)}$ be the Schur complement matrices

$$S^{(i)}:\ W_i \longrightarrow W_i,$$

relative to the degrees of freedom on $\partial\Omega_i \setminus \partial\Omega$.

We recall that the local Schur complements satisfy the following property:

$$u^{(i)^t} S^{(i)} u^{(i)} = a_{\Omega_i}(\mathcal{H}_i \mathbf{u}_i, \mathcal{H}_i \mathbf{u}_i); \tag{6}$$

see, e.g., [28, 32].

Following [19, 33], we can then write our mortar problem as

$$\begin{aligned} &Su + B^t\lambda = f,\\ &Bu = 0, \end{aligned} \tag{7}$$

where

$$u := \begin{bmatrix} u^{(1)} \\ \vdots \\ u^{(N)} \end{bmatrix} \in W, \quad S := \mathrm{diag}\{\mathrm{S}^{(1)}, \dots, \mathrm{S}^{(\mathrm{N})}\}, \quad \mathrm{f} := \begin{bmatrix} f^{(1)} \\ \vdots \\ f^{(N)} \end{bmatrix}.$$

The vector λ is a Lagrange multiplier relative to the weak continuity constraint $Bu = 0$.

We remark that the $S^{(i)}$ are always invertible, and, consequently, there is no natural coarse space associated to the substructures; we are in a similar case as the one considered in [13]. We first find u from the first equation in (7) and substitute its value in the second equation. We obtain the system

$$F\lambda = d, \tag{8}$$

where

$$F := B\,S^{-1}\,B^t, \quad d := B\,S^{-1}\,f.$$

Following [19, 33, 29], we now define a preconditioner. Since we assume that the coefficients do not have any jump, we do not need to introduce a set of scaling matrices as is required for problems with coefficient jumps; see, e.g., [31, 32, 19, 33]. We introduce the matrices

$$R := \left[R^{(1)}\ R^{(2)}\ \cdots\ R^{(M)}\right], \quad G := Q\,B\,R, \tag{9}$$

where $R^{(i)}$ are vectors in W, related to the substructures $\{\Omega_i\}$, and Q is a suitable invertible matrix that we will specify in the next section. More precisely, we suppose that $R^{(i)}$ is obtained from a local vector $\mathbf{r}_i \in W_i$ on $\partial\Omega_i \setminus \partial\Omega$ by extending it by zero on the boundaries of the other substructures. We will make a particular choice of R for problem (3) in section 5 and specify the dimension M.

Following [13, 33], we define the projection

$$P := I - G(G^tFG)^{-1}G^tF$$

onto the complement of Range(G). This projection is orthogonal with respect to the scalar product induced by F. Following [19, 33], we next define the preconditioner

$$\widehat{M}^{-1} := (BB^t)^{-1}\,BSB^t\,(BB^t)^{-1}.$$

It can be easily seen that BB^t is invertible and is block-diagonal only if the partition $\mathcal{F}_H$ is geometrically conforming.

Now we consider a projected conjugate gradient method as in [13, 33].

```
1. Initialize
       $\lambda^0 = G(G^t F G)^{-1} G^t d$
       $q^0 = d - F\lambda^0$

2. Iterate $k = 1, 2, \cdots$ until convergence
       Project: $w^{k-1} = P^t q^{k-1}$
       Precondition: $z^{k-1} = \widehat{M}^{-1} w^{k-1}$
       Project: $y^{k-1} = P z^{k-1}$
       $\beta^k = \langle y^{k-1}, w^{k-1} \rangle / \langle y^{k-2}, w^{k-2} \rangle \quad [\beta^1 = 0]$
       $p^k = y^{k-1} + \beta^k p^{k-1} \quad [p^1 = y^0]$
       $\alpha^k = \langle y^{k-1}, w^{k-1} \rangle / \langle p^k, F p^k \rangle$
       $\lambda^k = \lambda^{k-1} + \alpha^k p^k$
       $q^k = q^{k-1} - \alpha^k F p^k$
```

The first projection can be omitted; because of the choice of the initial vector λ^0, we have $w^{k-1} = q^{k-1}$ after the first projection step. Here we have denoted the residual at the kth step by q^k. In practice, partial or full reorthogonalization may be required; cf. [16].

The method presented here is equivalent to using the conjugate gradient method for solving the preconditioned system

$$P\widehat{M}^{-1}P^t F\lambda = P\widehat{M}^{-1}P^t d, \quad \lambda \in \lambda_0 + V, \tag{10}$$

with

$$V := \text{Range}(P). \tag{11}$$

We remark that the matrices S and S^{-1} do not need to be calculated in practice. The action of S on a vector requires the solution of a Dirichlet problem on each substructure, while the action of S^{-1} requires the solution of a Neumann problem on each substructure; see [28, Ch. 4].

Remark 4.1. The extension of FETI-type preconditioners to the case of problems with jump coefficients on nonmatching grids appears to be hard, both for nodal and Nédélec finite elements. In the conforming case, suitable scaling diagonal matrices are employed. For each degree of freedom on the subdomain interface (a node for nodal elements or an edge for Nédélec elements), the corresponding entry is constructed with the values of the coefficients on the subdomains that share this degree of freedom; see, e.g., [19, 33]. This can certainly be generalized to the corresponding mortar methods built on *geometrically conforming partitions*. However, in the general case, an edge on the interface may belong only *partially* to a subdomain, and it is reasonable to assume that the scaling matrices should also take into account the relative size of the intersection of an edge with a subdomain boundary. Presently, it is not clear to the authors how exactly this can be accomplished, and this generalization is left for a future work.

5. A particular choice of the matrices R and Q. In this section, we consider a particular choice of the matrices R and Q in the definition of the FETI algorithm for problem (3).

We proceed in a similar way as in [33, sect. 5], but we will need to introduce a suitable matrix Q, different from the identity.

The definition of R is the same as in the conforming case (see [33, sect. 5]) and is given in terms of local vectors.

DEFINITION 5.1. *The local vectors $\{\mathbf{r}_i,\ i = 1, \dots, N\}$ with the corresponding vectors of degrees of freedom $\{r^{(i)}\}$ are the unique vectors that satisfy*

$$r^{(i)^t} v^{(i)} = \sum_{\substack{e_k \subset \partial\Omega_i \setminus \partial\Omega \\ e_k \in \mathcal{E}_{i,h}}} r_k^{(i)} v_k^{(i)} = \int_{\partial\Omega_i \setminus \partial\Omega} \mathbf{v}_i \cdot \mathbf{t}_i \, ds, \quad \mathbf{v}_i \in W_i.$$

The global vectors $R^{(i)}$ are obtained by extending the local vectors $r^{(i)}$ by zero outside $\partial\Omega_i$.

We can easily find that

$$r_k^{(i)} = |e_k| \, t_k^{(i)}, \quad e_k \subset \partial\Omega_i \setminus \partial\Omega.$$

The vectors $\mathbf{r}_i$ have then the same direction as the $\mathbf{t}_i$ and are scaled using the lengths of the edges of the triangulations $\mathcal{T}_{i,h}$.

We then define the matrix Q as

$$Q := (BB^t)^{-1}. \tag{12}$$

Remark 5.1. In the case of a conforming triangulation the matrix Q is a multiple of the identity; see [33]. For matching grids, we then obtain the same preconditioner as introduced in [33] for conforming approximations. Here our choice of Q does not require any additional calculation since $(BB^t)^{-1}$ is also needed for the application of the preconditioner $\widehat{M}^{-1}$. When Q is applied to a vector, this requires the solution of a linear system involving BB^t. The matrix BB^t is block-diagonal only in the case of conforming partitions. However, it can be shown that if the two meshes across the interface are not too different, it is strongly diagonally dominant. It is thus well conditioned, and few iterations of the conjugate gradient method are enough to obtain the solution.

The idea behind the choice of the matrix Q can be explained in the following way. In the conforming case, a proof of an upper bound for the condition number of the operator $P\widehat{M}^{-1}P^tF$ that is independent of the number of substructures is possible if certain tangential vectors have a mean value of zero on the boundary of each substructure. In particular, for the case with no jumps, it is necessary that

$$u = B^t(BB^t)^{-1}Bw$$

has mean value zero if $w \in W$ satisfies

$$\mu^t Bw = 0, \quad \mu \in \mathrm{Range}(G);$$

see [33, sect. 5]. One can show that this is not satisfied in our more general case but that this condition holds if a modified matrix G is employed,

$$\tilde{G} := QBR,$$

and our numerical results in section 6 confirm our choice of modified coarse space.

It remains to decide how many of the local vectors $R^{(i)}$ need to be considered in the definition of the matrix R. We introduce $\mathcal{G}_H$ as the dual graph of the partition $\mathcal{F}_H$. Thus $\mathcal{G}_H$ has a vertex for each substructure of $\mathcal{F}_H$, and there is an edge in $\mathcal{G}_H$ between two vertices if the intersection of the boundaries of the corresponding substructures has positive measure. As in [33], we define the matrix R by

$$R := \begin{cases} \left[R^{(1)}\; R^{(2)}\; \cdots\; R^{(N-1)}\right] & \text{if } \mathcal{G}_H \text{ is two-colorable,} \\ \left[R^{(1)}\; R^{(2)}\; \cdots\; R^{(N)}\right] & \text{otherwise .} \end{cases} \tag{13}$$

The following result can be proven using [33, Lem. 5.2 and Thm. 5.1].

LEMMA 5.1. *Let R be defined in* (13)*. Then the matrix G has full rank.*

Remark 5.2. An analogous FETI method can also be devised for problems involving the bilinear form

$$\int_\Omega (a \operatorname{div} \mathbf{u} \operatorname{div} \mathbf{v} + A\, \mathbf{u} \cdot \mathbf{v})\, d\mathbf{x}, \quad \mathbf{u}, \mathbf{v} \in H(\operatorname{div}; \Omega),$$

discretized with the lowest-order Raviart–Thomas spaces. Here, $H(\operatorname{div}; \Omega)$ is the space of vectors in $(L^2)^2$ with divergence in L^2. Since, in two dimensions, vectors in the Raviart–Thomas spaces can be obtained from those in the Nédélec spaces by a rotation of 90 degrees, the unit outward normal vectors $\mathbf{n}_i$ to the boundaries $\partial\Omega_i$, instead of the unit tangent vectors $\mathbf{t}_i$, have to be employed in the construction of the local functions $\mathbf{r}_i$. All the definitions in this paper remain valid in this case. For Raviart–Thomas discretizations in three dimensions, an analogous method can also be defined, and all our definitions remain valid.

6. Numerical results.

The purpose of this section is to show that, for problems without jumps, the FETI method proposed here performs similarly to the corresponding method for conforming approximations; see [33, sect. 6]. In particular, our method appears to be scalable, its condition number depends only on the number of degrees of freedom per subdomain, and it is quite insensitive to variations of the ratios of the coefficients.

In many iterative substructuring methods, an important role is played by the ratio H/h that measures the number of degrees of freedom per subdomain. In particular, the condition number of these methods grows only quadratically with the logarithm of H/h; see, e.g., [28]. This ratio is regarded as a local quantity and can vary greatly from one subdomain to another. In our numerical results, we always report the maximum value of this ratio taken over the subdomains.

We consider the domain $\Omega = (0, 1)^2$ and assume that the coefficient matrix A is diagonal and equal to

$$A = \begin{bmatrix} b & 0 \\ 0 & b \end{bmatrix}.$$

In our first set of results, we consider a family of geometrically conforming partitions of Ω, into $2^d \times 2^d$ substructures of equal size, with $d = 1, 2, 3, 4$. For a fixed

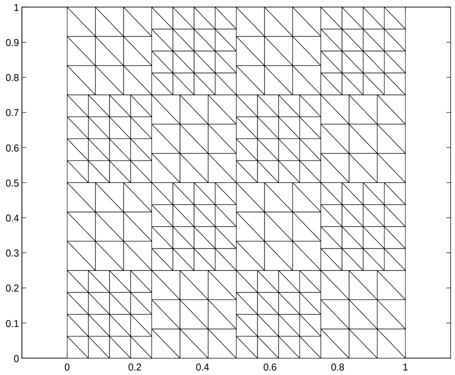


FIG. 3. *A conforming partition and a checkerboard-type discretization.*

TABLE 1

The FETI method for conforming partitions with $h_1/h_2 = 4/3$. Estimated condition number and number of CG iterations necessary to obtain a relative residual $\|q_k\|/\|f\|$ less than 10^{-6} (in parentheses), versus H/h and h. Case of $a = 1$, $b = 1$. The asterisks denote the cases for which we do not have enough memory to run the corresponding algorithm.

H/h	32	16	8	4
$1/h = 32$ (1600 el.)	-	1.805 (5)	2.941 (9)	2.179 (8)
$1/h = 64$ (6400 el.)	2.151 (6)	4.045 (11)	3.035 (10)	2.165 (7)
$1/h = 128$ (25600 el.)	5.314 (12)	4.175 (12)	3.013 (9)	*

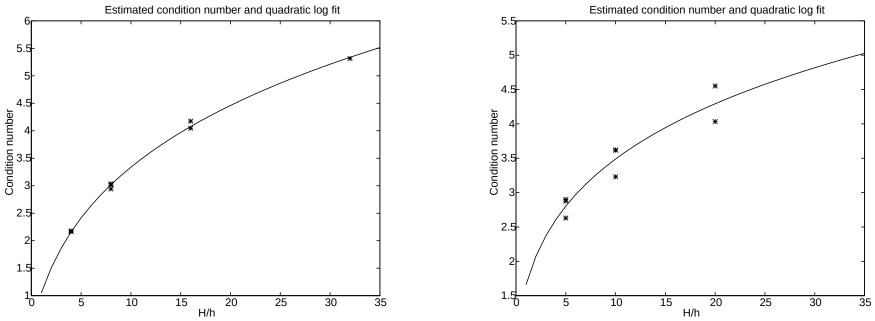


FIG. 4. *Case with $a = 1$, $b = 1$. Estimated condition numbers (asterisk) from Tables* 1 *(left) and* 3 *(right), and least-square second order logarithmic polynomial (solid line) versus $\rho = H/h$ for the FETI method for conforming (left) and nonconforming (right) partitions.*

partition, we consider two kinds of uniform triangulations for the substructures in such a way that on the interface between two adjacent substructures the meshes do not match. The ratio between the mesh-sizes of the two triangulations is $h_1/h_2 = 4/3$. Figure 3 shows an example of this checkerboard-type discretization for $d = 2$.

We first consider the case $a = 1$ and $b = 1$. In Table 1, we show the estimated condition number and the number of iterations to obtain a relative residual $\|q_k\|/\|f\|$ less than 10^{-6} as a function of the diameter of the finer mesh and the partition. Here q_k is the kth residual as defined in the algorithmic description given in section 4. For a fixed ratio H/h, the condition number and the number of iterations are quite insensitive to the dimension of the fine meshes. In addition, even for nonmatching

Table 2

The FETI method for conforming partitions with $h_1/h_2 = 4/3$. Estimated condition number and number of CG iterations necessary to obtain relative preconditioned residual ($\|q_k\|/\|f\|$) less than 10^{-6} (in parentheses), versus H/h and b. Case of $1/h = 128$ and $a = 1$.

H/h	8	16	32
b=0.0001	3.091 (16)	4.216 (20)	5.364 (18)
b=0.001	3.078 (14)	4.21 (18)	5.358 (16)
b=0.01	3.069 (13)	4.203 (16)	5.353 (15)
b= 0.1	3.044 (11)	4.192 (14)	5.346 (14)
b= 1	3.013 (9)	4.175 (12)	5.314 (12)
b= 10	2.992 (8)	4.114 (11)	5.154 (11)
b= 100	2.939 (9)	3.829 (11)	4.379 (11)
b= 1000	2.501 (7)	2.746 (8)	2.486 (7)
b=1e+04	1.418 (4)	1.493 (4)	1.533 (4)
b=1e+05	1.037 (2)	1.042 (2)	1.044 (2)
b=1e+06	1.06 (2)	1.046 (2)	1.044 (2)

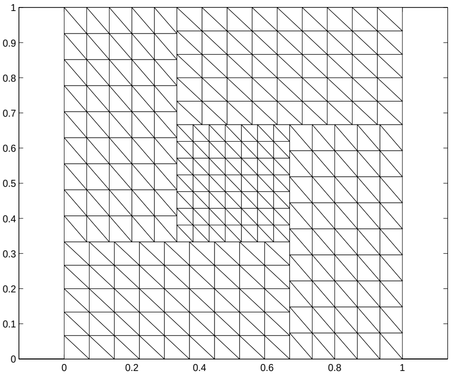


Fig. 5. *A block consisting of five subdomains, employed for building a nonconforming partition.*

grids, the ratio H/h appears to play an important role; see also Figure 4.

In Table 2, we show some results when the ratio of the coefficients b and a changes. For a fixed value of $1/h = 128$ and $a = 1$, and for the partitions into 2^d by 2^d substructures with $d = 2, 3, 4$, the estimated condition number and the number of iterations are shown as a function of H/h and b. The number of iterations and the condition number appear to be bounded independently of the ratio of the coefficients.

We then consider some test cases relative to geometrically nonconforming partitions of the domain $(0, 1)^2$. We consider partitions consisting of $2^d \times 2^d$ equal blocks, $d = 0, 1, 2, 3$. A block is made of five nonconforming subdomains and is shown in Figure 5 together with a possible triangulation. Figure 6 shows a partition for the case $d = 1$ (four blocks and twenty subdomains). The number of subdomains is five times the number of blocks. We then consider uniform triangulations for the subdomains in each block. The rectangular subdomains have the same mesh.

We first consider a case where the ratio between the mesh sizes of the rectangular and square subdomains is $h_1/h_2 = 7/5$; see Figures 5 and 6 for two examples. In Table 3, we show the estimated condition number and the number of iterations to obtain a relative residual $\|q_k\|/\|f\|$ less than 10^{-6} as a function of the diameter of the finer mesh and the ratio H/h. The condition number appears to increase slowly with H/h and to be quite insensitive to the size of the fine meshes.

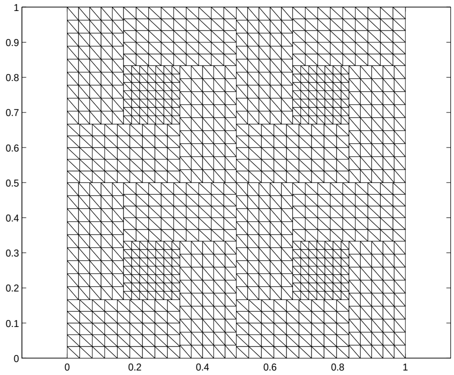


FIG. 6. *A nonconforming partition consisting of four blocks.*

TABLE 3

The FETI method for nonconforming partitions with $h_1/h_2 = 7/5$. Estimated condition number and number of CG iterations necessary to obtain a relative residual $\|q_k\|/\|f\|$ less than 10^{-6} (in parentheses), versus H/h and h. Case of $a = 1$, $b = 1$.

H/h	20	10	5
$1/h = 21$ (1832 el.)	-	2.728 (8)	2.63 (10)
$1/h = 42$ (7328 el.)	3.459 (8)	3.23 (11)	2.876 (10)
$1/h = 84$ (29312 el.)	4.034 (13)	3.619 (12)	2.901 (10)
$1/h = 168$ (117248 el.)	4.552 (14)	3.619 (12)	*

In Table 4, we show some results when the ratio of the coefficients b and a changes. For a fixed value of $1/h = 84$ (29312 elements) and $a = 1$, and for the partitions into 2^d by 2^d blocks with $d = 1, 2, 3$, the estimated condition number and the number of iterations are shown as a function of H/h and b.

For the same nonconforming partitions, we finally consider a case where the ratio between the diameters of the meshes of the rectangular and square subdomains is larger. We choose $h_1/h_2 = 2.8$. In Table 5, we show some results when the ratio of the coefficients b and a changes. For a fixed value of $1/h = 168$ (48128 elements) and $a = 1$, and for the partitions into 2^d by 2^d blocks with $d = 1, 2, 3$, the estimated condition number and the number of iterations are shown as a function of H/h and b. In this case, the meshes of adjacent substructures are fairly different, but the condition numbers and the number of iterations are still quite satisfactory.

To end this section, we give an upper bound for the condition number of the proposed method, both in the cases of conforming and nonconforming partitions. Figure 4 shows the estimated condition numbers (asterisk) from Tables 1 (left) and 3 (right) for $a = b = 1$ as a function of $\rho = H/h$ for different values of n. We have also plotted the best second-order logarithmic polynomial least-square fits. Our results for both conforming and nonconforming partitions are consistent with the bound for the condition number

$$\kappa(P\widehat{M}^{-1}P^tF) \leq C\left(1 + \log\frac{H}{h}\right)^2,$$

which was proven in [33, Thm. 5.2] for conforming approximations, and suggest that this bound is sharp.

Table 4

The FETI method for nonconforming partitions with $h_1/h_2 = 7/5$. *Estimated condition number and number of CG iterations necessary to obtain relative preconditioned residual* ($\|q_k\|/\|f\|$) *less than* 10^{-6} *(in parentheses), versus* H/h *and* b. *Case of* $1/h = 84$ *and* $a = 1$.

H/h	5	10	20
b=0.0001	2.966 (17)	3.667 (20)	4.068 (21)
b=0.001	2.955 (15)	3.663 (18)	4.064 (19)
b=0.01	2.951 (14)	3.658 (16)	4.06 (17
b= 0.1	2.933 (12)	3.651 (14)	4.054 (15)
b= 1	2.901 (10)	3.619 (12)	4.034 (13)
b= 10	2.882 (9)	3.561 (11)	3.93 (12)
b= 100	2.769 (9)	3.214 (10)	3.284 (10)
b= 1000	2.305 (7)	2.197 (7)	2.229 (7)
b=1e+04	1.656 (5)	1.523 (4)	1.54 (4)
b=1e+05	1.173 (3)	1.178 (3)	1.086 (2)
b=1e+06	1.135 (2)	1.115 (2)	1.089 (2)

Table 5

The FETI method for nonconforming partitions with $h_1/h_2 = 2.8$. *Estimated condition number and number of CG iterations necessary to obtain relative preconditioned residual* ($\|q_k\|/\|f\|$) *less than* 10^{-6} *(in parentheses), versus* H/h *and* b. *Case of* $1/h = 168$ *and* $a = 1$.

H/h	14	28	56
b=0.0001	5.058 (23)	5.062 (24)	6.275 (24)
b=0.001	5.054 (21)	5.056 (22)	6.27 (23)
b=0.01	5.045 (19)	5.043 (19)	6.258 (21)
b= 0.1	5.026 (16)	5.032 (17)	6.241 (18)
b= 1	4.994 (14)	5.008 (15)	6.205 (16)
b= 10	4.922 (13)	4.977 (14)	6.094 (15)
b= 100	4.833 (12)	4.761 (13)	5.48 (13)
b= 1000	4.448 (11)	3.938 (10)	4.078 (9)
b=1e+04	3.381 (8)	2.669 (6)	2.668 (6)
b=1e+05	2.257 (5)	1.901 (4)	1.906 (3)
b=1e+06	1.947 (3)	1.786 (3)	1.436 (2)

Acknowledgments. The authors are grateful to Olof Widlund and Yvon Maday for enlightening discussions of their work.

REFERENCES

[1] Y. Achdou, Y. A. Kuznetsov, and O. Pironneau, *Substructuring preconditioners for the* Q_1 *mortar element method*, Numer. Math., 71 (1995), pp. 419–449.

[2] Y. Achdou, Y. Maday, and O. B. Widlund, *Iterative substructuring preconditioners for mortar element methods in two dimensions*, SIAM J. Numer. Anal., 36 (1999), pp. 551–580.

[3] A. Ben Abdallah, F. Ben Belgacem, and Y. Maday, *Mortaring the Two-Dimensional "Nédélec" Finite Elements for the Discretization of the Maxwell Equations*, Technical report R99031, Laboratoire d'Analyse Numérique, Université Pierre et Marie Curie, Paris, France, 1999, M^3AS, submitted.

[4] F. Ben Belgacem, A. Buffa, and Y. Maday, *The Mortar Element Method for 3D Maxwell Equations: First Results*, Technical report, Laboratoire d'Analyse Numérique, Université Pierre et Marie Curie, Paris, France, 1999, SIAM J. Numer. Anal., to appear.

[5] O. Bíró, *Edge element formulations of eddy current problems*, Comput. Methods Appl. Mech. Engrg., 169 (1999), pp. 391–405.

[6] A. Bossavit, *Electromagnétisme en vue de la modélisation*, Math. Appl. 14, Springer-Verlag, Paris, France, 1993.

[7] D. Braess and W. Dahmen, *Stability estimates of the mortar finite element method for 3-dimensional problems*, East-West J. Numer. Math., 6 (1998), pp. 249–263.

[8] F. Brezzi and M. Fortin, *Mixed and Hybrid Finite Element Methods*, Springer-Verlag, New York, 1991.

[9] Y. Maday, C. Bernardi, and A. T. Patera, *A new nonconforming approach to domain decomposition: The mortar element method*, in Nonlinear Partial Differential Equations and Their Applications, H. Prezis and J. L. Lions, eds., Pitman, Boston, 1994, pp. 13–51.

[10] M. A. Casarin and O. B. Widlund, *A hierarchical preconditioner for the mortar finite element method*, Electron. Trans. Numer. Anal., 4 (1996), pp. 75–88.

[11] M. Dryja, *An additive Schwarz method for elliptic mortar finite element problems in three dimensions*, in Proceedings of the Ninth International Conference of Domain Decomposition Methods, P. E. Bjørstad, M. Espedal, and D. E. Keyes, eds., John Wiley and Sons, Strasbourg, France, 1997, pp. 88–96.

[12] B. Engelmann, R. H. W. Hoppe, Y. Iliash, Y. A. Kuznetsov, Yuri Vassilevski, and Barbara I. Wohlmuth, *Adaptive macro-hybrid finite element methods*, in Proceedings of the 2nd European Conference on Numerical Methods, H. Bock, F. Brezzi, R. Glowinski, G. Kanschat, Y. A. Kuznetsov, J. Périaux, and R. Rannacher, eds., World Scientific, Singapore, 1998, pp. 294–302.

[13] C. Farhat, P.-S. Chen, and J. Mandel, *A scalable Lagrange multiplier based domain decomposition method for time-dependent problems*, Internat. J. Numer. Methods Engrg., 38 (1995), pp. 3831–3853.

[14] C. Farhat, J. Mandel, and F.-X. Roux, *Optimal convergence properties of the FETI domain decomposition method*, Comput. Methods Appl. Mech. Engrg., 115 (1994), pp. 367–388.

[15] C. Farhat and F.-X. Roux, *A method of finite element tearing and interconnecting and its parallel solution algorithm*, Internat. J. Numer. Methods Engrg., 32 (1991), pp. 1205–1227.

[16] C. Farhat and F.-X. Roux, *Implicit parallel processing in structural mechanics*, in Computational Mechanics Advances, Vol. 2, J. Tinsley Oden, ed., North-Holland, Amsterdam, 1994, pp. 1–124.

[17] V. Girault and P.-A. Raviart, *Finite Element Methods for Navier-Stokes Equations*, Springer-Verlag, New York, 1986.

[18] A. Klawonn and D. Stefanica, *A Numerical Study of a Class of FETI Preconditioners for Mortar Finite Elements in Two Dimensions*, Technical report 773, Department of Computer Science, Courant Institute, New York University, New York, NY, 1998.

[19] A. Klawonn and O. B. Widlund, *FETI and Neumann–Neumann iterative substructuring methods: Connections and new results*, Comm. Pure Appl. Math., 54 (2001), pp. 57–90.

[20] Y. A. Kuznetsov, *Efficient iterative solvers for elliptic problems on nonmatching grids*, Russian J. Numer. Anal. Math. Modelling, 10 (1995), pp. 187–211.

[21] C. Lacour, *Iterative substructuring preconditioners for the mortar finite element method*, in Proceedings of the Ninth International Conference on Domain Decomposition Methods, P. E. Bjørstad, M. Espedal, and D. E. Keyes, eds., John Wiley and Sons, Strasbourg, France, 1997, pp. 406–412.

[22] P. Le Tallec and T. Sassi, *Domain decomposition with nonmatching grids: Augmented Lagrangian approach*, Math. Comp., 64 (1995), pp. 1367–1396.

[23] J. Mandel and R. Tezaur, *Convergence of a substructuring method with Lagrange multipliers*, Numer. Math., 73 (1996), pp. 473–487.

[24] J.-C. Nédélec, *Mixed finite elements in R^3*, Numer. Math., 35 (1980), pp. 315–341.

[25] F. Rapetti, *The mortar edge element method on non-matching grids for eddy current calculations in moving structures*, Internat. J. Numer. Modelling, to appear.

[26] F. Rapetti, *Approximation des équations de la magnétodynamique en domaine tournant par la méthode des éléments avec joints*, Ph.D. thesis, Université Pierre et Marie Curie, Paris, France, 2000.

[27] D. Rixen and C. Farhat, *A simple and efficient extension of a class of substructure based preconditioners to heterogeneous structural mechanics problems*, Internat. J. Numer. Methods Engrg., 44 (1999), pp. 489–516.

[28] B. F. Smith, P. E. Bjørstad, and W. D. Gropp, *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge University Press, Cambridge, UK, 1996.

[29] D. Stefanica, *Domain Decomposition Methods for Mortar Finite Elements*, Ph.D. thesis, Courant Institute of Mathematical Sciences, New York University, New York, NY, 1999.

[30] R. Tezaur, *Analysis of Lagrange Multiplier Based Domain Decomposition*, Ph.D. thesis, University of Colorado, Denver, CO, 1998.

[31] A. Toselli, *Domain Decomposition Methods for Vector Field Problems*, Ph.D. thesis, Courant Institute of Mathematical Sciences, New York University, New York, NY, 1999, Technical report 785, Department of Computer Science, Courant Institute of Mathematical Sciences, New York University, New York, NY, 1999.

[32] A. Toselli, *Neumann–Neumann methods for vector field problems*, Electron. Trans. Numer. Anal., 11 (2000), pp. 1–24.

[33] A. Toselli and A. Klawonn, *A FETI Domain Decomposition Method for Maxwell's Equations with Discontinuous Coefficients in Two Dimensions*, Technical report 788, Department of Computer Science, Courant Institute, New York University, New York, NY, 1999, SIAM J. Numer. Anal., to appear.

[34] B. I. Wohlmuth, *Discretization Methods and Iterative Solvers Based on Domain Decomposition*, Mathematisch–Naturwissenschaftliche Fakultät, Universität Augsburg, Habilitationsschrift, Augsburg, Germany, 1999.

SIAM J. Sci. Comput.
Vol. 23, No. 1, pp. 109–133

AMGe BASED ON ELEMENT AGGLOMERATION*

JIM E. JONES† AND PANAYOT S. VASSILEVSKI†

Abstract. This paper contains the main ideas for an AMGe (algebraic multigrid for finite elements) method based on element agglomeration. In the method, coarse grid elements are formed by agglomerating fine grid elements. Compatible interpolation operators are constructed which yield coarse grid basis functions with a minimal energy property. Heuristics based on interpolation quality measures are used to guide the agglomeration procedure. The performance of the resulting method is demonstrated in two-level numerical experiments.

Key words. algebraic multigrid, element agglomeration, algebraic coarsening, implementation

AMS subject classifications. 65F10, 65N20, 65N30

PII. S1064827599361047

1. Introduction.

The algebraic multigrid (AMG) [5], [6], [13], [14], was developed as a generalization of the standard geometric multigrid to problems that either had no grid or were posed on unstructured grids where standard geometric multigrid methods are difficult to apply. The standard AMG method works well for many problems; however, its performance on some finite element problems is unsatisfactory. The heuristics used in the standard AMG method are based on properties of M-matrices, and finite element discretizations can produce non-M-matrices. This deficiency in the standard AMG method led Brezina et al. [7] to develop the algebraic multigrid for finite elements (AMGe). This previous paper showed how to use multigrid convergence theory and the local stiffness matrices for the individual finite elements to produce interpolation operators superior to those produced by standard AMG. This current paper uses AMGe ideas to produce not only interpolation operators but coarse grids (and elements) as well. The coarse elements are based on agglomeration of fine elements. A key point is the construction of a local, compatible interpolation operator. The interpolation is local in the sense that degrees of freedom (dofs) in an agglomerate interpolate only from other dofs in the same agglomerate. The interpolation is compatible in that the interpolation to dofs shared by two or more agglomerates is uniquely defined. In this way, the coarse element matrices are variationally related to the assembled matrices in a given agglomerated element, and (due to the compatibility) the global coarse matrix is variationally obtained from the global fine grid matrix.

In the remainder of this introductory section, we outline the proposed agglomeration AMGe method. The goal is to solve a system

$$A\mathbf{u} = \mathbf{f},$$

where A is the positive definite matrix arising from a finite element discretization. In the agglomeration AMGe method, we assume that we have access to the individual

*Received by the editors September 14, 1999; accepted for publication (in revised form) October 22, 2000; published electronically June 5, 2001. This work was performed under the auspices of the U.S. Department of Energy by the University of California Lawrence Livermore National Laboratory under contract W-7405-Eng-48. The U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes. Copyright is owned by SIAM to the extent not limited by these rights.

http://www.siam.org/journals/sisc/23-1/36104.html

†Center for Applied Scientific Computing, UC Lawrence Livermore National Laboratory, Mail Stop L-560, 7000 East Avenue, Livermore, CA 94550 (jjones@llnl.gov, panayot@llnl.gov).

element matrices. Our goal is to produce the components needed for a two-level solver: a coarse grid, grid transfer operators, and the coarse grid operator. In order to apply the method recursively (i.e., multigrid as opposed to two-level), individual element matrices on the coarse level must be produced. These goals are outlined below.

- *Given information.*
  1. A list $\mathcal{D}_f = \{d\}$ of the fine grid dofs.
  2. A list $\mathcal{E}_f$ of fine grid elements $\{e\}$, where each element e, by definition, is a list of dofs, i.e., $e = \{d_1,\ d_2, \dots,\ d_{n_e}\}$. Typically, $\mathcal{E}_f$ provides an overlapping partition of the set $\mathcal{D}_f$.
  3. The element matrices A_e, i.e., a list of $n_e \times n_e$ real numbers associated with the dofs of $e = \{d_1,\ d_2, \dots,\ d_{n_e}\}$. Equivalently, one may say that a quadratic form $a_e(\mathbf{v},\ \mathbf{v}) = \mathbf{v}_e^T A_e \mathbf{v}_e$ is given, where $\mathbf{v}$ is a vector (or discrete function) defined on $\mathcal{D}_f$ restricted to e; i.e.,

$$\mathbf{v}_e = \mathbf{v}\big|_e = \begin{bmatrix} v(d_1) \\ v(d_2) \\ \vdots \\ v(d_{n_e}) \end{bmatrix}.$$

     Note that this will be the notation consistently used throughout this paper, namely, for any subset $\Omega \subset \mathcal{D}$ and a vector $\mathbf{v}$ defined on $\mathcal{D}$ we will denote by $\mathbf{v}_\Omega = \mathbf{v}|_\Omega$ the restriction of $\mathbf{v}$ to Ω. When it simplifies the notation, we will sometimes use superscripts instead of subscripts with the same meaning (restriction to subset).
- *Output coarse information.*
  1. A coarse set of dofs, $\mathcal{D}_c \subset \mathcal{D}_f$.
  2. A set of coarse elements $\mathcal{E}_c = \{E_c\}$, i.e., an overlapping partition of $\mathcal{D}_c$.
  3. The coarse element matrices A_{E_c} for each $E_c \in \mathcal{E}_c$.
  4. An interpolation mapping $P:\ \mathcal{D}_c \mapsto \mathcal{D}_f$ such that

$$P = \begin{bmatrix} P_c^f \\ I \end{bmatrix} \begin{matrix} \}\mathcal{D}_f \setminus \mathcal{D}_c \\ \}\mathcal{D}_c \end{matrix}.$$

To be specific, assume that our "algebraic" elements (i.e., a list of collections $\{e\}$ of dofs) come from a finite element triangulation of a three-dimensional (3D) domain and respective conforming finite element spaces with nodal dofs. To create the coarse information we propose the following steps.

- Create a set of agglomerated elements $\mathcal{E} = \{E\}$, where each $E = e_1 \cup e_2 \cup \cdots \cup e_{n_E}$, $e_i \in \mathcal{E}_f$, and E is a connected set. By connected we mean that for any two elements, $e_i, e_j \in E$, there exists a connecting path of elements also in E beginning with e_i and ending with e_j such that consecutive elements in the path have nonempty intersection. This is a result of the "topological" algorithm used in the agglomeration procedure (Algorithm 4.1). Note that each fine grid element e should belong to a unique agglomerated element.
- Define faces and vertices of the agglomerated elements as follows.
  1. Consider all intersections $E_i \cap E_j$ for all pairs of different agglomerated elements E_i and E_j. An intersection of this type is called a face if it is a maximal one, i.e., if it is not contained in any other intersection. This defines the set of faces $\mathcal{F} = \{F\}$. We will also assume that a list of boundary faces $\partial\mathcal{D}$ will be given, and we will append them to $\mathcal{E}_f$. A

formal definition of a boundary face is then simply a maximal set of the type $E \cap \partial\mathcal{D}$, i.e., it is not a proper subset of any other intersection set (either of type $E_i \cap E_j$ or of type $E_i \cap \partial\mathcal{D}$).

2. Finally, consider all faces $F \in \mathcal{F}$ as lists of dofs. For each dof d compute the intersection $\cap\{F :\ d \in F\}$. The minimal (nonempty) intersections define the set of vertices $\mathcal{V} = \{V\}$.

For true finite element applications the last set of vertices will be disjoint sets; each vertex may contain more than one dof. This is the case if the underlying problem is a finite element discretization of a system of PDEs, such as elasticity, for example. For 3D problems, one may refine the above algorithm to create edges of the agglomerated elements; edges are defined to be maximal intersections of faces. In order to keep the presentation simple, we will focus mostly on two-dimensional (2D) problems.

At any rate, the above "topological" information (faces and vertices of elements) is readily provided by most of the finite element grid generators. So one may assume that this information is given on the fine grid. If not, one can create it as explained above based on computing, for faces, the maximal intersection sets of the type $e_i \cap e_j$, $e_i \neq e_j$ or of the type $e_i \cap$ boundary surface.

In order to generate the same information on a coarse level, it can be advantageous to carry out the intersection sets algorithm by preserving the dimensionality (or topology) in the following sense. If E is an agglomerated element, one has the option to represent E either in terms of the dofs of the original elements or in terms of the faces of the original elements. If the agglomerated elements and the boundary surfaces $\partial\mathcal{D}$ are represented in terms of the faces of the original elements, then all nonempty intersections of the type $E_i \cap E_j$ or $E_i \cap \partial\mathcal{D}$ are maximal. This is the storage (agglomerated elements in terms of faces of elements) that we use in practice.

Definition 1.1 (coarse dofs). *Having computed the set of vertices, we define our (minimal) coarse set of dofs to be those dofs which are contained in a vertex of an agglomerated element:*

$$\mathcal{D}_c = \{d \in \mathcal{D}_f :\ \exists V \in \mathcal{V} \text{ with } d \in V\}.$$

Note that in practice, one may have to enrich the minimal (vertex) set of coarse dofs for better performance.

Figure 1 shows the coarse dofs for a 2D scalar problem. Note that for a scalar problem, vertex and degree of freedom are synonymous.

Definition 1.2 (coarse elements). *For each agglomerated element E, we define a coarse element E_c consisting of dofs contained in a vertex of E, i.e.,*

$$E_c = \mathcal{D}_c \cap E.$$

For each agglomerated element E (or, equivalently, for each coarse element E_c), we construct a local interpolation operator P_E. This operator maps a vector defined at coarse dofs in E_c to a vector defined at the fine dofs in E. We require the set of local interpolation operators be compatible in that if $d \in E_1 \cap E_2$, then $P_{E_1}\mathbf{v}_{E_1^c}(d) = P_{E_2}\mathbf{v}_{E_2^c}(d)$ for all vectors $\mathbf{v}$. In other words, compatibility means that at shared dofs, the interpolation rules for the agglomerates must agree. Compatibility implies the following restriction.

Requirement 1.1. For $d \in \mathcal{D}_f$, let $N(d) = \cap\{$all agglomerated elements $E(d)$ that contain $d\}$. Then the value $v(d)$ must be interpolated from the dofs at the vertices of $N(d)$. Note that we assume interpolation is the identity at the vertices.

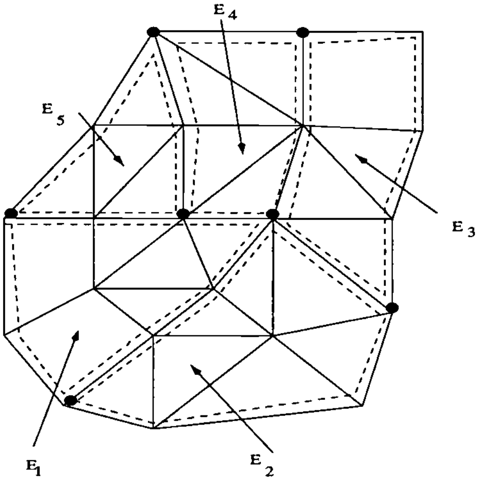


FIG. 1. *Triangulation of domain Ω into triangular and quadrilateral fine grid elements. Agglomerated elements $E_1, E_2, \dots, E_5$ and coarse dofs.*

DEFINITION 1.3 (interpolation mapping). *Having constructed a compatible set of local interpolation mappings $\{P_E\}$, define a global mapping $P : \mathcal{D}_c \mapsto \mathcal{D}$ by $P\mathbf{v}_c|_E \equiv P_E \mathbf{v}_{E_c}$. Compatibility implies that this uniquely defines P.*

DEFINITION 1.4 (coarse element matrices). *Assume that a compatible set of interpolation operators $\{P_E\}$ has been computed. Let A_E be the assembled matrix corresponding to the agglomerated element $E = e_1 \cup e_2 \cup \cdots \cup e_{n_E}$ defined by*

$$\mathbf{v}_E^T A_E \mathbf{w}_E \equiv \sum_{i=1}^{n_E} \mathbf{v}_{e_i}^T A_{e_i} \mathbf{w}_{e_i} \quad \text{for any } \mathbf{v}_E,\ \mathbf{w}_E. \tag{1.1}$$

Then, the coarse element matrix for the coarse element E_c is defined by

$$A_E^c \equiv P_E^T A_E P_E. \tag{1.2}$$

Note that the global coarse (stiffness) matrix A^c defined as

$$A^c = P^T A P$$

can be assembled from the coarse element matrices, i.e., that

$$\mathbf{v}_c^T A^c \mathbf{w}_c = \sum_{i=1}^{n_c} \mathbf{v}_{E_i^c}^T A_{E_i}^c \mathbf{w}_{E_i^c}.$$

Indeed, for $E_i = \bigcup_{j=1}^{n_{E_i}} e_i^j$,

$$\begin{aligned}
\sum_{i=1}^{n_c} \mathbf{v}_{E_i^c}^T A_{E_i}^c \mathbf{w}_{E_i^c} &= \sum_{i=1}^{n_c} (P_{E_i} \mathbf{v}_{E_i^c})^T A_{E_i} (P_{E_i} \mathbf{w}_{E_i^c}) \\
&= \sum_{i=1}^{n_c} (P\mathbf{v}_c|_{E_i})^T A_{E_i} (P\mathbf{w}_c|_{E_i}) \\
&= \sum_{i=1}^{n_c} \sum_{j=1}^{n_{E_i}} (P\mathbf{v}_c|_{e_i^j})^T A_{e_i^j} (P\mathbf{w}_c|_{e_i^j}) \\
&= \sum_{i=1}^{n_f} (P\mathbf{v}_c|_{e_i})^T A_{e_i} (P\mathbf{w}_c|_{e_i}) \\
&= \mathbf{v}_c^T P^T A P \mathbf{w}_c.
\end{aligned}$$

We should mention at this point that there are other approaches of constructing AMG methods that target non-M-matrices. One example is the aggregation based AMG of Vanek, Mandel, and Brezina [15]. In this method, one constructs aggregates (nonoverlapping partitions of the dofs) and forms a generally unstable (but simple) tentative prolongator. Finally, a smoothing step is applied in order to get a better quality interpolation. In Wan, Chan, and Smith [17], a direct approach of constructing coarse bases is proposed. The bases are selected by minimizing a quadratic energy functional while enforcing locality and a partition of unity property. In Mandel, Brezina, and Vanek [12], this approach was further developed by proposing fast algorithms for minimizing the quadratic functional. In Chan, Xu, and Zikatanov [9], the construction of the agglomerated elements is used a posteriori in the sense that one first selects a coarse grid (as a maximal independent set) and then agglomerated elements are constructed (based on the dual matrix graph). The agglomerates are subsequently divided into triangles, and the procedure can be recursively applied. The interpolation weights are computed based on averaging. In that sense, the present paper substantially differs from [9]. Our agglomeration algorithm is different (the coarse dofs are selected after the agglomeration is performed), and we assume more information. Namely, similar to the original AMGe paper [7], we require access to the individual elements and the respective element matrices on the fine grid. Note that this information is readily provided by most finite element grid generators. In contrast to [7] we are able to more systematically generate the input information (elements and their respective element matrices) on the coarse levels. This allows straightforward recursive use of the same two-level algorithm.

The remainder of the present paper is organized as follows. In section 2 we consider the construction of the local interpolation mappings based on a minimal energy principle. Section 3 deals with the energy minimization property of the coarse basis. In section 4, we specify an algorithm for agglomerating elements, which provides nicely matched agglomerated elements for structured triangular or quadrilateral meshes. We also discuss using measures of interpolation quality to guide the agglomeration procedure yielding semicoarsening for problems with anisotropy. In the final section, the performance of the resulting method is demonstrated in two-level numerical experiments.

2. The local interpolation mappings.

In this section we present an algorithm for generating the local interpolation mappings in a way that produces coarse grid basis functions with a quasi-minimal energy property. Most of the proofs in this section rely on basic properties of Schur complements of symmetric positive semidefinite matrices. A summary of these properties can be found, for example, in [1, section 3.2]. The problems that we target are second-order scalar elliptic problems without the low-order term as well as elasticity in two and three dimensions.

We begin by defining, for each fine grid dof d, the following sets:

- a neighborhood $\Omega(d) = \cup\{$all agglomerated elements $E(d)$ that contain $d\}$;
- a minimal set $N(d) = \cap\{$all agglomerated elements $E(d)$ that contain $d\}$.

Note that $N(d)$ can be a vertex, a face, or even an agglomerated element. From the definition of vertices, each $N(d)$ contains at least one vertex. Note also that there might be multiple copies of $N(d)$, i.e., $N(d_i) = N(d_j)$ for a $d_i \neq d_j$. We next introduce the following definition for the boundary of the sets $N(d)$.

Definition 2.1. *For any set $N(d)$ different than a face or an agglomerated element, define the boundary of $N(d)$, denoted $\partial N(d)$, to be the vertices contained in $N(d)$ (which is a nonempty set). If $N(d)$ is a face of an agglomerated element, define*

$\partial N(d)$ as the dofs in $N(d)$ that belong to more than one face. Finally, if $N(d)$ is an agglomerated element E, define the boundary, ∂E, as the union of all faces of E.

We now describe the construction of the local and compatible interpolation mappings. The set of interpolatory coarse dofs $d_1^c, \ldots, d_p^c$ that will be used to interpolate to d is constructed according to Requirement 1.1. That is, $d_1^c = d$ if d belongs to a vertex; otherwise, the interpolatory coarse dofs are the vertices of the set $N(d)$.

To define the interpolation weights for a dof d we use the following recursive procedure. The interpolation is the identity at the vertices. Then, for the set $N(d)$ assume that the interpolation at the dofs on $\partial N(d)$ has already been defined, i.e., $(P\mathbf{v}_c)|_{\partial N(d)}$ is well defined for $\mathbf{v}_c$ specified at the vertices of $N(d)$. Now extend the definition of $P\mathbf{v}_c$ on $N(d) \setminus (\partial N(d))$ by considering the neighborhood $\Omega(d)$ of all agglomerated elements that contain d. Let $A_{\Omega(d)}$ be the assembled matrix corresponding to all elements contained in that neighborhood. Consider the following two-by-two block structure of $A_{\Omega(d)}$, corresponding to the partitioning $(\Omega(d) \setminus \partial N(d)) \cup \partial N(d)$,

$$A_{\Omega(d)} = \begin{bmatrix} A_{ii} & A_{ib} \\ A_{bi} & A_{bb} \end{bmatrix} \begin{matrix} \}\Omega(d) \setminus \partial N(d), \\ \}\partial N(d). \end{matrix}$$

Here "i" stands for interior, and "b" stands for boundary dofs. Note that $\{d_1^c, \ldots, d_p^c\} \subset \partial N(d)$. The interpolation coefficients $w_{d,\, d_i^c}$, $i = 1, 2, \ldots, p$ are obtained by solving the following equation ($\mathbf{x}^c$ given):

$$A_{ii}\mathbf{x}^i + A_{ib}(P\mathbf{x}^c)_{\partial N(d)} = 0.$$

Then the equation corresponding to a dof d_f in $N(d) \setminus \partial N(d)$ gives

$$(\mathbf{x}^i)_{d_f} = \left(-A_{ii}^{-1}A_{ib}(P\mathbf{x}^c)_{\partial N(d)}\right)\big|_{d_f}.$$

That is, in particular for $d_f = d$, and $\mathbf{x}^c = \left[\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}\right] \begin{smallmatrix} \}\text{vertices of } N(d)\setminus\{d_i^c\}, \\ \}d_i^c \end{smallmatrix}$, one gets the interpolatory coefficient

$$w_{d,\, d_i^c} = \left(-\, A_{ii}^{-1}A_{ib} \left(P \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{matrix} \}\text{vertices of } N(d) \setminus \{d_i^c\} \\ \}d_i^c \end{matrix} \right)_{\partial N(d)} \right) \Bigg|_d .$$

This approach assumes that A_{ii} is invertible. As the following lemma shows, this is always the case for symmetric positive semidefinite matrices $A_{\Omega(d)}$ if the set of boundary dofs $\partial N(d)$ is sufficiently rich.

Lemma 2.2. *Given a set E, a union of fine elements, partition it into two groups: "f"-dofs denoted by $\mathcal{D}_{E,\, f}$ and "c"-dofs denoted $\mathcal{D}_{E,\, c}$. Let A_E be the assembled matrix corresponding to E partitioned as follows:*

$$A_E = \begin{bmatrix} A_{E,\, ff} & A_{E,\, fc} \\ A_{E,\, cf} & A_{E,\, cc} \end{bmatrix}.$$

If there exists a basis $\{\mathbf{d}_i\}$ for the null-space of the assembled, symmetric positive semidefinite matrix A_E, such that $\{\mathbf{d}_i\}$ restricted to $\mathcal{D}_{E,\, c}$ remain linearly independent, then $A_{E,\, ff}$ is invertible.

Proof. Assume that $A_{E,\, ff}\mathbf{x}^f = 0$. This implies that

$$\begin{bmatrix} \mathbf{x}^f \\ 0 \end{bmatrix}^T A_E \begin{bmatrix} \mathbf{x}^f \\ 0 \end{bmatrix} = 0,$$

and since A_E is positive semidefinite, this implies

$$A_E \begin{bmatrix} \mathbf{x}^f \\ 0 \end{bmatrix} = 0.$$

That is, $\left[\begin{smallmatrix} \mathbf{x}^f \\ 0 \end{smallmatrix}\right]$ is in the null-space of A_E. Therefore, we can expand it in terms of the basis of the null-space, i.e.,

$$\begin{bmatrix} \mathbf{x}^f \\ 0 \end{bmatrix} = \sum_i c_i \mathbf{d}_i.$$

The second block equation implies

$$0 = \sum_i c_i \mathbf{d}_i^c.$$

The assumption that $\{\mathbf{d}_i\}$ remains linearly independent when restricted to $\mathcal{D}_{E,\,c}$ means that $\{\mathbf{d}_i^c\}$ are linearly independent. Thus all $c_i = 0$ and $\mathbf{x}^f = 0$. That is, $A_{E,\,ff}\mathbf{x}^f = 0$ implies $\mathbf{x}^f = 0$; hence $A_{E,\,ff}$ is invertible. □

Remark 2.1. For the model case of second-order scalar elliptic equations, $\mathcal{L}u \equiv -\operatorname{div}(a\nabla u) = F$, a basis of the null-space of A_E is $\begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$, and its restriction onto the set of coarse dofs is again the constant vector; hence it is linearly independent. The above lemma shows that the corresponding $A_{E,\,ff}$ will be invertible.

Remark 2.2. If $\mathbf{x}$ is in the null-space of A_E, i.e.,

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}^f \\ \mathbf{x}^c \end{bmatrix} \text{ and } A_E\mathbf{x} = 0,$$

then

$$A_{E,\,ff}\mathbf{x}^f + A_{E,\,fc}\mathbf{x}^c = 0.$$

Thus the previously defined interpolation procedure is exact for vectors in the null-space of A_E.

In showing that the interpolation mappings produce coarse basis functions enjoying a certain energy minimization property, we rely on the following relationships between energy minimization and Schur complements.

Remark 2.3. Consider a matrix A with any two-by-two blocking

$$A = \begin{bmatrix} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{bmatrix}.$$

Assume A_{ff} is invertible, and define the Schur complement of A on c as $S_c \equiv A_{cc} - A_{cf}A_{ff}^{-1}A_{fc}$. If A is symmetric positive semidefinite, then

$$\mathbf{v}_c^T S_c \mathbf{v}_c = \inf_{\mathbf{v}|_c = \mathbf{v}_c} \mathbf{v}^T A \mathbf{v}. \tag{2.1}$$

In cases where A_{ff} is not invertible, (2.1) can be used to define the Schur complement. Note that if A is symmetric positive semidefinite, then so is S_c. Finally, one has the identity

$$A\mathbf{v} = \begin{bmatrix} 0 \\ S_c\mathbf{v}_c \end{bmatrix} \tag{2.2}$$

for any minimizer $\mathbf{v}$, i.e., for any vector $\mathbf{v}$ for which $\mathbf{v}_c^T S_c \mathbf{v}_c = \mathbf{v}^T A \mathbf{v}$ and $\mathbf{v}|_c = \mathbf{v}_c$. The following lemma is a straightforward consequence of Remark 2.3.

LEMMA 2.3. *Using the notation of the previous remark, assume A_{ff} is invertible, and let $\mathbf{v}_c$ be a null-vector of S_c; then $\mathbf{v}_c$ can be uniquely extended to the null-space of A.*

We are now ready to show several energy minimization properties of the local interpolation mappings P_E formulated for simplicity for 2D elements.

We first demonstrate an energy minimization property for dofs interior to an agglomerated element. Let d belong to a unique agglomerated element E. Thus the neighborhood $\Omega(d)$, used to define interpolation, consists of the fine-grid elements that are contained in E. Then, $P = P_E$ is constructed based on the following block-ordering of A_E:

$$A_E = \begin{bmatrix} A_{ii} & A_{ib} \\ A_{bi} & A_{bb} \end{bmatrix} \begin{matrix} \} & E \setminus \partial E, \\ \} & \partial E. \end{matrix}$$

The coefficients of P_E are obtained by solving the equation ($\mathbf{x}^c$ given)

$$A_{E,\, ii}\mathbf{x}^i + A_{E,\, ib}(P_E \mathbf{x}^c)_{\partial E} = 0.$$

It is equivalent then to say that $\mathbf{x}^i = -A_{ii}^{-1} A_{ib}(P_E \mathbf{x}^c)$ solves the minimization problem

$$\min_{\mathbf{x}:\ \mathbf{x}|_{\partial E} = (P_E \mathbf{x}^c)_{\partial E}} \mathbf{x}^T A_E \mathbf{x}. \tag{2.3}$$

By definition, $P_E \mathbf{x}_c|_d = -\left. A_{ii}^{-1} A_{ib}(P_E \mathbf{x}^c)_{\partial E}\right|_d$ for all $d \in E$ that do not belong to a face of E.

We next show an energy minimization property for dofs on faces; this is used later to show a global energy minimization property of the coarse grid basis functions. For every face F, the neighborhood used to define interpolation is $E_F^+ \cup E_F^-$, where E_F^+ and E_F^- are the two neighboring agglomerated elements that form the face F (one of them can be $\emptyset$ if F is a boundary face).

LEMMA 2.4. *For every face $F = E_F^+ \cap E_F^-$, the interpolation P minimizes the quadratic form $(\mathbf{w}_F)^T(S_{E_F^+,\, F} + S_{E_F^-,\, F})\mathbf{w}_F$ for $\mathbf{w}_F$ fixed at the vertices of F, where $S_{E,\, F}$ denotes the Schur complement of A_E on F.*

Proof. Denote $E_1 = E_F^-$ and $E_2 = E_F^+$. Each dof on F which is not a vertex is interpolated from the vertices of F based on the assembled matrix $A_{E_1 \cup E_2}$ corresponding to the domain $E_1 \cup E_2$. To define P on F, one looks at the matrix

$$A_{E_1 \cup E_2} = \begin{bmatrix} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{bmatrix} \begin{matrix} \}E_1 \cup E_2 \setminus (\text{ vertices of } F), \\ \}(\text{ vertices of } F). \end{matrix}$$

Then $(P\mathbf{v}^c)(d_f) = (-A_{ff}^{-1} A_{fc}\mathbf{v}^c)(d_f)$ for any $d_f \in F \setminus (\text{ vertices of } F)$. Equivalently, from the equations that define P on F,

$$A_{ff}\mathbf{w}^f + A_{fc}\mathbf{w}^c = 0,$$

one can eliminate the dofs that are on $E_1 \cup E_2 \setminus F$, thus ending up with the Schur complement problem

$$S_{E_1 \cup E_2,\, F}\mathbf{w}^F|_{F \setminus (\text{ vertices of } F)} = 0, \quad \mathbf{w}^F = \begin{bmatrix} \mathbf{w}^f \\ \mathbf{w}^c \end{bmatrix} \begin{matrix} \}F \setminus (\text{ vertices of } F), \\ \}\text{ vertices of } F. \end{matrix} \tag{2.4}$$

Since F is a separator for $E_1 \cup E_2$, one has that $S_{E_1 \cup E_2,\, F} = S_{E_1,\, F} + S_{E_2,\, F}$. Since $S_{E_1 \cup E_2,\, F}$ is symmetric semidefinite, (2.4) is equivalent to the following minimization problem:

$$\inf_{\mathbf{w}^F|_{\text{vertices of } F} = \mathbf{w}^c} (\mathbf{w}^F)^T S_{E_1 \cup E_2,\, F} \mathbf{w}^F.$$

By definition $\mathbf{w}^F = P\mathbf{w}^c$ solves (2.4) and thus has this equivalent minimization property. □

Throughout the remainder of the paper we will assume the following relations between the null-spaces of the assembled matrices A_{E_-} and A_{E_+} for any two neighboring agglomerated elements E_- and E_+ that share a common face F.

Assumption 2.1. For any $\mathbf{x}_{E_-}$ such that $A_{E_-}\mathbf{x}_{E_-} = 0$ there is an extension $\mathbf{x}$ of $\mathbf{x}_{E_-}$ defined on $E_- \cup E_+$ such that $A_{E_- \cup E_+}\mathbf{x} = 0$ and $\mathbf{x}|_{E_-} = \mathbf{x}_{E_-}$. Equivalently, $A_{E_+}\mathbf{x}_{E_+} = 0$ and $\mathbf{x}|_F = \mathbf{x}_{E_-}|_F$.

As a corollary of the above assumption, the respective Schur complements $S_{E_-;\, F}$ and $S_{E_+;\, F}$ of A_{E_-} and A_{E_+} on the face F are spectrally equivalent or, equivalently, have the same null-space.

Actually, the following local estimates hold.

Lemma 2.5. *Assume, in addition to Assumption* 2.1*, that every null-vector* $\mathbf{v}$ *of* A_E *restricted to a face* F *of* E *is uniquely determined from its vertex values* $\mathbf{v}_c$ *on* F*. Note that this is always the case if the set of coarse dofs on any* F *is sufficiently rich (see Lemma* 2.2*). If we have determined* $\mathbf{x} = P_E\mathbf{x}_c$ *first on* ∂E *and then in the interior of* E *as specified above, the local quadratic forms*

$$(P_E\mathbf{x}_c)^T A_E P_E \mathbf{x}_c, \quad \inf_{\mathbf{x}:\ \mathbf{x}|_{\mathcal{D}_c} = \mathbf{x}_c} \mathbf{x}^T A_E \mathbf{x}$$

are spectrally equivalent. That is, there exists a constant η_E *such that*

$$\inf_{\mathbf{x}:\ \mathbf{x}|_{\mathcal{D}_c} = \mathbf{x}_c} \mathbf{x}^T A_E \mathbf{x} \le (P_E\mathbf{x}_c)^T A_E P_E \mathbf{x}_c \le \eta_E \inf_{\mathbf{x}:\ \mathbf{x}|_{\mathcal{D}_c} = \mathbf{x}_c} \mathbf{x}^T A_E \mathbf{x}.$$

In other words, the coarse element matrix $A^c_{E_c}$ *and the Schur complement* S_c *of* A_E *on* $\mathcal{D}_c \cap E$ *are spectrally equivalent.*

Proof. To prove the result it is sufficient to show that both matrices have the same null-space. Assume now that $S_c\mathbf{x}_c = 0$. For any face F of E one can compute the Schur complement of S_c on F denoted by $S_{c,\, F}$. It is clear then (see (2.2)) that

$$S_{c,\, F}\mathbf{x}_{c,\, F} = 0. \tag{2.5}$$

Our goal is to show that $(P_E)^T A_E P_E \mathbf{x}_c = 0$, which is equivalent to $A_E(P_E\mathbf{x}_c) = 0$. By construction, one has $A_E(P_E\mathbf{x}_c) = 0$ in the interior of E. Also, from the definition of P_E for dofs on faces F (see (2.4)) one has

$$\left(S_{E,F} + S_{E_+,F}\right)(P_E\mathbf{x}_c)_F|_{F \setminus \text{vertices of } F} = 0.$$

Here, E_+ is the neighboring element to E which shares a common face F with E. From Assumption 2.1 it follows that $S_{E,F} + S_{E_+,F}$ and $S_{E,F}$ have the same null-space. Therefore, their respective Schur complements on the vertices of F ($F \cap \mathcal{D}_c$), $\sigma_{c,\, F}$ and $S_{c,\, F}$ will have the same null-space. Then (2.5) implies that $\sigma_{c,\, F}\mathbf{x}_{c,\, F} = 0$. Applying identity (2.2) (based on Lemma 2.4) yields

$$\left(S_{E,F} + S_{E_+,F}\right)(P_E\mathbf{x}_c)_F = \begin{bmatrix} 0 \\ \sigma_{c,\, F}\mathbf{x}_{c,\, F} \end{bmatrix} \begin{matrix} \} & F \setminus \text{vertices of } F, \\ \} & \text{vertices of } F, \end{matrix}$$

from which it follows that

$$\left(S_{E,F} + S_{E_+,F}\right)\left(P_E \mathbf{x}_c\right)_F = 0 \text{ on } F.$$

Again, the fact that $S_{E,F} + S_{E_+,F}$ and $S_{E,F}$ have the same null-space implies that

$$S_{E,F}\left(P_E \mathbf{x}_c\right)_F = 0 \ \text{ on } F.$$

This shows that $\left(P_E \mathbf{x}_c\right)_F$ is a restriction of a null-vector of A_E on F. Assumption 2.1 and the additional assumption we have made that every vector in the null-space of A_E restricted to a face is uniquely determined by its vertex values on that face then imply that $\left(P_E \mathbf{x}_c\right)_{\partial E}$ is the restriction of a null-vector of A_E on ∂E. This together with the fact that $A_E\left(P_E \mathbf{x}_c\right) = 0$ in the interior of E finally show that

$$A_E\left(P_E \mathbf{x}_c\right) = 0 \ \text{ on } E.$$

This completes the proof that $P_E \mathbf{x}_c$ is in the null-space of A_E, i.e., that $\mathbf{x}_c$ is in the null-space of $A^c_{E_c}$. The converse is also true. Namely, $A^c_{E_c}\mathbf{x}_c = 0$ implies that $(P_E \mathbf{x}_c)^T A_E P_E \mathbf{x}_c = 0$, and since A_E is symmetric positive semidefinite, one gets that $A_E P_E \mathbf{x}_c = 0$ or that $P_E \mathbf{x}_c$ belongs to the null-space of A_E. Therefore, $\mathbf{x}_c = P_E \mathbf{x}_c|_{\text{vertices of } E}$ belongs to the null-space of the Schur complement S_c of A_E. □

We then have the following global estimate by summing up the local estimates over the individual agglomerated elements.

THEOREM 2.6. *The compatible local interpolation mapping $P = P_E$ is approximately harmonic in the sense that its norm in the energy inner product is bounded, i.e.,*

$$\begin{aligned}\mathbf{v}_c^T A_c \mathbf{v}_c &= (P\mathbf{v}_c)^T A (P\mathbf{v}_c)\\ &\leq \sum_E \eta_E \inf_{\mathbf{v}_E|_{\mathcal{D}_c \cap E} = \mathbf{v}_c} \mathbf{v}_E^T A_E \mathbf{v}_E\\ &\leq \eta \inf_{\mathbf{v}|_{\mathcal{D}_c} = \mathbf{v}_c} \mathbf{v}^T A \mathbf{v}.\end{aligned}$$

The exact harmonic mapping corresponds to the best constant $\eta = 1$. As shown in Lemma 2.5, *$\eta = \max_{E\in\mathcal{E}} \eta_E$, and thus the individual η_E can be estimated locally. With this result, a classical two-level Gauss–Seidel iteration (see, e.g., Bank and Dupont* [3] *or Bank* [2]*) will have a convergence factor bounded by $\gamma^2 = 1 - \frac{1}{\eta}$.*

Remark 2.4. Note that the proof of Theorem 2.6 does not require uniqueness of the minimizers (hence of P). Note, however, that we assumed uniqueness on the faces (see Lemma 2.5). Hence it applies to element matrices coming from 2D and 3D elasticity. If one assumes a little more (see Assumption 2.2) the uniqueness of P (or of the minimizers) is guaranteed. Namely, one may assume the following.

Assumption 2.2. If d_c is a dof at a vertex and E is an agglomerated element containing that vertex, the only vector in the null-space of A_E and vanishing at d_c is the zero vector.

For the model case of 2D and 3D second-order scalar elliptic equations (of the form $\mathcal{L}u \equiv -\operatorname{div} a\nabla u = f$), this assumption holds. However, it may not hold for systems of PDEs. (It is not true for elasticity problems, for example.) If Assumption 2.2 holds, P_E is defined uniquely at the interior of $N(d)$ (edge, face, or agglomerated element E) based on a Schur complement of $A_{\Omega(d)}$ (to $N(d)$) by harmonically extending the values from the boundary of $N(d)$ into its interior. In particular, one has (see (2.3)) that for each E (2.6) holds, $\mathbf{w}_F = P_E \mathbf{w}_c|_F$, for any face (or edge) $F \subset E$:

$$\mathbf{w}_c^T A_E^c \mathbf{w}_c = \inf_{\mathbf{v}_E|_F = \mathbf{w}_F,\text{ for all } F\subset E} \mathbf{v}_E^T A_E \mathbf{v}_E. \tag{2.6}$$

Remark 2.5. The constants η_E in Lemma 2.5 are computable and can be used as local measures for interpolation quality in the sense that smaller η_E implies better interpolation. Theorem 2.6 shows that the local measures imply the approximate harmonic property of P. More details on how to compute measures of interpolation quality and their relation with other local constants are found in section 4.

3. Energy minimization properties of coarse basis functions. With the local interpolation operators defined, one can construct a coarse grid basis function v_d for each $d \in \mathcal{D}_c$ as follows. Define the coarse grid vector $\mathbf{v}_d^c$ that is one at d and zero elsewhere, and define $\mathbf{v}_d$ as this vector interpolated to the fine grid (i.e., $\mathbf{v}_d = P\mathbf{v}_d^c$). It is clear then that it will be zero outside the neighborhood $\Omega(d) = \cup_{i=1}^{p} E_i$ of the given dof d. In this way, $\mathbf{v}_d$ can be viewed as a basis vector (function) of the interpolated coarse space. Using finite element terminology, one may also say that $\mathbf{v}_d$ is a fine grid vector representation of a coarse-grid basis function.

Lemma 3.1. *For the model problem of finite element matrices (before imposing Dirichlet boundary conditions) coming from second-order scalar elliptic problems (*2D *or* 3D*), the* $\{\mathbf{v}_d\}$ *provide the partition of unity, i.e.,*

$$\sum_{d \in \mathcal{D}_c} \mathbf{v}_d = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}. \tag{3.1}$$

Proof. In the case of finite element matrices coming from 2D (or 3D) second-order scalar elliptic problems, constant vectors are in the null-space of the element matrices. By Remark 2.2, if $\mathbf{v}_c = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \in \mathcal{R}^{n_c}$, then $\mathbf{v} = P\mathbf{v}_c = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \in \mathcal{R}^n$. This holds since $\mathbf{v}_E = P_E \mathbf{v}_{c,\, E_c}$ for each coarse element E_c (or agglomerated element E). This, in particular, implies that $\sum_{d \in \mathcal{D}_c} \mathbf{v}_d = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \in \mathcal{R}^n$. □

Corollary 3.2. *Consider the model case of finite element matrices (before imposing Dirichlet boundary conditions) coming from second-order scalar elliptic problems (*2D *or* 3D*) on quasi-uniform triangulation. Let* $\{\mathbf{v}_d\}$ *be the set of basis functions generated by the local interpolation operators. Let* $\{\mathbf{w}_d\}$ *be any other potential set of local basis functions, i.e., a basis function exists for each* $d \in \mathcal{D}_c$ *with* $\mathbf{w}_d(d) = 1$ *and* $\mathbf{w}_d = 0$ *outside of the neighborhood* $\Omega(d)$*. Then the following energy minimization property of* $\{\mathbf{v}_d\}$ *holds:*

$$\sum_{d \in \mathcal{D}_c} \mathbf{v}_d^T A \mathbf{v}_d \le C \sum_{d \in \mathcal{D}_c} \inf_{\mathbf{w}_d} \mathbf{w}_d^T A \mathbf{w}_d. \tag{3.2}$$

Proof. Applying the approximate harmonic property of P_E for each agglomerated element E (Lemma 2.5), one ends up with the estimate

$$(\mathbf{v}_d|_E)^T A_E (\mathbf{v}_d|_E) \le \eta_E \inf_{\mathbf{w}_E:\ \mathbf{w}|_{\text{vertices of } E} = \mathbf{v}_d|_{\text{vertices of } E}} \mathbf{w}_E^T A_E \mathbf{w}_E.$$

Summing up over the agglomerated elements $E : \ E \subset \Omega(d)$, where $\Omega(d)$ is the union of all agglomerated elements that contain the vertex d (note that $\mathbf{v}_d$ is zero outside $\Omega(d)$), one ends up with the global estimate

$$\mathbf{v}_d^T A \mathbf{v}_d \le \eta \inf_{\mathbf{w}_d:\ \mathbf{w}_d|_{\mathcal{D}_c} = \mathbf{v}_d|_{\mathcal{D}_c}} \mathbf{w}_d^T A_{\Omega(d)} \mathbf{w}_d, \quad \eta = \max_E \eta_E.$$

Note that $\mathbf{w}_d = 1$ at the vertex d and is zero at the remaining vertices, and it is also zero outside $\Omega(d)$, i.e., it is locally supported.

Finally, summing over all $d \in \mathcal{D}_c$, one ends up with the desired estimate

$$\begin{aligned}\sum_{d\in\mathcal{D}_c} \mathbf{v}_d^T A \mathbf{v}_d &\leq C \sum_{d\in\mathcal{D}_c} \inf_{\mathbf{w}_d:\ \mathbf{w}_d|_{\mathcal{D}_c} = \mathbf{v}_d|_{\mathcal{D}_c}} \mathbf{w}_d^T A_{\Omega(d)} \mathbf{w}_d \\ &= C \sum_{d\in\mathcal{D}_c} \inf_{\mathbf{w}_d:\ \mathbf{w}_d|_{\mathcal{D}_c} = \mathbf{v}_d|_{\mathcal{D}_c}} \mathbf{w}_d^T A \mathbf{w}_d. \qquad \square\end{aligned}$$

Remark 3.1. Theorem 3.2 shows, for the model case of finite element matrices coming from second-order scalar elliptic equations as well as in the elasticity, that the coarse basis functions corresponding to the coefficient vectors $\mathbf{v}_d$ solve the energy minimization functional as defined in Wan, Chan, and Smith [17] up to a multiplicative constant. Fast algorithms to solve the problem of the energy minimization functional are proposed and analyzed in Mandel, Brezina, and Vanek [12].

Remark 3.2. For finite element matrices coming from 2D and 3D second-order scalar elliptic problems on quasi-uniform triangulation, the coarse space produced by the above algorithm also admits a weak approximation property (or, equivalently, provides partition of unity—see Lemma 3.1 and also estimate (4.2)) since the element matrices contain the constants in their null-space. Therefore, the constant is exactly interpolated from the vertices of the agglomerated elements as the same constant on the rest of the agglomerated element. That is, with the above minimization property, the AMGe method can actually become an optimal- (or almost optimal) order MG method if one can control the local constants η_E from Lemma 2.5 which depend on the way we agglomerate the elements at every coarsening step. If η gets large, a potential remedy might be the algebraic multilevel iteration (AMLI) stabilization procedure (cf. Vassilevski [16]) which is like the W-cycle or even more cycles. Approaches to rigorously study the convergence of the underlined AMG method can draw on the existing analytical tools for geometric MG convergence theory for finite element problems (see, e.g., the book by Bramble [4]). In the present paper we do not deal with multilevel convergence results.

Remark 3.3. One can actually apply the same interpolation procedure on agglomerated elements using it recursively to fine-grid element matrices coming from a nonsymmetric elliptic operator like convection-diffusion, e.g., $\mathcal{L}u \equiv -\operatorname{div}(\epsilon\nabla u) + \underline{b}\cdot\nabla u$. In Figures 2 and 3 a coarse basis function is shown (face and rotated) using four levels of coarsening procedure for a constant convection field $b_1 = 1$, $b_2 = -0.5$, and $\epsilon = 0.1$. Note also that in this case of the convection-diffusion operator the basis functions computed on the coarse levels by the proposed AMGe method will provide a partition of unity (as in the symmetric operator case), and hence the coarse spaces will admit a certain weak approximation property. The same applies for the so-called streamline diffusion operator $\mathcal{L}_\delta u \equiv -\operatorname{div}((\epsilon + \delta\underline{b}\,\underline{b}^T)\nabla u) + \underline{b}\cdot\nabla u$, where δ is a mesh-dependent parameter.

Remark 3.4. We finally remark that the presented AMGe method can be used in the so-called “homogenization” procedures to generate averaged coarse problems from problems on computationally unfeasible highly refined meshes and possibly with oscillatory coefficients (cf., e.g., [11] and references therein; see also [10]). The difference that we see here is that our coarsening procedure is local. We require the solution of small local problems (involving a few elements) rather than large subdomain solves in order to compute the effective coarse grid basis functions (or coarse-grid element matrices).

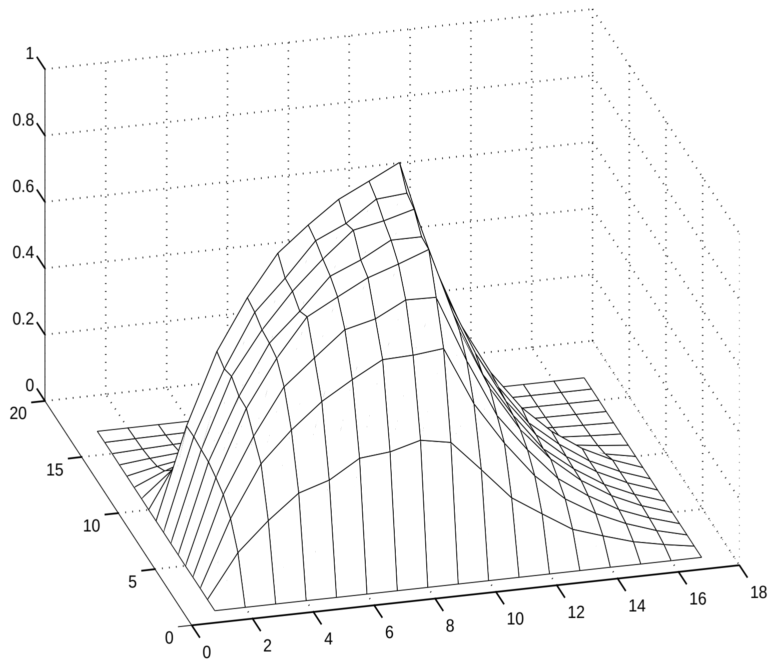


FIG. 2. *AMGe constructed "minimum energy" coarse basis function for convection-diffusion operator.*

4. Algorithms for element agglomeration. This section introduces the algorithm we have used in selecting the coarse grid agglomerates. The algorithm relies of the faces and edges of the original elements $\{e\}$; to simplify the discussion, we will focus mainly on 2D elements (i.e., having faces and vertices only). The method is based on the face-face graph of the fine grid elements (i.e., face f_1 and f_2 are neighbors if they share a common vertex) and uses an integer weight $w(f)$ for each face f. The eliminated faces f will have a weight $w(f) = -1$.

ALGORITHM 4.1 (element agglomeration based on the face-face graph).

- **initiate**. *Set* $w(f) = 0$ *for all faces* f*;*
- **global search**. *Find a face* f *with maximal* $w(f)$*; set* $E = \emptyset$*;*
  1. *Set* $E = E \cup e_1 \cup e_2$*, where* $e_1 \cap e_2 = f$*, and set* $w_{\max} = w(f)$*,* $w(f) = -1$*;*
  2. *Increment* $w(f_1) = w(f_1) + 1$ *for all faces* f_1 *such that* $w(f_1) \neq -1$ *and* f_1 *is a neighbor of* f*;*
  3. *Increment* $w(f_2) = w(f_2) + 1$ *for all faces* f_2 *such that* $w(f_2) \neq -1$*,* f_2 *is a neighbor of* f*, and* f_2 *and* f *are faces of a common element;*
  4. *From the neighbors of* f*, choose a face* g *with a maximal* $w(g)$*; if* $w(g) \geq w_{\max}$*, set* $f = g$*, and go to step* (1)*;*
  5. *If all neighbors of* f *have smaller weight than* $w_{\max}$*, the agglomerated element* E *is complete; set* $w(g) = -1$ *for all faces of the elements* e *contained in* E*; go to step* **global search***;*

This algorithm tends to produce nicely matched agglomerated elements and produces standard multigrid coarsening (up to boundary effects) for structured grid problems using linear or bilinear elements. See Figures 4 and 5 for the results of this procedure applied to a uniform triangular mesh after one and two agglomeration

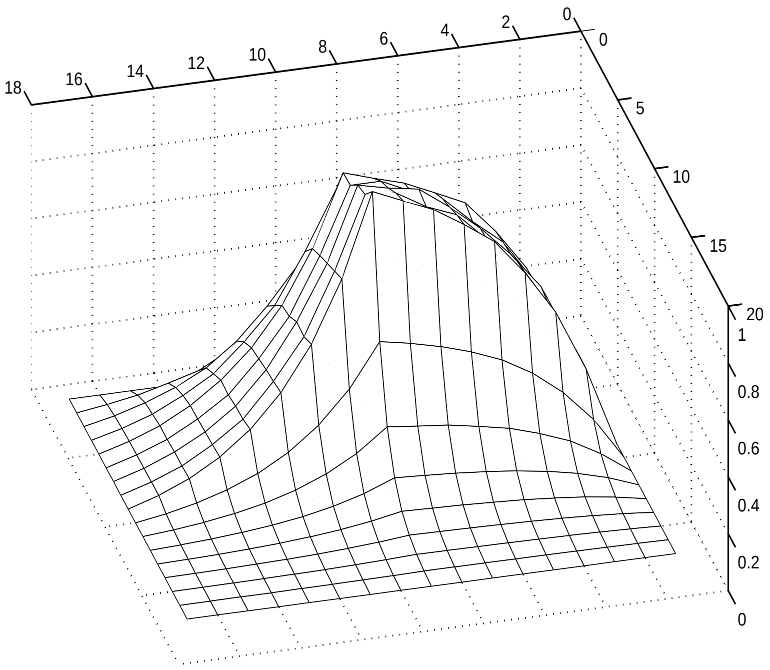


FIG. 3. *AMGe constructed "minimum energy" coarse basis function for convection-diffusion operator, rotated.*

steps, respectively. The setup cost of the algorithm is linear, i.e., proportional to the total number of dofs. The algorithm is easily implemented using, for example, double linked lists.

Figures 6 and 7 show the results of the algorithm for several unstructured problems. Figures 8, 9, 10, and 11, show fine unstructured grids using triangular elements, the agglomerated elements are shown in Figures 12, 13, 14, and 15 respectively. The latter are the actual grids on which the first set of numerical tests was performed.

In three dimensions one has the opportunity to introduce edges. Then one may construct more refined agglomeration algorithms that exploit this additional topological information, namely, the edge-edge and edge-face graphs. This information, however, has not been utilized in the present paper.

It is important to note that the above algorithm does not take into account any matrix entries while agglomerating the elements. For structured grid problems with anisotropy, it will produce full-coarsening. To produce semicoarsening for such problems, one can introduce barriers. This can be implemented by assigning to each face another (binary) weight $a(f) = \begin{cases} 0, \text{ acceptable,} \\ 1, \text{ unacceptable.} \end{cases}$ To prevent agglomeration through a face f, one can simply set $a(f) = 1$, and then in step 4 of Algorithm 4.1 one searches for a face g, a neighbor to f, which is with a maximal weight $w(g)$, and if $a(g) = 1$ (i.e., unacceptable), one looks for an acceptable face g_a (neighbor to f) such that $w(g_a) = w(g)$. If such a face does not exist, the agglomeration step is terminated and the agglomerated element E is ready.

The way we have put barriers on the faces is based on the element matrices; namely, given a face $f = e_1 \cap e_2$, assemble $A_{e_1 \cup e_2}$ and ask if the dofs on f can be well interpolated from the rest of the dofs in $e_1 \cup e_2$. If the resulting measure of

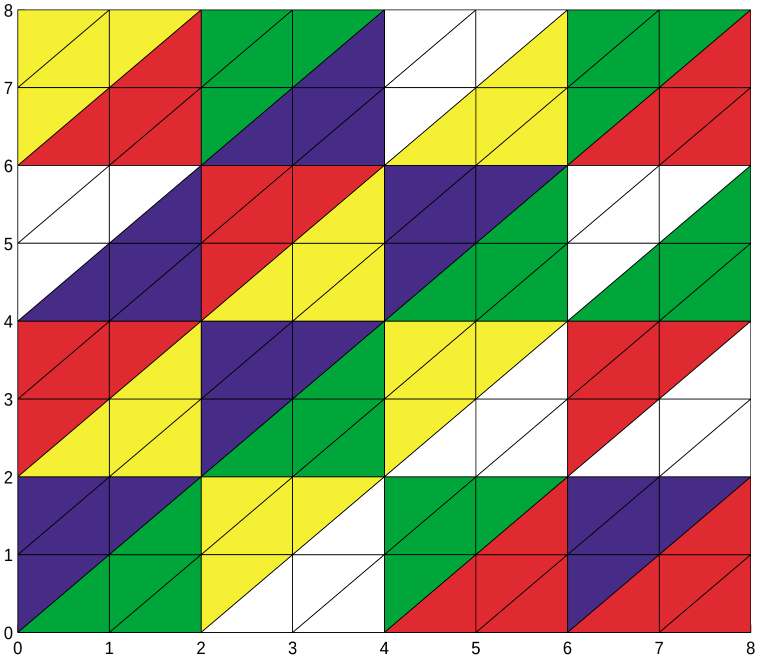


FIG. 4. *Agglomerated elements for structured triangular mesh: One step of agglomeration.*

interpolation quality is reasonable, we say that the face f is acceptable; otherwise, we label f as unacceptable by initializing $a(f) = 1$ to prevent agglomeration of e_1 and e_2. To implement this approach, one must be able to access the quality of the interpolation for the dofs on f. A measure of interpolation quality was proposed in [7]. In our setting, it can be reformulated as follows. Given the interpolation mapping P defined by interpolating dofs on f from the rest of the dofs in $e_1 \cup e_2$, define the quadratic form (or matrix) W_{ff} for vectors on f by

$$\mathbf{v}_f^T W_{ff} \mathbf{v}_f = \inf_{\mathbf{v}_c} (\mathbf{v} + P\mathbf{v}_c)^T A_{e_1 \cup e_2} (\mathbf{v} + P\mathbf{v}_c); \quad \mathbf{v} = \begin{bmatrix} \mathbf{v}_f \\ 0 \end{bmatrix} \begin{matrix} \}\text{dofs on } f, \\ \}e_1 \cup e_2 \setminus f. \end{matrix}$$

Then the measure of interpolation quality (denoted by M_1 in [7]) is

$$m_P = \frac{1}{\lambda_{\min}[D_{ff}^{-1} W_{ff}]}, \tag{4.1}$$

where D_{ff} is, for example, the diagonal of $A_{e_1 \cup e_2}$ restricted to f. Small m_P indicates good quality interpolation; interpolation well approximates functions with low energy. In finite element notation, small m_P means that the functions v_c from the coarse space can approximate well the fine-grid functions v in a weighted L^2-norm $\|.\|_0$. To show this, let m be a bound such that

$$\inf_{v_c} \|v - v_c\|^2_{0,\, e_1 \cup e_2} \le m\, a_{e_1 \cup e_2}(v, v) \quad \text{for all } v: \ v|_{\mathcal{D}_c} = v_c|_{\mathcal{D}_c}. \tag{4.2}$$

This is equivalent (letting $v = v_f + v_c$ above) to

$$\|v_f\|^2_{0,\, e_1 \cup e_2} \le m \inf_{v_c} a_{e_1 \cup e_2}(v_f + v_c,\ v_f + v_c) \quad \text{for all } v_f: \ v_f|_{\mathcal{D}_c} = 0.$$

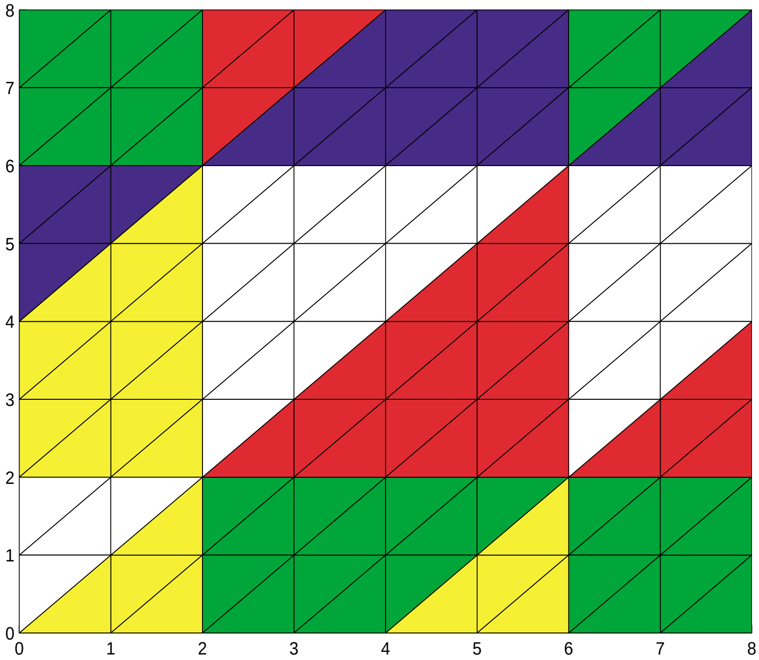


FIG. 5. *Agglomerated elements for structured triangular mesh: Two steps of agglomeration.*

In vector notation, this becomes

$$\begin{aligned}(\mathbf{v}_f)^T D_{ff}\mathbf{v}_f &\le m \inf_{\mathbf{v}_c} (\mathbf{v}+P\mathbf{v}_c)^T A_{e_1\cup e_2}(\mathbf{v}+P\mathbf{v}_c) \text{ for all } \mathbf{v} = \begin{bmatrix} \mathbf{v}_f \\ 0 \end{bmatrix}\\ &= m\ (\mathbf{v}_f)^T W_{ff}\mathbf{v}_f, \text{ for all } \mathbf{v}_f.\end{aligned}$$

This, with the best choice of m, leads to the definition (4.1) of the measure m_P. It is clear, from (4.2), that smaller m_P corresponds to better interpolation quality.

Remark 4.1. One can actually compute the minimum

$$\mathbf{v}_f^T W_{ff}\mathbf{v}_f = \min_{\mathbf{v}_c} (\mathbf{v}+P\mathbf{v}_c)^T A_{e_1\cup e_2}(\mathbf{v}+P\mathbf{v}_c), \quad \mathbf{v} = \begin{bmatrix} \mathbf{v}_f \\ 0 \end{bmatrix} \begin{matrix} \} & \text{dofs on } f, \\ \} & (e_1\cup e_2)\setminus f. \end{matrix}$$

One has, with $A := A_{e_1\cup e_2}$ and $\mathbf{v}_c := t\mathbf{v}_c$ for any $t\in\mathcal{R}$,

$$(\mathbf{v}+tP\mathbf{v}_c)^T A(\mathbf{v}+tP\mathbf{v}_c) = \mathbf{v}^T A\mathbf{v} + 2t\mathbf{v}^T AP\mathbf{v}_c + t^2(P\mathbf{v}_c)^T AP\mathbf{v}_c.$$

The minimum with respect to t is achieved for $t = -\frac{\mathbf{v}^T AP\mathbf{v}_c}{(P\mathbf{v}_c)^T AP\mathbf{v}_c}$ and equals

$$\mathbf{v}^T A\mathbf{v} - \frac{(\mathbf{v}^T AP\mathbf{v}_c)^2}{(P\mathbf{v}_c)^T AP\mathbf{v}_c}.$$

Hence,

$$\mathbf{v}_f^T W_{ff}\mathbf{v}_f = \min_{\mathbf{v}_c}\left(\mathbf{v}_f^T A_{ff}\mathbf{v}_f - \frac{(\mathbf{v}^T AP\mathbf{v}_c)^2}{(P\mathbf{v}_c)^T AP\mathbf{v}_c}\right), \quad \mathbf{v} = \begin{bmatrix} \mathbf{v}_f \\ 0 \end{bmatrix}.$$

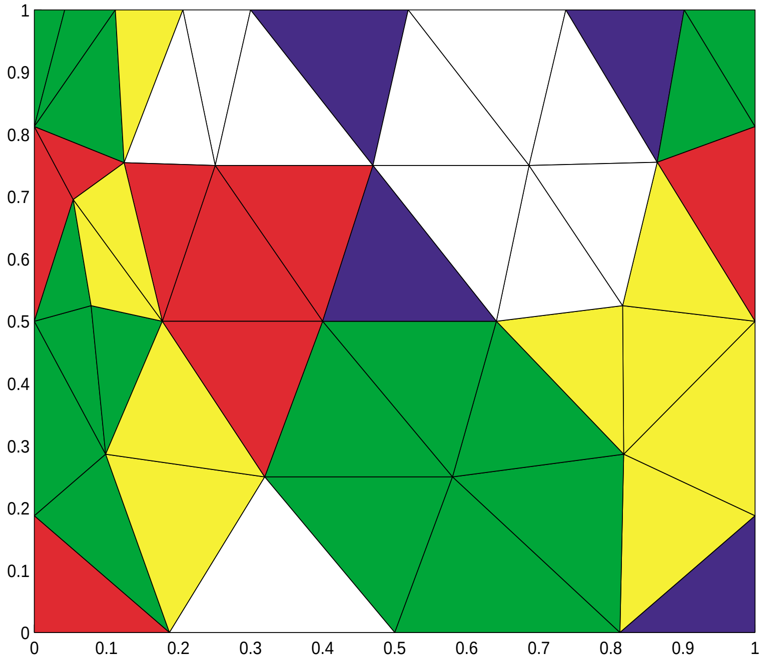


FIG. 6. *Agglomerated elements: Rectangular domain with unstructured triangular elements.*

In particular, $\mathbf{v}_f^T W_{ff} \mathbf{v}_f \le \mathbf{v}_f^T A_{ff} \mathbf{v}_f$. Here, A_{ff} represents the f-f block of $A := A_{e_1 \cup e_2}$ (see (4.3) below). Note that if there is a $\mathbf{v}_c$ such that $(AP\mathbf{v}_c)|_f = 0$, then $\mathbf{v}_f^T W_{ff} \mathbf{v}_f = \mathbf{v}_f^T A_{ff} \mathbf{v}_f$. The latter is true also for the so-called "optimal" P, i.e., such that $P = -A_{ff}^{-1} A_{fc}$, where $A_{e_1 \cup e_2}$ is partitioned as follows:

$$A_{e_1 \cup e_2} = \begin{bmatrix} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{bmatrix} \begin{array}{l} \} \quad \text{dofs on } f, \\ \} \quad (e_1 \cup e_2) \setminus f. \end{array} \tag{4.3}$$

In that case, $m_P = \frac{1}{\lambda_{\min}[D_{ff}^{-1} A_{ff}]}$.

Remark 4.2. Note that if instead of D_{ff} one uses in (4.1) the principal submatrix A_{ff} of A corresponding to the fine dofs that are not coarse, then $m_P = \frac{1}{1-\gamma^2}$, where $\gamma \in [0, 1)$ stands for the cosine of the abstract angle between the coarse space $V_c = \{\mathbf{v}^c = P\mathbf{v}_c\}$ and its hierarchical complement $V_f = \{\mathbf{v}^f = [\begin{smallmatrix} \mathbf{v}_f \\ 0 \end{smallmatrix}]\}$. The angle is measured in the energy inner product, i.e.,

$$(\mathbf{v}^f)^T A_{e_1 \cup e_2} \mathbf{v}^c \le \gamma \sqrt{(\mathbf{v}^f)^T A_{e_1 \cup e_2} \mathbf{v}^f} \sqrt{(\mathbf{v}^c)^T A_{e_1 \cup e_2} \mathbf{v}^c} \;\text{ for all } \mathbf{v}^f \in V_f, \; \mathbf{v}^c \in V_c.$$

For a proof of the relation $m_P = \frac{1}{1-\gamma^2}$, see, e.g., Vassilevski [16].

Instead of m_P one can use γ as a measure of the interpolation quality. Then small γ will correspond to small m_P and hence to good quality interpolation, whereas γ close to one will imply large m_P and hence poor quality interpolation.

In following example, we will use γ to define a measure for strength on connections between neighboring elements and thus label faces as acceptable or unacceptable. Consider two fine elements e_1 and e_2 sharing a face f as shown in Figure 16. Let i_{F_2} be an interpolation rule for dof x_3 from x_1 and x_5, and let i_{F_1} be an interpolation rule for

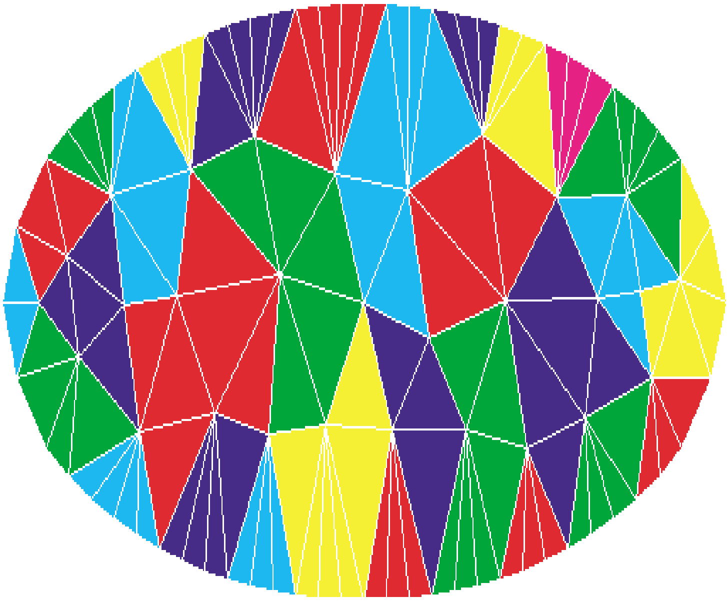


FIG. 7. *Agglomerated elements: Elliptical domain with triangular elements.*

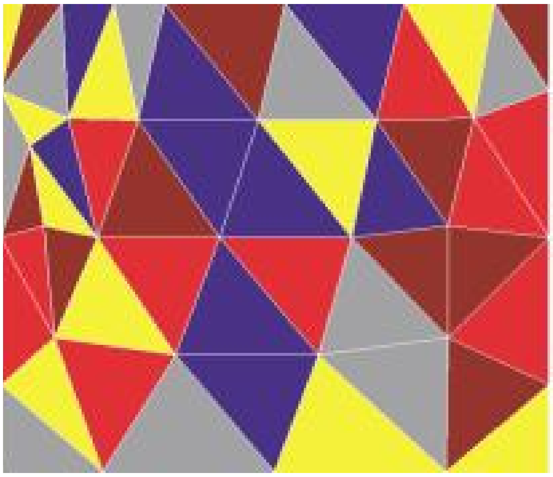


FIG. 8. *Fine elements: Rectangular domain with* 48 *unstructured triangular elements.*

dof x_4 from x_2 and x_6; these could be constructed as proposed in the previous section. For 2D scalar elliptic problems with constant coefficients, these are linear interpolants along the faces F_1 and F_2 treating x_1, x_2, x_5, and x_6 as coarse-grid nodes and x_3 and x_4 as complementary to the coarse-grid, fine-grid nodes. Then, given a coarse function v_c defined at the nodes x_1, x_2, x_5, and x_6, the mapping $P_c^f v_c = \left\{ \begin{smallmatrix} i_{F_1} v_c, & x=x_4, \\ i_{F_2} v_c, & x=x_3 \end{smallmatrix} \right.$ defines a coarse-to-fine prolongation operator.

Let $E = e_1 \cup e_2$, and let A_E be the assembled matrix corresponding to E. Given

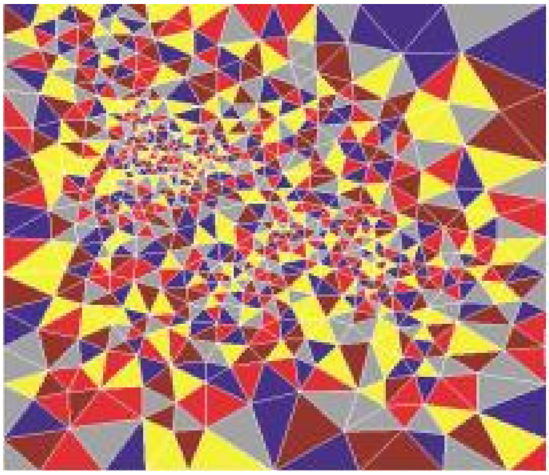


FIG. 9. *Fine elements: Rectangular domain with* 1001 *unstructured triangular elements.*

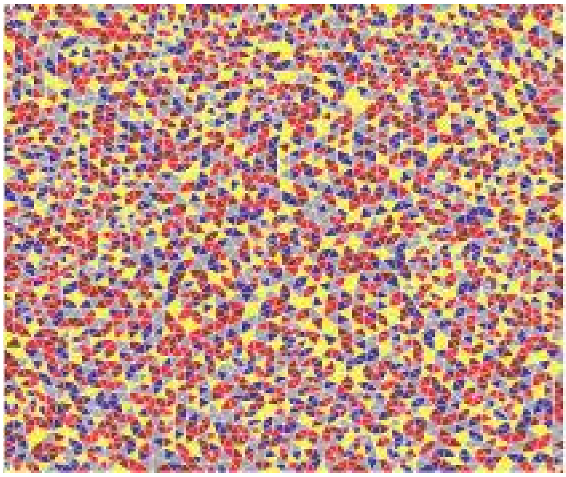


FIG. 10. *Fine elements: Rectangular domain with* 4016 *unstructured triangular elements.*

a coarse-grid vector $\mathbf{v}_c$, let $\widehat{v}_c = P_c^f v_c$ be its representation on the fine-grid. Then the local fine-grid space is decomposed as $P_c^f v_c \oplus v_f^0$, where v_f^0 are the fine-grid functions which vanish on the coarse-grid. As mentioned, the cosine $\gamma \in [0, 1)$ of the angle between these components can be used to measure a strength of connection between e_1 and e_2 with respect to the given matrix A_E (or pair of element matrices A_{e_1} and A_{e_2} that correspond to the pair of elements e_1 and e_2). Recall that the constant γ is defined as the best constant in the strengthened Cauchy inequality

$$a_E(\widehat{v}_c,\ v_f^0) \leq \gamma\ \sqrt{a_E(\widehat{v}_c,\ \widehat{v}_c)}\ \sqrt{a_E(v_f^0,\ v_f^0)} \quad \text{for all } \widehat{v}_c,\ v_f^0. \tag{4.4}$$

To write this inequality in matrix-vector notation, let

$$P = \left[\left[\begin{array}{c} I \\ 0 \end{array} \right], P_c^f \right]$$

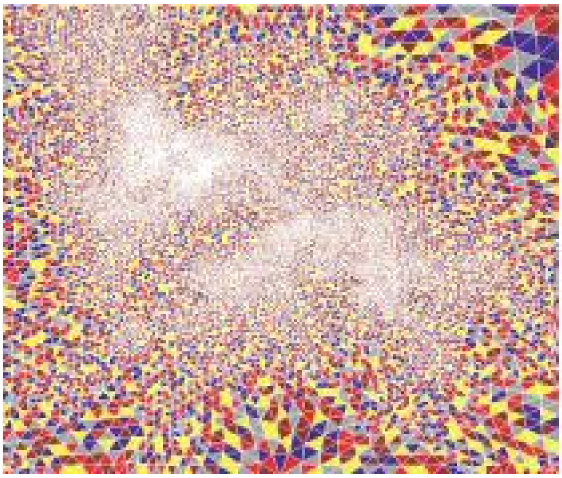


Fig. 11. *Fine elements: Rectangular domain with* 16000 *unstructured triangular elements.*

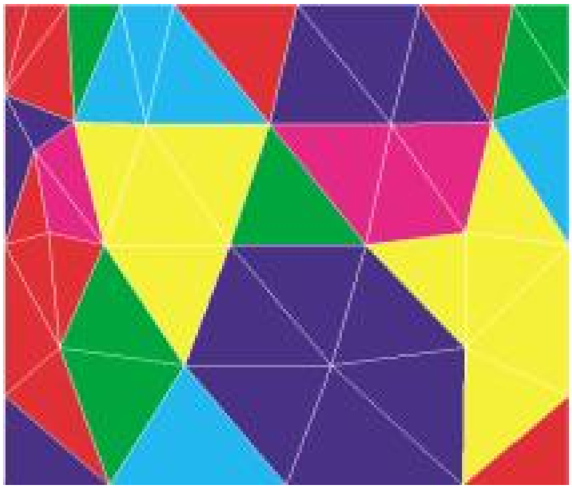


Fig. 12. *Agglomerated elements: Rectangular domain with unstructured triangular elements.*

and $\widehat{A}_E = P^T A_E P$. Consider the following two-by-two blocking of $\widehat{A}_E$:

$$\widehat{A}_E = \begin{bmatrix} A_{E;\,ff} & \widehat{A}_{E;\,fc} \\ \widehat{A}_{E;\,cf} & A_{E;\,cc} \end{bmatrix} \begin{array}{l} \} \quad \text{complementary fine-grid nodes; i.e., } x_3,\ x_4, \\ \} \quad \text{coarse nodes; i.e., } x_1, x_2, x_5,\ x_6. \end{array}$$

Note that $A_{E;\,cc}$ is the resulting coarse matrix corresponding to E. Then the strengthened Cauchy inequality (4.4) reads

$$\mathbf{v}_c^T \widehat{A}_{E;\,cf} \mathbf{v}_f^0 \le \gamma \sqrt{\mathbf{v}_c^T A_{E;\,cc} \mathbf{v}_c} \sqrt{{\mathbf{v}_f^0}^T A_{E;\,ff} \mathbf{v}_f^0} \quad \text{for all } \mathbf{v}_c,\ \mathbf{v}_f^0.$$

A way to compute γ is to find the largest eigenvalue $m = \lambda_{\max} \ge 1$ of the generalized eigenvalue problem

$$A_{E,\,cc}\mathbf{q} = \lambda S_{E,\,f}\mathbf{q},$$

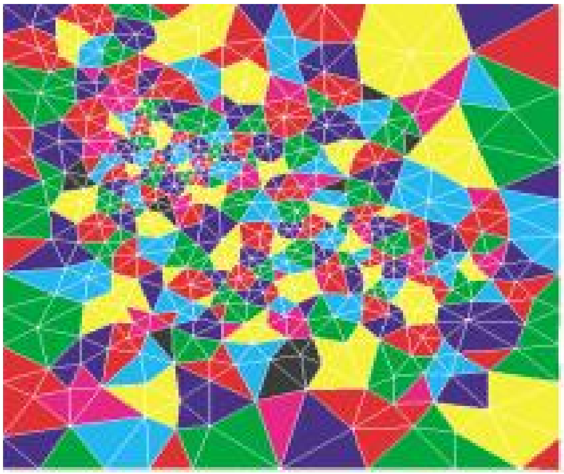


Fig. 13. *Agglomerated elements: Rectangular domain with unstructured triangular elements.*

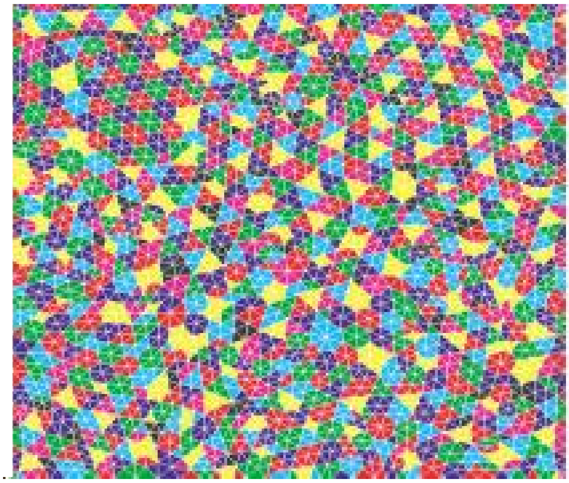


Fig. 14. *Agglomerated elements: Rectangular domain with unstructured triangular elements.*

where $S_{E,\,f}$ is the Schur complement of $\widehat{A}_E$ on f, i.e.,

$$S_{E,\,f} = A_{E,\,cc} - \widehat{A}_{E;\,cf}\,(A_{E;\,ff})^{-1}\,\widehat{A}_{E;\,fc}.$$

Then $\gamma = \sqrt{1 - \frac{1}{m}}$.

Definition 4.1 (strongly connected elements). *We call e_1 and e_2 strongly connected if γ is close to zero, i.e., when the resulting local coarse space is almost orthogonal to its complementary (the so-called two-level hierarchical complementary) space.*

Algorithm 4.1 can be modified to agglomerate only strongly connected elements. One would set a threshold α and label a face f unacceptable if $\gamma > \alpha$ by initializing $a(f) = 1$.

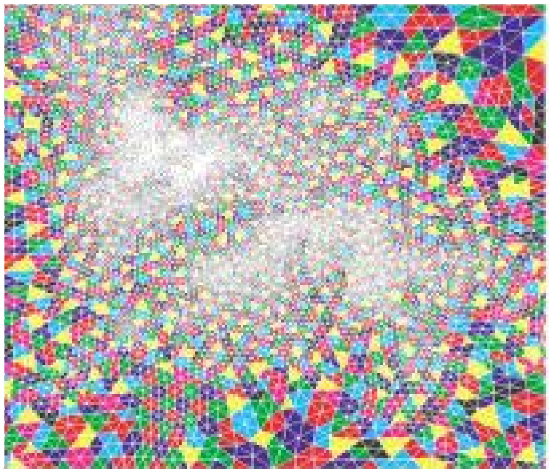


FIG. 15. *Agglomerated elements: Rectangular domain with unstructured triangular elements.*

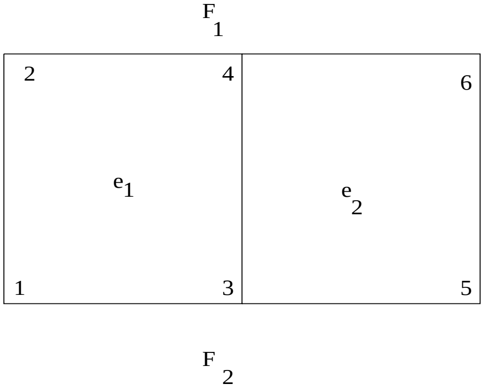


FIG. 16. *Neighboring elements* e_1 *and* e_2 *with a common face* $f = \{x_3, x_4\}$*; the nodes* x_1, x_2, x_5*, and* x_6 *are viewed as coarse-grid nodes.*

4.1. Examples of γ. We conclude this section with examples showing that this definition of strongly connected elements can lead to the correct semicoarsening for anisotropic problems. Consider the model second-order elliptic bilinear form, which, restricted to an element e, reads

$$a_e(\varphi, \psi) = \int_e \left(\frac{\partial \varphi}{\partial x} \frac{\partial \psi}{\partial x} + \frac{\partial \varphi}{\partial y} \frac{\partial \psi}{\partial y} \right) dx\, dy. \tag{4.5}$$

Consider two vertically adjacent rectangular elements (see Figure 17) and bilinear test functions. Consider the following cases.

(a) Anisotropic elements $h_x < h_y$; $h_x = 0.1h_y$, $\gamma = 0.8649$; $h_x = 0.01h_y$, $\gamma = 0.8660$. These values of γ indicate that the elements are weakly connected and one should not agglomerate them.

(b) Anisotropic elements $h_x > h_y$; $h_x = 10h_y$, $\gamma = 0.1698$; $h_x = 100h_y$, $\gamma = 0.0173$. This example shows that since γ is close to zero, the elements are strongly connected, and hence one should agglomerate this pair of elements.

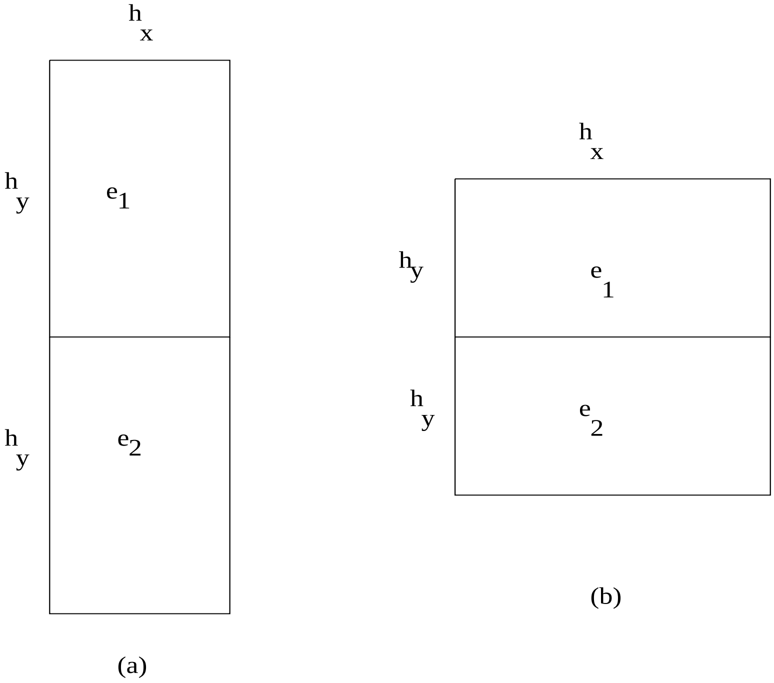


FIG. 17. *Neighboring elements* e_1 *and* e_2*;* (*a*) $h_x < h_y$, (*b*) $h_x > h_y$.

(c) For comparison, if $h_x = h_y$, $\gamma = 0.7746$ (or $\gamma^2 = \frac{3}{5}$).

Thus, this measure correctly leads to coarsening only in the direction of small mesh size.

5. Numerical experiments.

5. Numerical experiments. In this section we present some preliminary numerical results that show the potential of the proposed element agglomeration AMGe method.

We have tested the two-grid method with the coarse-grid obtained using the agglomeration algorithm described in section 4. After the coarse dofs were selected the interpolation mapping was constructed as described in section 2. We used one forward Gauss–Seidel iteration as a presmoother and one backward Gauss–Seidel iteration for a postsmoothing. The stopping criterion was a relative reduction of the residual ℓ^2-norm by a factor of 10^{-6}.

We tested two sets of problems.

- The Poisson equation discretized on a square domain on four “unstructured” rectangular grids are shown in Figures 8, 9, 10, and 11, and the respective grids with agglomerated elements are shown in Figure 12, 13, 14, and 15. Dirichlet boundary conditions were imposed, and the results are collected in Table 1.
- The elasticity equation which comes from minimizing the quadratic functional discretized with square bilinear elements.

(5.1)
$$\int_{\Omega} \left[\frac{1+\nu}{2}(\partial_x u + \partial_y v)^2 + \frac{1-\nu}{2}(\partial_x u - \partial_y v)^2 + \frac{1-\nu}{2}(\partial_y u + \partial_x v)^2 \right] \, dxdy.$$

Table 1

Two-grid convergence results; unstructured triangular grid; Laplace operator; Gauss–Seidel smoother.

Grid #	1	2	3	4
# fine elements	48	1 001	4 016	16 016
# coarse elements	20	242	1 016	3 859
# fine dof	35	523	2 085	8 095
# coarse dof	27	281	1 083	3 515
# iterations	7	9	8	8
ϱ	0.159	0.320	0.256	0.260

Table 2

Two-grid convergence results; structured rectangular grid; elasticity operator; Gauss–Seidel smoother.

Grid #	1	2	3	4
# fine elements	400	900	1600	2500
# coarse elements	118	253	438	673
# fine dof	882	1922	3362	5202
# coarse dof	314	624	1034	1544
# iterations	9	9	9	9
ϱ	0.251	0.245	0.254	0.248

Here $\nu = \frac{1}{3}$. Again, Dirichlet boundary conditions were imposed, and these results are in Table 2.

One notices the similar convergence factors ϱ and the number of iterations for Poisson and elasticity problems.

Acknowledgments. The present authors would like to acknowledge the stimulating discussions with the authors of [7] without which this research direction may not have been pursued. The unstructured triangular grids used in our numerical tests were provided by Dr. Ludmil Zikatanov. Finally, the careful reading of the paper by the referees is gratefully appreciated.

REFERENCES

[1] O. Axelsson, *Iterative Solution Methods*, Cambridge University Press, New York, 1994.

[2] R. E. Bank, *Hierarchical bases and the finite element method*, in Acta Numerica, 1996, Acta Numer. 5, Cambridge University Press, Cambridge, UK, 1996, pp. 1–43.

[3] R. E. Bank and T. Dupont, *Analysis of a Two-Level Scheme for Solving Finite Element Equations*, Report CNA-159, Center for Numerical Analysis, The University of Texas at Austin, Austin, TX, 1980.

[4] J. H. Bramble, *Multigrid Methods*, 2nd ed., Pitman Res. Notes Math. Ser. 234, Longman, Harlow, UK, 1995.

[5] A. Brandt, S. McCormick, and J. W. Ruge, *Algebraic Multigrid (AMG) for Automatic Multigrid Solutions with Application to Geodetic Computations*, Report, Institute for Computational Studies, Fort Collins, CO, 1982.

[6] A. Brandt, S. McCormick, and J. W. Ruge, *Algebraic multigrid (AMG) for sparse equations*, in Sparsity and Its Applications (Loughborough, 1983), D. J. Evans, ed., Cambridge University Press, Cambridge, UK, 1985, pp. 257–284.

[7] M. Brezina, A. J. Cleary, R. D. Falgout, V. E. Henson, J. E. Jones, T. A. Manteuffel, S. F. McCormick, and J. W. Ruge, *Algebraic multigrid based on element interpolation (AMGe)*, SIAM J. Sci. Comput., 22 (2000), pp. 1570–1592.

[8] T. F. Chan and B. F. Smith, *Domain decomposition and multigrid algorithm for elliptic problems on unstructured meshes*, in Domain Decomposition Methods in Scientific and Engineering Computing, Proceedings of the 7th International Conference on Domain Decomposition Methods, Contemporary Math. 180, AMS, Providence, RI, 1994, pp. 175–189.

[9] T. F. Chan, J. Xu, and L. T. Zikatanov, *An agglomeration multigrid method for unstructured grids*, in Domain Decomposition Methods 10, Proceedings of the 10th International Conference on Domain Decomposition Methods, Contemporary Math. 218, AMS, Providence, RI, 1998, pp. 67–81.

[10] R. P. Fedorenko, *Finite superelements method*, in Recent Advances in Numerical Methods and Applications, II, Proceedings of the Fourth International Conference, NMA'98, Sofia, Bulgaria, Iliev, Kaschiev, Margenov, Sendov, and Vassilevski, eds., World Scientific, Singapore, 1999, pp. 16–26.

[11] T. Y. Hou, X.-H. Wu, and Z. Cai, *Convergence of a multiscale finite element method for elliptic problems with rapidly oscillating coefficients*, Math. Comp., 68 (1999), pp. 913–943.

[12] J. Mandel, M. Brezina, and P. Vanek, *Energy optimization of algebraic multigrid bases*, Computing, 62 (1999), pp. 205–228.

[13] J. W. Ruge and K. Stüben, *Algebraic multigrid*, in Multigrid Methods, S. McCormick, ed., SIAM, Philadelphia, 1987, pp. 73–130.

[14] K. Stüben, *Algebraic Multigrid (AMG): An Introduction with Applications*, GMD report 53, GMD - Forschungszentrum Informationstechnik GmbH, Schloss Birlinghoven, Sankt Augustin, Germany, 1999.

[15] P. Vanek, J. Mandel, and M. Brezina, *Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems*, Computing, 56 (1996), pp. 179–196.

[16] P. S. Vassilevski, *On two ways of stabilizing the hierarchical basis multilevel methods*, SIAM Rev., 39 (1997), pp. 18–53.

[17] W. L. Wan, T. F. Chan, and B. Smith, *An energy-minimizing interpolation for robust multigrid methods*, SIAM J. Sci. Comput., 21 (2000), pp. 1632–1649.

SIAM J. Sci. Comput.
Vol. 23, No. 1, pp. 134–156

ASYNCHRONOUS PARALLEL PATTERN SEARCH FOR NONLINEAR OPTIMIZATION*

PATRICIA D. HOUGH†, TAMARA G. KOLDA†, AND VIRGINIA J. TORCZON‡

Abstract. We introduce a new asynchronous parallel pattern search (APPS). Parallel pattern search can be quite useful for engineering optimization problems characterized by a small number of variables (say, fifty or less) and by objective functions that are expensive to evaluate, such as those defined by complex simulations that can take anywhere from a few seconds to many hours to run. The target platforms for APPS are the loosely coupled parallel systems now widely available. We exploit the algorithmic characteristics of pattern search to design variants that dynamically initiate actions solely in response to messages, rather than routinely cycling through a fixed set of steps. This gives a versatile concurrent strategy that allows us to effectively balance the computational load across all available processors. Further, it allows us to incorporate a high degree of fault tolerance with almost no additional overhead. We demonstrate the effectiveness of a preliminary implementation of APPS on both standard test problems as well as some engineering optimization problems.

Key words. asynchronous parallel optimization, pattern search, direct search, fault tolerance, distributed computing, cluster computing

AMS subject classifications. 65K05, 90C56, 65Y05, 68W15, 90C90

PII. S1064827599365823

1. Introduction.

We consider solving the unconstrained nonlinear optimization problem, minimize $f(x)$, where $x \in \mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}$. The problems of particular interest to us are defined by computationally expensive computer simulations of complex physical processes. Such simulations may take anywhere from a few seconds to many hours of computation on a single processor. In addition, we often cannot use derivative-based methods to solve these problems because no procedure exists for the evaluation of the gradient and the function evaluations are not precise enough to produce an accurate finite-difference gradient.

Pattern search is a class of direct search methods that is popular for solving the problems described above because no derivative information is required. Further, pattern search methods admit a wide range of algorithmic possibilities; see, e.g., [15, 16, 26]. The dominant computational cost for pattern search methods lies in the evaluation of the objective function. We can exploit the definition of pattern search to derive variants that perform multiple independent function evaluations simultaneously. We then can take advantage of parallel computing platforms to reduce the overall computational cost of the search.

[*] Received by the editors January 19, 2000; accepted for publication (in revised form) January 23, 2001; published electronically June 8, 2001. This research was sponsored by the Mathematical, Information, and Computational Sciences Division at the U.S. Department of Energy and by Sandia National Laboratory, a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the U.S. Department of Energy under contract DE–AC04–94AL85000. The U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes. Copyright is owned by SIAM to the extent not limited by these rights.

http://www.siam.org/journals/sisc/23-1/36582.html

[†] Computational Sciences and Mathematics Research Department, Sandia National Laboratories, Livermore, CA 94551–9217 (pdhough@sandia.gov, tgkolda@sandia.gov).

[‡] Department of Computer Science, College of William and Mary, Williamsburg, VA 23187–8795 (va@cs.wm.edu). The research of this author was funded in part by the Computer Science Research Institute at Sandia National Laboratories and by the National Science Foundation under grant CCR–9734044.

Both the nature of the problems of interest and the features of current distributed computing environments raise some issues that we address in this work.

The original investigation into parallel direct search methods [7, 25] made two fundamental assumptions about the parallel computing environment: (1) that the processors were both homogeneous and tightly coupled and (2) that the amount of time needed to finish a single evaluation of the objective function was effectively constant. It is time to reexamine these two assumptions. Clearly, given the current variety of parallel computing platforms, including distributed systems comprising loosely coupled, often heterogeneous, off-the-shelf commercial components [24], the first assumption is no longer valid. The second assumption may not hold in our case because we focus on problems defined by the simulations of complex physical processes. Typically, the simulations themselves are based on iterative numerical techniques and so the assumption that evaluations of the objective finish in constant computational time on equivalent processors often does not hold. In fact, the behavior of the simulation for any given input is difficult to assess in advance since it can vary substantially depending on a variety of factors.

Because the original assumptions underlying parallel direct search are not valid for the situations we now face, we can no longer assume that the computation proceeds in lockstep. A single synchronization step at the end of every iteration, as in [25], is neither appropriate nor effective when any of the following factors holds: function evaluations finish in varying amounts of time (even on equivalent processors), the processors employed in the computation possess different performance characteristics, or the processors have varying loads. Our goal is to introduce a class of asynchronous parallel pattern search (APPS) methods that make more effective use of a variety of computing environments, as well as to devise strategies that accommodate the variation in completion time for function evaluations. Our approach is outlined in section 3.

Another consideration we address in this paper is incorporating fault-tolerant strategies into APPS since one intent is to use this software on large-scale systems. As the number of individual computers participating in a computation grows, the chance that one (or more) will fail also grows. If we embark on a lengthy computation, we want reasonable assurance of producing a final result, even if a subset of processors fails. Thus, our goal is to design methods that respond to such failures and protect the solution process. Rather than simply checkpointing intermediate computations to disk and then restarting in the event of a failure, we are instead considering methods with heuristics that adaptively modify the search strategy. We discuss the technical issues in further detail in section 4.

In section 5 we provide numerical results, for both standard and engineering optimization test problems, that compare a preliminary implementation of APPS with an implementation of parallel pattern search (PPS) that incorporates a blocking synchronization point within each iteration. Finally, in section 6 we outline additional questions to pursue.

Although we are not the first to embark on the design of asynchronous parallel optimization algorithms, we are aware of little other work, particularly in the area of nonlinear programming. Approaches to developing asynchronous parallel Newton or quasi-Newton methods are proposed in [4, 10], though the assumptions underlying these approaches differ markedly from those we address. Specifically, both assume that solving the Newton equation at each iteration is the dominant computational cost of the optimization algorithm because the dimensions of the problems of interest are relatively large. A different line of inquiry [23] considers the use of quasi-Newton methods in the context of asynchronous stochastic global optimization algorithms; we consider only the problem of identifying local stationary points.

2. Parallel pattern search.

Before proceeding to a discussion of APPS, let us first review some key features of pattern search.

A primary characteristic of pattern search methods is that they sample the function over a predefined pattern of points, all of which lie on a rational lattice. By enforcing structure on the form of the points in the pattern, as well as simple rules on both the outcome of the search and the subsequent updates, we are guaranteed global convergence to a stationary point [9, 16, 26].

For our purposes, the feature of pattern search that is amenable to parallelism is that once the candidates in the pattern have been defined, the function values at these points can be computed independently and, thus, concurrently.

To make this more concrete, consider the following particularly simple version of a parallel pattern search algorithm. At iteration k, we have an iterate $x_k \in \mathbb{R}^n$ and a steplength control parameter $\Delta_k > 0$. The pattern of p search directions is denoted by $\mathcal{D} = \{d_1, \ldots, d_p\}$. Although other choices for $\mathcal{D}$ are possible, for our simple variant we choose $\mathcal{D} \equiv \{e_1, \ldots, e_n, -e_1, \ldots, -e_n\}$, where e_j represents the jth unit vector. Figure 2.1 illustrates an example of this search pattern when $n = 2$.

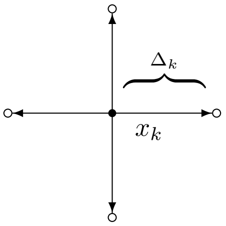


FIG. 2.1. *A simple instance of a pattern for pattern search.*

Now that we have selected $\mathcal{D}$, multiple algorithmic options are open to us. An obvious strategy for concurrent computing is to identify an $x_+ \in \{x_k + \Delta_k d_i,\ i = 1, \ldots, p\}$ such that $f(x_+) = \min\{f(x_k + \Delta_k d_i),\ i = 1, \ldots, p\}$. This strategy requires us to compute $f(x_k + \Delta_k d_i)$ for all p vectors in the set $\mathcal{D}$. To ensure global convergence of some subsequence to a stationary point we can accept any point $x_k + \Delta_k\, d_i$ for which $f(x_k + \Delta_k d_i) < f(x_k)$ [26]. Thus, finding $f(x_+) = \min\{f(x_k + \Delta_k d_i),\ i = 1, \ldots, p\}$ is in some sense more than is really needed. However, concurrency masks the computational expense of the stronger acceptance condition.

If none of the points in the pattern reduces the objective, then we set $x_{k+1} = x_k$ and reduce Δ by setting $\Delta_{k+1} = \frac{1}{2}\Delta_k$; otherwise, we set $\Delta_{k+1} = \Delta_k$ and $x_{k+1} = x_+$. We repeat this process until some reasonable stopping criterion, such as $\Delta_k \leq tol$, is satisfied [8, 9]. This basic strategy leads us to the algorithm we call parallel pattern search (PPS), which is given in Figure 2.2.

There still remains the question of what constitutes an acceptable pattern. Following the examples in [16], we borrow the following definition from [6].

DEFINITION 2.1. *A set of vectors $\{d_1, \ldots, d_p\}$ positively spans $\mathbb{R}^n$ if any vector $v \in \mathbb{R}^n$ can be written as a nonnegative linear combination of the vectors in the set; i.e., for any $v \in \mathbb{R}^n$ there exist $\alpha_1, \alpha_2, \ldots, \alpha_p \geq 0$ such that*

$$v = \alpha_1 d_1 + \cdots + \alpha_p d_p.$$

```
Initialization:
    • Set the iteration counter $k = 0$.
    • Select a set of search directions $\mathcal{D} = \{d_1, \dots, d_p\}$.
    • Select a steplength control parameter $\Delta_0$.
    • Select a stopping tolerance $tol$.
    • Select a starting point $x_0$ and evaluate $f(x_0)$.
Iteration:
    1. Compute $x_k + \Delta_k d_i$ and evaluate $f(x_k + \Delta_k d_i)$, for $i = 1, \dots, p$, concurrently.
    2. Determine $x_+$ and $f(x_+)$ such that $f(x_+) = \min \{ f(x_k + \Delta_k d_i),\ i = 1, \dots, p \}$
       (synchronization point).
    3. If $f(x_+) < f(x_k)$, then $x_k \leftarrow x_+$ and $f(x_k) \leftarrow f(x_+)$. Else $\Delta_k \leftarrow \frac{1}{2}\Delta_k$.
    4. If $\Delta_k > tol$, $k \leftarrow k + 1$, go to step 1. Else, exit.
```

FIG. 2.2. *The PPS algorithm.*

We require $\mathcal{D}$ to be a *positive spanning set* for $\mathbb{R}^n$. (This is a bit of a misnomer; given the definition, it perhaps would be more apt to call it a “nonnegative” spanning set.) We add the condition that $\mathcal{D}$ be composed of rational vectors [16].

3. Asynchronous parallel pattern search.

3. Asynchronous parallel pattern search. The appeal of the PPS strategy outlined in Figure 2.2 is that it is straightforward to implement. Unfortunately, inefficiencies in processor utilization for PPS arise when the objective function evaluations do not finish in approximately the same amount of time. This may happen for several reasons. First, the objective function evaluations may be complex simulations that require different amounts of work depending on the input parameters. Second, the computational loads on the individual processors may vary. Third, the processors participating in the calculation may possess different computational characteristics. When the objective function evaluations take varying amounts of time, those processors that can finish their share of the computation more quickly wait for the remaining processors to contribute their results. Fourth, the number of processes we are interested in executing may not exactly match the number of available processors. Finally, there is the real risk that either processes or processors may fail during the course of the computation. For all these reasons, we pursue a more versatile concurrent strategy, which we call asynchronous parallel pattern search (APPS), that allows us to effectively balance the computational load across the available processors.

Were we simply interested in load-balancing issues, the *master-slave* paradigm for the design of parallel programs would be inviting. Given such a design perspective, we could localize all decision making to a single process (the master) and devote all remaining processes (the slaves) to the evaluation of $f(x_{\text{trial}})$ for various choices of x_{trial} determined by the master process. Since we are assuming that the evaluation of the objective is the dominant computational cost, we would not have to be overly concerned about the communication bottlenecks that can sometimes occur using such a paradigm. However, fault tolerance is our other prominent concern. If we localize the decision making to a single process and the master process fails, we would be unable to finish the computation. (Recovery in the event that one of the slave processes fails is easy; once a failure is detected, the master process can simply restart the failed slave process.)

Such concerns lead us to the *peer-to-peer* paradigm. We want each process to be an independent unit, capable of making its own decisions and equipped to respond intelligently whenever it detects that other processes have failed.

```
Initialization:
    • Determine my search direction $d \in \mathcal{D}$.
    • Receive the initial value of the steplength control parameter $\Delta_{\text{trial}}$.
    • Receive the value of the stopping tolerance $tol$.
    • Receive the starting point $x_{\text{best}}$ and receive (or evaluate) $f_{\text{best}} \equiv f(x_{\text{best}})$.
Iteration:
    1. Compute $x_{\text{trial}} \leftarrow x_{\text{best}} + \Delta_{\text{trial}}\, d$ and evaluate $f_{\text{trial}} \equiv f(x_{\text{trial}})$.
    2. Perform a global reduction to determine $f_+$ (and the associated $x_+$) such that $f_+$ is the minimum of all $f_{\text{trial}}$ values on all $p$ processes.
    3. (a) If $f_+ < f_{\text{best}}$, then
              i. $\Delta_{\text{trial}} \leftarrow \lambda\, \Delta_{\text{trial}}$ with $\lambda \in \{2^{\ell} \mid \ell \in \mathbb{Z}_+\}$ and
              ii. $\{x_{\text{best}}, f_{\text{best}}\} \leftarrow \{x_+, f_+\}$.
       (b) Else $\Delta_{\text{trial}} \leftarrow \frac{1}{2}\Delta_{\text{trial}}$.
    4. If $\Delta_{\text{trial}} > tol$, go to step 1. Else, exit.
```

FIG. 3.1. *Peer-to-peer version of (synchronous) PPS.*

3.1. Peer-to-peer synchronous PPS.

In order to better understand APPS, let us first consider a peer-to-peer version of synchronous PPS.

For PPS, there are p processes, with each process in charge of a single search direction in the set $\mathcal{D}$. In Figure 3.1, we show the peer-to-peer version of synchronous PPS from the perspective of a single process. We drop the subscript $i = \{1, \ldots, p\}$ to emphasize that each process is only concerned with its own unique direction. In the initialization, each process determines its search direction and "receives" the values of Δ_{trial}, tol, and x_{best} either by reading them from an input file or by receiving them in a message from another process.

In the main iteration of PPS, the only communication a process has with its peers is the reduction in step 2, where all the processes participating in the computation contribute their values for x_{trial} and f_{trial}. The reduction operation returns f_+, the minimum value of f_{trial} over all processes, and x_+, the associated point. This reduction operation is the synchronization point for PPS—the minimum value of f_{trial} over all processes cannot be determined until all processes have finished their evaluation of $f(x_{\text{trial}})$.

As indicated in step 3(a)i, we may increase Δ_{trial} when a decrease in f is obtained. We have two possible reasons for doing so. First, we do not want Δ_{trial} to become too small based on the outcome of a search along a single direction. So if we find a step that produces decrease in f, but for which Δ_{trial} is smaller than some $\Delta_{\min} > tol$, then we choose the least nonnegative integer ℓ (i.e., $\ell \in \mathbb{Z}_+$) such that $2^{\ell}\Delta_{\text{trial}} > \Delta_{\min}$ (in our implementation we somewhat arbitrarily choose $\Delta_{\min} \equiv 2^3 \cdot tol$). Assuming, instead, that we ended the search successfully with a choice of Δ_{trial} that satisfies $\Delta_{\text{trial}} > \Delta_{\min}$, we may still choose to expand Δ_{trial}. In our implementation we double Δ_{trial} (i.e., we choose $\ell = 1$) if the same search direction produces at least two successful iterates in a row. Our reason for this condition is straightforward: if we have just completed a sequence of reductions in Δ_{trial} to arrive at a steplength that is sufficiently small to produce descent, it is counterproductive to follow this with an immediate doubling of Δ_{trial}. However, if the same search direction produces at least two successful iterates in a row, then this would indicate that the size of the step we are taking is probably too short, so we double Δ_{trial} in an effort to accelerate the search along that direction. If neither of the above two situations holds, then we do not alter Δ_{trial} (i.e., we choose $\ell = 0$).

On the other hand, if there is no decrease in f, in step 3(b) we reduce Δ_{trial} by a factor of one-half.

In step 4, all processes simultaneously check for convergence, each using its own locally stored, locally updated copy of Δ_{trial}. We note that in a heterogeneous environment, there exists the possibility that the processes may not have identical values for Δ_{trial} because of slight differences in both storage and arithmetic for floating-point numbers; see [2]. We address this issue in further detail in section 3.3.2.

```
Iteration:
    0. For each **new best** message in my queue:
        (a) If $f_+ < f_{\text{best}}$ for an incoming triplet $\{x_+, f_+, \Delta_+\}$, then
            i. $\{x_{\text{best}}, f_{\text{best}}, \Delta_{\text{best}}\} \leftarrow \{x_+, f_+, \Delta_+\}$, and
            ii. $\Delta_{\text{trial}} \leftarrow \Delta_{\text{best}}$.
        (b) Else, discard the triplet $\{x_+, f_+, \Delta_+\}$.
    1. Compute $x_{\text{trial}} \leftarrow x_{\text{best}} + \Delta_{\text{trial}}\, d$ and evaluate $f_{\text{trial}} \equiv f(x_{\text{trial}})$.
    2. Set $\{x_+, f_+, \Delta_+\} \leftarrow \{x_{\text{trial}}, f_{\text{trial}}, \Delta_{\text{trial}}\}$.
    3. (a) If $f_+ < f_{\text{best}}$, then
            i. $\Delta_{\text{trial}} \leftarrow \lambda\, \Delta_{\text{trial}}$ with $\lambda \in \{2^\ell \mid \ell \in \mathbb{Z}_+\}$; $\Delta_+ \leftarrow \Delta_{\text{trial}}$;
            ii. $\{x_{\text{best}}, f_{\text{best}}, \Delta_{\text{best}}\} \leftarrow \{x_+, f_+, \Delta_+\}$; and
            iii. *broadcast* a nonblocking **new best** message with the triplet
                 $\{x_{\text{best}}, f_{\text{best}}, \Delta_{\text{best}}\}$.
        (b) Else $\Delta_{\text{trial}} \leftarrow \frac{1}{2}\Delta_{\text{trial}}$.
    4. If $\Delta_{\text{trial}} > tol$, go to step 0. Else *broadcast* a nonblocking **single direction
       convergence** message with the triplet $\{x_{\text{best}}, f_{\text{best}}, \Delta_{\text{best}}\}$.
    5. Wait and process each incoming message in my queue until either
       (a) enough of the processes report single direction convergence for this same point
       or
       (b) a better point is received.
       In case (a), exit. In case (b), go to step 0.
```

FIG. 3.2. *Peer-to-peer version of APPS.*

3.2. Peer-to-peer APPS. The peer-to-peer version of APPS, from the perspective of a single process, is given in Figure 3.2. Note that the process's local values for x_{best}, x_+, Δ_{trial}, etc., *may not always agree with the local values on other processes.* This is in contrast to PPS, where all values except f_{trial} and x_{trial} are synchronized. While PPS relies on a global reduction operation to synchronize all critical values, APPS relies on nonblocking broadcasts to exchange information between processes. Descriptions of the individual steps of APPS follow. (The initialization for APPS is unchanged from that for PPS.) As we examine these steps, keep in mind that at each step, every process decides what to do next based only on its current *local* information.

Step 0: Checking for candidates from other processes. Before a process undertakes a new evaluation of the objective function, it considers any "new best" messages that may have arrived during the previous function evaluation. The receiving process considers each incoming triplet $\{x_+, f_+, \Delta_+\}$ as a candidate for a new best; hence the test in step 0(a). To make the procedure robust, we handle tie-breaking (i.e., the case where $f_+ = f_{\text{best}}$) in a consistent fashion, the details of which are deferred to section 3.3.2.

Step 1: Evaluating the function. Step 1 is the computational workhorse of PPS and is equivalent to the same step in synchronous PPS. The one substantive difference is that in PPS, x_{best} and Δ_{trial} are identical across all processes. In APPS, these values are no longer synchronized; they depend only on the information that is currently known to the process when it constructs $x_{\rm trial}$.

Step 2: Assigning the local candidate. This step does not actually require any action; it is here to emphasize that, in contrast to PPS, accepting the local trial point as a possible candidate for the new best does not involve the other processes. Instead, input from other processes is assessed in step 0, as it becomes available.

Step 3: Assessing the local candidate. If f_+, the function value at x_+, is better than $f_{\rm best}$, then $\Delta_{\rm trial}$ (and Δ_+) are increased (using the same strategy given in section 3.1 for PPS), the process's triplet $\{x_{\rm best}, f_{\rm best}, \Delta_{\rm best}\}$ is updated, and a message is broadcast to the other processes to inform them of this improvement. Otherwise, the process reduces $\Delta_{\rm trial}$ and continues.

Step 4: Checking for convergence along my search direction. There are two possible outcomes for step 3: either f_+ replaces $f_{\rm best}$ (in which case, $\Delta_{\rm trial}$ may be increased) or $f_{\rm best}$ is unchanged and $\Delta_{\rm trial}$ is reduced. If the second outcome occurs and $\Delta_{\rm trial} \leq tol$, this signals that no improvement can be found from the current $x_{\rm best}$ along the search direction d that this process owns and thus we may have arrived at a stationary point [9]. The process then notifies the other processes, by broadcasting a "single direction convergence" message, that it has converged (within tolerance) along its search direction.

Step 5: Waiting for a more complete picture of the entire search. The last step in APPS is the one step where a process may wait in an idle loop. Step 5 is reached only when a process has converged along its search direction. The idle process waits until either one of two things happens: it receives enough single direction convergence messages to verify global convergence to a stationary point of the objective function, or another process produces a point with a function value that is lower than $f_{\rm best}$. The details regarding what constitutes "enough" single direction convergence messages are deferred to section 3.3.3, where we discuss the precise measure of "enough" and how this can be determined.

3.3. Handling messages and exploiting parallelism. Now that we have discussed the essential logic of APPS, we change it slightly to better handle the message traffic and to better exploit parallelism.

There are technical considerations underlying the implementation of APPS that cause us to modify the algorithm slightly from the version presented in Figure 3.2. In particular, in the discussion above we have referred to a set of p processes, each of which handles computation, communication, and decision making. However, it is convenient to split the computation (i.e., the evaluation of the objective function) from the communication and decision making. One motivation for spawning a separate process to handle each evaluation of the objective function is that as a consequence of receiving a new best point from another process, it may be desirable to terminate an evaluation at some $x_{\rm trial}$ in order to move to the search to the new $x_{\rm best}$. A second motivation is that it should eliminate the accumulation of a large number of unprocessed messages, which can cause the message queue to overflow.

We start with a group of APPS *agent* processes that are in charge of the communication and decision making. Each evaluation of $f(x_{\rm trial})$ is spawned as a separate process that is subservient to a single APPS agent. The result is a set of APPS agents working in peer-to-peer mode, with each APPS agent spawning function evaluation processes as necessary. In contrast with the description of APPS given in Figure 3.2, APPS agents now dynamically initiate actions solely in response to messages, rather than routinely cycling through a fixed set of steps. It should be noted that the APPS agents require very little processing time, relative to the amount of time devoted to evaluating $f(x_{\text{trial}})$. Essentially, each APPS agent lies dormant until the arrival of an incoming message, which then triggers some action.

The types of incoming messages that an APPS agent receives are categorized as follows: a "return" from the process it spawned for the evaluation of $f(x_{\text{trial}})$, a "new best" message from another APPS agent, a "single direction convergence" message from another APPS agent, or a "shutdown" message from another APPS agent. We now investigate in more detail an APPS agent's reaction to each type of incoming message.

3.3.1. Handling "return" messages. An APPS agent receives a "return" message when the process it spawned to evaluate $f(x_{\text{trial}})$ returns the computed value f_{trial}. In Figure 3.3 we show an APPS agent's actions in response. In the discussions that follow, we introduce here an additional item to be associated with each point—a *convergence table* Π. The convergence table is used to detect a stationary point. It lists which of the p search directions from x have converged to within tolerance. We defer a further discussion of how this information is processed to section 3.3.3, where we discuss an APPS agent's action in response to a "single direction convergence" message in more detail.

```
Return from evaluation of the objective. Receive $f_{\text{trial}}$.
    1. Update $x_{\text{best}}$ and/or $\Delta_{\text{trial}}$.
        (a) If $f_{\text{trial}} < f_{\text{best}}$, then
                i. $\Delta_{\text{trial}} \leftarrow \lambda\, \Delta_{\text{trial}}$ with $\lambda \in \{2^\ell \mid \ell \in \mathbb{Z}_+\}$,
               ii. $\{x_{\text{best}}, f_{\text{best}}, \Delta_{\text{best}}, \Pi_{\text{best}}\} \leftarrow \{x_{\text{trial}}, f_{\text{trial}}, \Delta_{\text{trial}}, \Pi_{\text{trial}}\}$, and
              iii. broadcast a nonblocking new best message with the quadruple
                   $\{x_{\text{best}}, f_{\text{best}}, \Delta_{\text{best}}, \Pi_{\text{best}}\}$.
        (b) Else if $x_{\text{best}}$ is not the point used to generate $x_{\text{trial}}$, then $\Delta_{\text{trial}} \leftarrow \Delta_{\text{best}}$.
        (c) Else $\Delta_{\text{trial}} \leftarrow \frac{1}{2}\Delta_{\text{trial}}$.
    2. Check for convergence and spawn next objective function evaluation.
        (a) If $\Delta_{\text{trial}} > tol$, then compute $x_{\text{trial}} \leftarrow x_{\text{best}} + \Delta_{\text{trial}}\, d$, initialize $\Pi_{\text{trial}}$ to
            FALSE, and spawn a new process to evaluate $f(x_{\text{trial}})$.
        (b) Else update $\Pi_{\text{best}}$ (to signal convergence to $x_{\text{best}}$ along my direction $d$) and
            broadcast a nonblocking single direction convergence message with the
            quadruple $\{x_{\text{best}}, f_{\text{best}}, \Delta_{\text{best}}, \Pi_{\text{best}}\}$.
```

Fig. 3.3. *APPS agent's response to a return message.*

After receiving a return message, an APPS agent first must determine if a new best point has been identified, as shown in step 1(a). If so, the steplength Δ_{trial} may be increased (using the same rule as for PPS, given in section 3.1) and $\{x_{\text{trial}}, f_{\text{trial}}, \Delta_{\text{trial}}, \Pi_{\text{trial}}\}$ replaces $\{x_{\text{best}}, f_{\text{best}}, \Delta_{\text{best}}, \Pi_{\text{best}}\}$. The improvement is broadcast to all other APPS agents.

Upon first inspection, the need for step 1(b) may not be clear. An APPS agent constructs x_{trial} using its current values of x_{best} and Δ_{trial} (step 2(a) in Figure 3.3). While the process spawned by an APPS agent is busy evaluating $f(x_{\text{trial}})$, there is always the chance that another APPS agent will broadcast a quadruple $\{x_+, f_+, \Delta_+, \Pi_+\}$ whose value of f_+ improves upon the resident value of f_{best}. As we shall see in section 3.3.2, when an APPS agent receives such an incoming message, it replaces $\{x_{\text{best}}, f_{\text{best}}, \Delta_{\text{best}}, \Pi_{\text{best}}\}$ with $\{x_+, f_+, \Delta_+, \Pi_+\}$. Before constructing the next x_{trial}, the APPS agent ascertains if, while $f(x_{\text{trial}})$ was being computed, x_{best} was replaced as the result of an incoming message from another APPS agent. If so, then this new x_{best} will be used to compute the next x_{trial}. In this case, the APPS agent replaces Δ_{trial} with Δ_{best}; using the value of Δ associated with the best point retains some scaling information.

In step 1(c), we halve Δ_{trial} after confirming that f_{best} has not been replaced either by f_{trial} or by some f_+ contained in a message that arrived from another APPS agent while $f(x_{\text{trial}})$ was being computed.

Step 2 in Figure 3.3 checks the value of Δ_{trial}, our measure of progress toward a solution, and reacts appropriately. If Δ_{trial} is greater than *tol*, we continue the search. Otherwise, when $\Delta_{\text{trial}} \leq tol$, the search along the APPS agent's direction d has converged to x_{best}, and this information needs to be broadcast to all APPS agents.

3.3.2. Handling "new best" messages. When an APPS agent receives a "new best" message from another APPS agent, the first thing to check is whether or not the incoming quadruple really does contain the best function value seen thus far.

Here we encounter an important caveat of heterogeneous computing [2]. The comparison of floating-point values (in particular, f's and Δ's) controls the flow of APPS and we depend on these comparisons to give consistent results across all processes. Therefore, we must ensure that values are compared only to a level of precision available on all processors. In other words, a "safe" comparison declares a equivalent to b if

$$\frac{|\,a-b\,|}{\max\{|a|,|b|\}} < \epsilon^*_{\text{mach}}, \tag{3.1}$$

where ϵ^*_{mach} is greater than or equal to the maximum value of machine epsilon across the values for machine epsilon on all processors participating in the computation. If both $|a|$ and $|b|$ are below ϵ^*_{mach}, then they are automatically considered equal and (3.1) is not evaluated.

The second concern raised by the concurrency of the processes is what to do when f_+ and f_{best} are equivalent. Currently, APPS uses the following tie-breaking scheme. If f_+ and f_{best} satisfy (3.1), then compare Δ_+ and Δ_{best} and select the candidate with the larger value of Δ. If Δ_+ and Δ_{best} also satisfy (3.1), check next to see if x_+ and x_{best} are the same. Rather than comparing x_+ and x_{best} directly, by computing some norm of the difference, we use a unique global identifier with which APPS tags each point. Thus, two points are considered equivalent if and only if their f-values, Δ-values, and unique global identifiers are equivalent. This means that two points that actually are equal, but were generated via different paths on different processes, will be considered to be "different" points since their global identifiers do not match. However, since the purpose of the identification is to break ties in a consistent fashion, all we need worry about is what to do when both the f-values and the Δ-values are equivalent but the global identifier is not. In this last case, ties are broken in favor of the point with the lower global identifier. Since the global identifier of each point is a unique integer, the resolution is unambiguous. So, whenever we compare f_+ and f_{best}, the comparison incorporates this tie-breaking strategy.

Now that we can assess "improvement" on f_{best} in a way that both handles the vagaries of floating-point representation and breaks ties in a consistent fashion, we examine in more detail an APPS agent's actions to a "new best" message, shown in Figure 3.4.

```
New best. Receive $\{x_+, f_+, \Delta_+, \Pi_+\}$.
    1. If $f_+ < f_{\text{best}}$, then
        (a) If I had converged to $x_{\text{best}}$ along my search direction $d$, then
            i. $\{x_{\text{best}}, f_{\text{best}}, \Delta_{\text{best}}, \Pi_{\text{best}}\} \leftarrow \{x_+, f_+, \Delta_+, \Pi_+\}$, $\Delta_{\text{trial}} \leftarrow \Delta_{\text{best}}$,
            ii. compute $x_{\text{trial}} \leftarrow x_{\text{best}} + \Delta_{\text{trial}}\, d$, initialize $\Pi_{\text{trial}}$ to all FALSE, and
                spawn a new process to evaluate $f(x_{\text{trial}})$.
        (b) Else if $\Delta_{\text{trial}} < \Delta_+$, then
            i. terminate the process evaluating $f(x_{\text{trial}})$,
            ii. $\{x_{\text{best}}, f_{\text{best}}, \Delta_{\text{best}}, \Pi_{\text{best}}\} \leftarrow \{x_+, f_+, \Delta_+, \Pi_+\}$, $\Delta_{\text{trial}} \leftarrow \Delta_{\text{best}}$,
            iii. compute $x_{\text{trial}} \leftarrow x_{\text{best}} + \Delta_{\text{trial}}\, d$, initialize $\Pi_{\text{trial}}$ to all FALSE, and
                spawn a new process to evaluate $f(x_{\text{trial}})$.
        (c) Else, $\{x_{\text{best}}, f_{\text{best}}, \Delta_{\text{best}}, \Pi_{\text{best}}\} \leftarrow \{x_+, f_+, \Delta_+, \Pi_+\}$.
    2. Else discard $\{x_+, f_+, \Delta_+, \Pi_+\}$.
```

FIG. 3.4. *APPS agent's response to a new best message.*

Assuming improvement on f_{best}, the first action taken by an APPS agent is to determine the status of the search along the direction d. There are three possibilities to consider.

The first possibility, shown in step 1(a), is that at some point the search along d had converged within tolerance and so the APPS agent is now waiting for incoming messages to either confirm overall convergence of the search or, as in this case, produce a new best point (see step 5 in Figure 3.2). When the latter occurs, the incoming quadruple is accepted and the search is resumed from the new x_{best}.

The second possibility, shown in step 1(b), is that the search along d is still in progress, but that the steps along d have become small, i.e., $\Delta_{\text{trial}} < \Delta_{\text{best}}$. If so, then the search along d has reduced Δ_{trial}—perhaps repeatedly—in an effort to find improvement on f_{best}. In this case, it is particularly useful to have an APPS agent acting independently of the function evaluation process. An APPS agent can terminate the current evaluation of $f(x_{\text{trial}})$ before it actually finishes (step 1(b)i) in favor of starting a new evaluation of the objective based on a new value of x_{best} (step 1(b)iii). The question to ask is why we would choose to do so.

In certain cases, the current evaluation of the objective function is terminated in favor of starting one based on a new best point. Imagine the following scenario. Suppose three APPS agents, A, B, and C, start off with the same value for x_{best}, generate their own x_{trial}'s, and spawn their own evaluations of $f(x_{\text{trial}})$. Each evaluation of the objective function takes several hours. The evaluation for Agent A completes first and there is no improvement, so Agent A reduces its steplength, generates a new trial point, and spawns a new evaluation of the objective function. A few minutes later, Agent B's evaluation finishes and it produces improvement. Agent B broadcasts a "new best" message to the other APPS agents. Agent A receives this message and terminates its current evaluation of the objective function in order to move to the better point. This may save several hours of wasted computing time. However, Agent C, which is still working on its first evaluation of the objective function, waits for that to complete before considering a move to the new x_{best} because the inequality on Δ_{trial} does not hold in step 1(b) of Figure 3.4.

The third possibility when the incoming value of f_+ improves upon the local value of f_{best} is to simply accept the incoming quadruple, as shown in step 1(c). This is exactly the strategy for Agent C outlined in the scenario described above.

The final observation to be made is that if f_+ does not improve upon f_{best}, the

```
Single direction convergence. Receive ${x_+, f_+, \Delta_+, \Pi_+}$.
    1. If $f_+ < f_{\text{best}}$, then go through the steps for a new best message.
    2. If $f_+ = f_{\text{best}}$, then
        (a) Update $\Pi_{\text{best}}$ to include any new information contained in $\Pi_+$.
        (b) If I am the temporary master, then check for convergence.
            If enough of the other processes have converged (i.e., their associated
            directions form a positive spanning set), then
              i. report ${x_{\text{best}}, f_{\text{best}}}$,
             ii. broadcast a nonblocking shutdown message to the remaining APPS
                 agents, and
            iii. exit.
    3. Else discard ${x_+, f_+, \Delta_+, \Pi_+}$.
```

FIG. 3.5. *APPS agent's response to a single direction convergence message.*

quadruple $\{x_+, f_+, \Delta_+, \Pi_+\}$ is simply discarded; it already has been superseded by another point and thus is of no interest.

3.3.3. Handling "single direction convergence" messages. Detecting convergence for APPS is a trickier issue than it is for PPS because the APPS agents do not perform a synchronized test for convergence. Instead, each APPS agent stops spawning processes to evaluate $f(x_{\text{trial}})$ when its local value of Δ_{trial} satisfies $\Delta_{\text{trial}} \leq tol$. Any APPS agent that arrives at this conclusion then waits until either enough other APPS agents stop at the same best point (we describe "enough" below) or another APPS agent produces a better point from which to resume the search. Since every quadruple $\{x_{\text{trial}}, f_{\text{trial}}, \Delta_{\text{trial}}, \Pi_{\text{trial}}\}$ which improves upon f_{best} is broadcast to all APPS agents, every APPS agent eventually agrees on the best point.

When an APPS agent receives a "single direction convergence" message (see Figure 3.5), it checks to make sure that this function value and associated point have been seen before. If not (a distinct possibility since messages may arrive out of order), then the APPS agent handles the incoming quadruple as if it were part of a "new best" message.

If the incoming point is the same as the best point we have, i.e., $f_+ = f_{\text{best}}$, then the APPS agent receiving the message must update its convergence table Π_{best} to include any new information regarding the convergence of other search directions to the same point x_{best}. Again, timing issues must be taken into account as either the sending or the receiving APPS agent may have information that has not yet been seen by the other.

Next, in order to check for convergence of a sufficient number of the p independent search directions, it is useful to have a *temporary master* to avoid redundant computation. We define the temporary master to be the APPS agent with the lowest process identification number. While this is usually process 0, it is not necessarily the case if a fault occurs; we discuss this scenario further in section 4. The temporary master checks to see if the set of directions along which the search has converged forms a positive spanning set. If so, it reports to the user the final result of the search, broadcasts a "shutdown" message, and exits.

Checking for a positive spanning set can be done as follows. We know that a positive spanning set for $\mathbb{R}^n$ must contain at least $n + 1$ vectors [6]. So if the convergence table has at least $n + 1$ entries, it is time for the temporary master to check for convergence of the overall process. (Every APPS agent knows $\mathcal{D}$, which is why it is possible for any APPS agent to serve as temporary master.) Let $\mathcal{V} \subseteq \mathcal{D}$ be the candidate for a positive spanning set. We solve $n + 1$ nonnegative least squares problems according to the following theorem.

THEOREM 3.1. *A set $\mathcal{V} = \{v_1, v_2, \ldots, v_m\}$ is a positive spanning set if the set $\mathcal{E} = \{e_1, e_2, \ldots, e_n, -\mathbf{1}\}$ is in its positive span (where $-\mathbf{1}$ is the vector of all -1's).*

Alternatively, we can check the positive basis by first verifying that $\mathcal{V}$ is a spanning set using, say, a QR factorization with pivoting, and then solving a linear program.

THEOREM 3.2 (see Wright [27]). *A spanning set $\mathcal{V} = \{v_1, v_2, \ldots, v_m\}$ is a positive spanning set if the maximum of the following linear program is 1.*

$$\max\ t \quad \text{subject to } Vx = 0,\ x_i \geq t\ \forall\, i,\ 0 \leq t \leq 1,$$

where V is a matrix representing the spanning set $\mathcal{V}$.

We make use of Theorem 3.1 since Netlib provides freely available software, due to Lawson and Hanson [14], for solving nonnegative least squares problems. To make use of Theorem 3.2 requires software both for QR factorizations and for the solution of linear programs; the latter is particularly difficult to come by in a freely available, portable, and easy-to-use format.

3.3.4. Handling "shutdown" messages. The reactions of the other APPS agents to a "shutdown" message from the temporary master should be clear after the discussion in section 3.3.3; they are given in Figure 3.6. Again we note the value of having both an APPS agent and a separate process for evaluating $f(x_{\text{trial}})$; once the shutdown message has been received, an APPS agent can immediately terminate the process evaluating $f(x_{\text{trial}})$ and exit.

Shutdown. Receive the shutdown message from the temporary master.
1. *Terminate* the process evaluating $f(x_{\text{trial}})$ and
2. exit.

FIG. 3.6. *APPS agent's response to a shutdown message.*

4. Fault tolerance in APPS.

The move toward a variety of computing environments, including heterogeneous distributed computing platforms, brings with it increased attention to the fault tolerance of parallel algorithms. The large size, diversity of components, and complex architecture of such systems create numerous opportunities for hardware failures, and our computational experience confirms that these failures do, in fact, occur.

In addition, the size and complexity of current simulation codes call into question the robustness of the function evaluations. For example, our experience has been that it is possible to generate input parameters that are both physically and mathematically feasible but for which the simulation codes fail to finish successfully. Thus, we must contend with software failures as well as hardware failures.

A great deal of work has been done in the computer science community with regard to fault tolerance; however, much of that work has focused on making fault tolerance as transparent to the user as possible. This often entails strategies such as checkpointing the entire state of an application to disk or replicating processes. Fault tolerance has traditionally been used with loosely coupled distributed applications that do not depend on each other to finish, such as business database applications. This lack of interdependence is atypical of most scientific applications. While checkpointing and replication are adequate techniques for scientific applications, they incur a substantial amount of unwanted overhead; however, certain scientific applications have characteristics that can be exploited to derive more efficient and elegant stratagems for fault tolerance. Algorithm-dependent strategies for incorporating fault tolerance have already received attention in the scientific computing community; see, e.g., [21]. These approaches rely primarily on the use of diskless checkpointing, a significant improvement over traditional approaches. The nature of APPS is such that we can even further reduce the overhead for fault tolerance and dispense with checkpointing altogether.

Two important observations should be made regarding fault tolerance in APPS. First, there are no single points of failure in the APPS algorithm itself. Assuming initialization is successful, there is just one scenario that requires a single APPS agent to coordinate efforts among all agents (i.e., the *temporary master* used to check convergence of the entire search, as shown in Figure 3.5). However, the choice of master is not fixed. If the APPS agent serving as temporary master should fail while performing its tasks, another APPS agent steps up to take over. This means the degree of fault tolerance in APPS is constrained only by the underlying communication architecture. The current implementation of APPS uses PVM [11], which provides a rich library of communication and process management procedures needed by the APPS agents. The one limitation we inherit from PVM is that it executes multiple processes on multiple processors under the control of a single master PVM daemon. Thus the PVM daemon introduces a single point of failure within our current implementation of APPS. We expect HARNESS [1], the successor to PVM, to eliminate this disadvantage. The second observation to be made is that no checkpointing or replication of processes is necessary. The APPS agents can be reconfigured dynamically. New APPS agents require only a small packet of information from any active APPS agent in order to take over where a failed APPS agent left off. Therefore, we have been able to take advantage of algorithmic characteristics of pattern search in order to incorporate a high degree of fault tolerance into APPS with almost no additional overhead.

Having made these two observations, we now describe how fault tolerance is addressed in APPS. Every APPS agent keeps a record of active and inactive APPS agents (one per search direction), the available hosts, and a mapping of the active APPS agents to the available hosts. There are three types of faults with which we are concerned: (1) the failure of a process evaluating the objective function, (2) the failure of an APPS agent, and (3) the failure of a host processor. Once again we note the advantage of maintaining pairs of processes: an APPS agent to handle all communication (including information from PVM regarding the failure of processes) that is separate from the processes tasked with the major computations, the evaluations of the objective function. An individual APPS agent uses its record of active and inactive APPS agents to decide whether or not it is the temporary master and to determine the other APPS agents with whom it should interact in response to a failure. The responses to these three scenarios are shown in Figure 4.1.

When a process evaluating $f(x_{\text{trial}})$ fails, the failure is reported to its master (i.e., the APPS agent that originally spawned it), and that APPS agent respawns the evaluation of the objective function at the current trial point. If several (e.g., five) attempts to evaluate the objective function fail at the same trial point, the APPS agent that was spawning those evaluations exits, triggering an APPS agent failure message to be sent to the other APPS agents. The failure of an evaluation could be handled in different ways for different applications; for instance, attempts to evaluate

- **An APPS agent detects failure of its function evaluation process.**
  1. If the number of attempts to evaluate $f(x_{\mathrm{trial}})$ is less than the maximum number allowed, then respawn a new process to evaluate $f(x_{\mathrm{trial}})$.
  2. Else exit.
- **An APPS agent detects failure of another APPS agent.**
  1. Record the failure.
  2. If "my" process number is the lowest among the APPS agents still active, then assume the responsibility of *temporary master*.
  3. If I am the *temporary master*, then
     - (a) Check for convergence. If enough of the other APPS agents report convergence (i.e., their associated directions form a positive spanning set), then
       - i. report $\{x_{\mathrm{best}}, f_{\mathrm{best}}\}$,
       - ii. *broadcast* a nonblocking **shutdown** message to the remaining APPS agents, and
       - iii. exit.
     - (b) If the directions corresponding to the remaining APPS agents do not form a positive spanning set, respawn all failed APPS agents on available host processors.
- **An APPS agent detects failure of a host processor.**
  1. Remove failed host from list of available host processors.
  2. Determine all APPS agents residing on the failed host processor and treat each as a failed APPS agent.

FIG. 4.1. *Fault tolerance messages and actions.*

the objective function at a certain point could be abandoned without necessarily terminating the APPS agent.

When an APPS agent fails, all the remaining APPS agents record this failure. If the APPS agent that failed happened to be serving the role of temporary master, then another APPS agent must assume this responsibility. We maintain the convention that the active APPS agent with the lowest process number serves as temporary master. Once the question of who is temporary master is resolved, the first thing the new temporary master does is check for convergence since the now defunct APPS agent may have been in the midst of that check when it failed. If the search has not yet converged, the temporary master checks whether or not the set of directions owned by the remaining active APPS agents forms a positive spanning set. If so, then it is still possible to reliably determine whether or not the algorithm has converged, so nothing is done. Otherwise, all defunct APPS agents are restarted on the available hosts by the temporary master. Note that multiple APPS agents may be assigned to a single host.

If a host fails, the defunct host processor is removed from the list of viable hosts. The APPS agents that were running on the defunct host are regarded individually as failed APPS agents, which are then handled using the rules stated for APPS agent failures.

Despite the growing attention to fault tolerance in the parallel computing world, we are aware of only one other parallel optimization algorithm that incorporates fault tolerance, FATCOP [3]. FATCOP is a parallel mixed integer program solver that has been implemented using a Condor-PVM hybrid as the communication substrate. FATCOP is implemented in a master-slave fashion, which means that there is a single point of failure at the master process. This is addressed by having the master checkpoint information to disk (via Condor), but recovery requires user intervention to restart the program in the event of a failure. In contrast, once APPS has finished initialization, it can recover from the failure of any process of its own creation, including the failure of the temporary master. It does so on its own, with no checkpointing whatsoever.

5. Numerical results.

We compare APPS and PPS on several test problems as well as two engineering problems: a thermal design problem and a circuit simulation problem.

The tests were performed on the CPlant supercomputer at Sandia National Labs in Livermore, CA. CPlant is a cluster of DEC Alpha Miata 433 MHz processors. For our tests, we used 50 processors dedicated to our sole use.

5.1. Standard test problems.

We compare APPS and PPS with 8, 16, 24, and 32 processors on six four-dimensional test problems [20, 5], shown in Table 5.1.

Table 5.1
Six standard test problems.

1	2	3	4	5	6
`broyden2a`	`broyden2b`	`chebyquad`	`epowell`	`toint_trig`	`vardim`

Since the function evaluations are extremely fast, we added extra "busy work" (in the form of solving a 100×101 nonnegative least squares problem) in order to slow down the processes evaluating f and better simulate the computational behavior of the optimization problems in which we are interested.

The parameters for APPS and PPS were set as follows. Let $n = 4$ be the problem dimension, and let $p \in \{8, 16, 24, 32\}$ be the number of processors. The first $2n$ search directions in $\mathcal{D}$ are $\{e_1, e_2, \ldots, e_n, -e_1, -e_2, \ldots, -e_n\}$. The remaining $p-2n$ directions are generated randomly (with a different seed for every run) and normalized to unit length. This construction ensures that $\mathcal{D}$ is a positive spanning set. We initialize $\Delta = 1.0$ and $tol = 0.001$. We start each of these six problems from the standard starting point [20, 5].

Before considering the summary results, we discuss the details of two sample runs (one each for APPS and PPS) given in Table 5.2. Each process reports its own counts and timings. All times are reported in seconds and are wall clock times. Because APPS is asynchronous, the number of function evaluations spawned by each APPS agent varies considerably. Furthermore, the APPS agents sometimes terminate ("break") processes evaluating $f(x_{\text{trial}})$. On the other hand, because PPS is synchronous, every process executes the same number of function evaluations and there are no breaks. For both APPS and PPS, the initialization time is longer for Process 0 since it is in charge of spawning all remaining tasks. The idle time varies from process to process, but is overall lower for APPS than PPS. An APPS agent is idle only when it has converged along its search direction, but a PPS process may potentially have some idle time every iteration while it waits for the completion of the global reduction. The total wall clock time varies from process to process since each starts and stops at slightly different times. The summary information reports the mean over all eight processes, except in the case of total time, where the *maximum* total time over all eight processes is reported.

Because some of the search directions are generated randomly, every run of APPS and PPS follows a different path to the solution and generates possibly different solutions in the case of multiple minima. (The exception is PPS with $p = 8$. Because there are no "extra" search directions, the path to the solution is the same for every run—only the timings differ. The nondeterministic nature of APPS causes us to see

Table 5.2
Detailed results for `epowell` *on eight processors.*

Method	Process	Function	Function	Init	Idle	Total
	ID	evals	breaks	time	time	time
APPS	0	237	66	0.17	0.00	24.72
	1	266	70	0.02	0.12	22.36
	2	302	89	0.02	0.12	24.32
	3	274	77	0.02	0.15	22.31
	4	270	62	0.02	0.04	24.56
	5	282	81	0.02	0.04	24.58
	6	273	59	0.02	0.04	24.59
	7	276	61	0.02	0.03	24.55
Summary statistics		272.5	70.6	0.04	0.07	24.72
PPS	0	235	0	0.74	2.55	30.63
	1	235	0	0.39	7.23	30.28
	2	235	0	0.25	6.74	30.14
	3	235	0	0.13	6.94	30.01
	4	235	0	0.10	6.36	29.98
	5	235	0	0.07	6.51	29.95
	6	235	0	0.04	6.23	29.92
	7	235	0	0.02	6.26	29.90
Summary statistics		235	N/A	0.22	6.10	30.63

different counts and different timings for every run, even if the search directions for each run are identical.) Therefore, for each problem in Table 5.1 we report the mean of the summary statistics from 25 runs; for each *individual* run we collected the same summary statistics (except the initialization time) reported in Table 5.2.

The test results are summarized in Table 5.3. These tests were executed in what should have been a particularly favorable environment for PPS—a cluster of homogeneous, dedicated processors. The primary difficulty for PPS is the cost of synchronization in the global reduction. In terms of average function evaluations per processor, APPS and PPS typically required about the same number. In general, for both APPS and PPS, the number of function evaluations per processor decreased as the number of processes increased. We expected the idle time for APPS to be less than that for PPS; and, indeed, the idle time is two orders of magnitude less. Furthermore, the idle time for PPS increases as the number of processors goes up. APPS was faster (on average) than PPS in 23 out of 24 cases. The total time (on average) for APPS either stayed more or less steady or actually decreased as the number of processors increased. In contrast, the total time (on average) for PPS almost always increased as the number of processors increased, due to the synchronization penalty incurred with the addition of more processes.

Comparing APPS and PPS on simple problems is not necessarily indicative of results for typical engineering problems. The results in sections 5.2 and 5.3 yield more meaningful comparisons, given the types of problems for which pattern search is best suited.

5.2. TWAFER: A thermal design problem. In this set of tests, the engineering application is an optimal control problem for a thermal deposition furnace for silicon wafers. The furnace contains a vertical stack of wafers and several heater zones. The goal is to choose power settings for the heaters in each of n zones to achieve a prescribed constant temperature across each wafer and throughout the stack. The simulation code, TWAFER [12], yields measurements at a discrete collection of points

Table 5.3
Summary statistics (across 25 *runs) for the four-dimensional test problems shown in Table* 5.1.

Prob	No.	Function evals		APPS	Idle time		Total time	
no.	procs	APPS	PPS	breaks	APPS	PPS	APPS	PPS
1	8	40.59	37.00	8.14	0.07	0.95	3.88	4.88
	16	41.77	40.12	7.93	0.02	2.04	3.98	6.68
	24	38.30	37.36	6.98	0.02	4.68	3.80	9.33
	32	36.57	37.92	6.88	0.03	7.81	3.83	12.81
2	8	40.35	37.00	8.28	0.06	0.97	3.84	4.92
	16	41.07	39.11	7.38	0.02	2.06	3.95	6.62
	24	38.47	39.60	7.20	0.02	4.77	3.77	9.68
	32	35.10	36.76	6.23	0.03	7.04	3.72	11.92
3	8	73.06	62.00	16.74	0.05	1.61	6.86	8.11
	16	48.33	40.44	9.54	0.02	2.11	4.69	6.92
	24	45.67	38.64	9.26	0.02	4.59	4.47	9.39
	32	44.34	37.60	9.14	0.04	7.54	4.59	12.56
4	8	272.29	235.00	68.27	0.30	6.64	24.50	30.48
	16	139.63	153.04	37.39	0.05	8.04	12.24	24.76
	24	139.38	126.96	36.40	0.03	14.10	12.26	28.46
	32	98.88	102.64	26.20	0.03	28.07	9.41	41.03
5	8	53.83	41.00	10.97	0.04	1.11	4.99	5.60
	16	51.40	39.12	10.47	0.02	1.97	4.91	6.51
	24	47.86	36.88	9.24	0.02	4.43	4.69	9.03
	32	45.90	33.04	8.70	0.04	6.41	4.81	10.83
6	8	205.39	77.00	51.24	0.05	2.00	18.15	9.97
	16	101.46	80.44	25.58	0.02	3.97	8.93	12.83
	24	72.44	49.96	17.19	0.02	5.61	6.57	11.63
	32	64.09	46.04	15.96	0.03	9.58	6.14	15.51

on the wafers. The objective function f is defined as

$$f(x) = \sum_{j=1}^{N} \left(T_j(x) - T_*\right)^2 , \tag{5.1}$$

where N is the number of discrete temperature measurement points, $T_j(x)$ is the simulated temperature at the jth point for the power settings defined by x, and T_* is the prescribed ideal temperature.

We consider the four- and seven-zone (or variable) problems with $N = 40$ and $N = 400$, respectively. For the four-zone problem, the initial guess produced a function value of 2.26×10^6. The initial guess for the seven-zone problem produced a function value of 7.43×10^4. (The initial guess for the seven-zone problem was much closer to the final solution.)

We used the following settings for APPS and PPS. The first $n+1$ search directions are the points of a regular simplex centered about the origin. The remaining $p-n-1$ points are generated randomly and normalized to unit length. Because the magnitude of the variables was $O(100)$, we set $\Delta = 10.0$. Note that it can be quite useful to choose the initial Δ based on the magnitudes of the components in x_0 as a way to capture some scaling information about the problem [25]. We chose $tol = 0.1$, which corresponds to a level of accuracy that is reasonable in the power settings.

There are some difficulties from the implementation point of view that are quite common when dealing with simulation codes. Because TWAFER is a legacy code, it expects an input file with a specific name and produces an output file with a specific name. The names of these files cannot be changed, and TWAFER cannot be hooked directly to PVM. As a consequence, we must write a "wrapper" program that runs an input filter, executes TWAFER via a system call, and runs an output filter. Because TWAFER is executed via a system call, APPS has no way of terminating its execution prematurely. (APPS can terminate the wrapper program, but TWAFER itself will continue to run, consuming system resources.) Therefore, we allow all function evaluations to run to completion; that is, we do not allow any breaks.

Another feature of TWAFER is that there are nonnegativity constraints on the power settings. The solution is known to be strictly positive, and the constraints play only a minor role in finding the solution. We did not invoke TWAFER at any point that had one or more negative components; to accomplish this, we use a simple barrier function that returns a large value (e.g., 10^{50}). This is a classic trick used by direct search methods for dealing with bound constraints. With the correct choice of $\mathcal{D}$, pattern search methods that use such a strategy can be shown to have at least one subsequence of iterates that converge to a Karush–Kuhn–Tucker point [18, 19].

Table 5.4
Summary statistics (across multiple runs) for the four- and seven-zone TWAFER problems.

Problem	Method	Procs	$f(x*)$	Function evals	Idle time	Total time
4 Zone	APPS	20	0.67	334.6	0.17	395.94
4 Zone	PPS	20	0.66	379.9	44.77	503.88
7 Zone	APPS	35	3.30	240.4	71.48	2260.46
7 Zone	PPS	35	2.85	202.2	213.90	2306.83

Results for the TWAFER problem are given in Table 5.4. The four-zone results report the means across all twenty processors over all ten runs. The seven-zone results report the means across all 35 processors over all nine runs. (We started ten runs for the seven-zone problem. One of the ten PPS runs failed due to a processor fault. One of the ten APPS runs experienced several faults and, although it did get the final solution, the summary data was incomplete.)

Recall that the goal is to choose power settings to achieve a constant temperature across each wafer and throughout the stack. In Figure 5.1 we show the temperatures computed by TWAFER at each wafer along a line of discretization points from the bottom to the top of the furnace. We show results for both the initial settings we were given for the seven-zone problem and the best and worst settings returned by APPS, corresponding to function values of 1.48 and 7.74, respectively. (The plots of the results from the best and worst PPS solutions are indistinguishable from the best and worst plots for APPS.) Table 5.4 shows that for this problem, on average, PPS yields slightly better function values than APPS (less than 1/1000th of a percent relative difference compared to the function value at the starting point) but required more total time. Figure 5.1 demonstrates that, qualitatively, all the solutions produced were comparable, particularly given the modest choice of $tol = 0.1$.

Clearly, the idle time figures prominently in the overall performance of PPS. The average simulation time is 1.3 seconds for the four-zone problems and 10.4 seconds for the seven-zone problem. However, when the nonnegativity constraints are violated, TWAFER is not called, so the execution time is essentially zero since we simply return 10^{50} after checking the coordinates of x_{trial}. The relatively high mean idle time for APPS (for the seven-zone problem) can be traced to a single run for which the idle time was particularly high for some processors (634 seconds on average across all 35 processors); on the remaining runs, the average APPS idle time per processor was

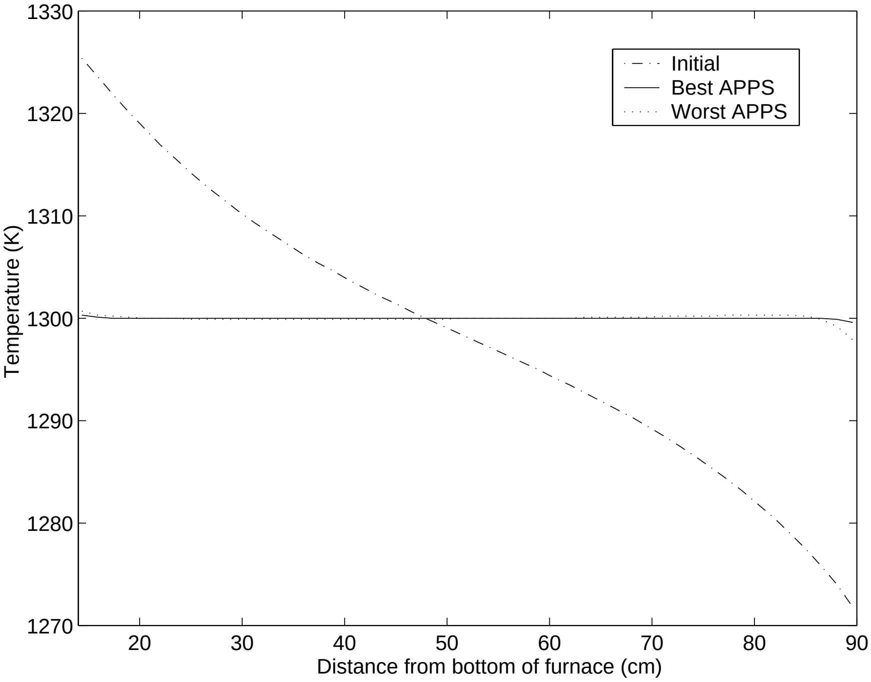


FIG. 5.1. *TWAFER results for the seven-zone problem using APPS with tol* = 0.1. *The solid line represents the simulation output for the best settings found by APPS and the dotted line represents the simulation output for the worst settings found by APPS. The dashed line represents the simulation output for the initial settings. The target is a constant temperature of* 1300.

lower by several orders of magnitude. We were unable to determine the cause of the unusually large idle time.

5.3. SPICE: A circuit simulation problem. The problem is to match simulation data to experimental data for a particular circuit in order to determine its characteristics. In our case, we have 17 variables representing inductances, capacitances, diode saturation currents, transistor gains, leakage inductances, and transformer core parameters. The objective function is defined as

$$f(x) = \sum_{j=1}^{N} \left(V_j^{\mathrm{SIM}}(x) - V_j^{\mathrm{EXP}}\right)^2, \tag{5.2}$$

where N is the number of time steps, $V_j^{\mathrm{SIM}}(x)$ is the simulation voltage at time step j for input x, and V_j^{EXP} is the experimental voltage at time step j.

The SPICE3 [22] package is used for the simulation. Like TWAFER, SPICE3 communicates via file input and output and so we again use a wrapper program.

The input filter for SPICE is more complicated than that for TWAFER because the variables for the problem are on different scales. Since APPS has no mechanism for scaling, we handled this within the input filter by computing an affine transformation of the variables used to formulate the objective function (5.2). Additionally, all the variables have upper and lower bounds. Once again, we use a simple barrier function.

The output filter for SPICE is also more complicated than that for TWAFER. The SPICE output files consist of voltages that are to be matched to the experimental data.

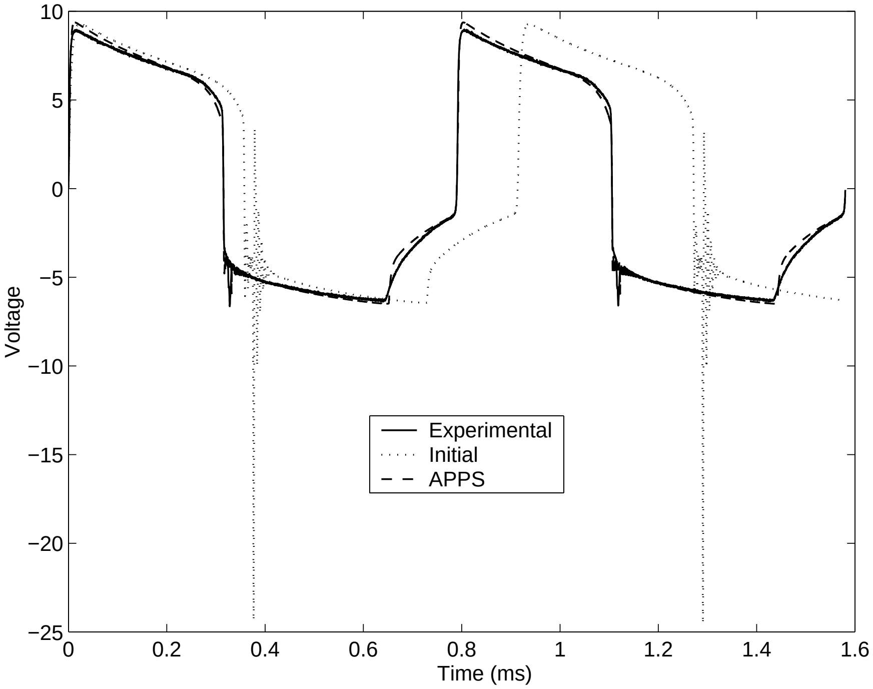


Fig. 5.2. *Spice results using APPS with tol* = 0.1*. The solid line represents the experimental output. The dashed line represents the simulation output after optimization. The dotted line represents the simulation output for the initial point.*

The experimental data is two cycles of output voltage measured at approximately $N = 2700$ discrete time steps (see Figure 5.2). The simulation data contains approximately 10 or more cycles, but only the last few complete cycles are used because the early cycles are not stable. The cycles must be automatically identified so that the data can be aligned with the experimental data. Furthermore, the time steps from the simulation may differ from the time steps in the experiment, and so the simulation data are interpolated (piecewise constant) to match the experimental data. The function value at the initial point is 465.

The APPS parameters were set as follows. The search directions were generated in the same way as those for the test problems in section 5.1. We set $\Delta = 1.0$ (the affine transformation means the variables are well scaled) and *tol* is 0.1 (the tolerance corresponds to a less than 1% change in the circuit parameter). Once again, we do not allow "breaks" since the function evaluation is called from a wrapper program via a system call.

The results from APPS and PPS on the SPICE problem are reported in Table 5.5. In this case, we are reporting the results of single runs; we give results for 34 and 50 processors. The average SPICE run time is approximately 20 seconds; however, once again we do not differentiate between times when the boundary conditions are violated and when the SPICE code is actually executed. Increasing the number of processors by 47% results in a 39% reduction in execution time for APPS but only 4% for PPS. For both 34 and 50 processors, APPS is faster than PPS and even produces a slightly better objective value (compared to the starting value of more than 400). At the solution, two constraints are binding.

TABLE 5.5
Results (one run each) for the 17 *variable SPICE problem.*

Method	Procs	$f(x*)$	Function evals	Idle time	Total time
APPS	34	26.3	57.5	111.92	1330.55
APPS	50	26.9	50.6	63.22	807.29
PPS	34	28.8	53.0	521.48	1712.24
PPS	50	34.9	47.0	905.48	1646.53

TABLE 5.6
APPS results for the 17 *variable SPICE with a failure approximately every* 30 *seconds.*

Initial procs	Final procs	$f(x*)$	Total time
34	34	27.8	1618.46
50	32	54.2	1041.14

Table 5.6 shows the results of running APPS with faults. In this case, we used a program that automatically killed one PVM process every 30 seconds. The PVM processes are the APPS agents and the wrapper programs. The SPICE3 simulation is executed via a system call, and so continues to execute even if its wrapper terminates; regardless, the SPICE3 program can no longer communicate with APPS and is effectively dead.

The results are quite good. In the case of 34 processors, every APPS task that fails must be restarted in order to maintain a positive basis. So, the final number of APPS processes is 34. The total time is only increased by 21% despite approximately 50 failures; furthermore, this time is still faster than PPS. In the case of 50 processors, the final number of processors is 32. (Recall that tasks are only restarted if there are not enough remaining to form a positive basis.) In the case of 50 processors, the solution time is only increased by 29% with faults, and is once again still faster than PPS. In this case, however, the quality of the solution is degraded. This is likely due to the fact that the solution lies on the boundary and some of the search directions that failed were needed to ensure convergence to a KKT point (see [18, 19]).

6. Conclusions. Our preliminary numerical results make clear that because APPS dynamically initiates actions solely in response to messages, it is a more effective method—even in a homogeneous cluster environment—than PPS, where "more effective" means that APPS requires less total time to return results that are comparable to those returned by PPS. We expect the differences to be even more pronounced for larger problems (where by "larger" we mean in terms of both the execution time and the number of variables) and for heterogeneous cluster computing environments. Unlike PPS, which routinely cycles through a fixed set of steps, APPS does not have any required synchronizations and, thus, appears to gain most of its advantage by reducing idle time.

Further, APPS is a fault-tolerant algorithm. We accomplish this by making algorithmic changes to PPS that introduce almost no additional overhead. As we saw in the results for the SPICE problem solved using 34 processors (section 5.3), APPS does not suffer much slow-down when faults do occur.

Finally, in forthcoming work, Kolda and Torczon [13] will show that in the unconstrained case, APPS is globally convergent (even when faults occur) under the standard assumptions for pattern search [16, 26].

These features duly noted, we are investigating further improvements to the implementation of APPS. For instance, in the implementation described here, each APPS agent is responsible for exactly one process to evaluate the objective function. For multiprocessor (MPP) compute nodes, this means there will be multiple agents per node. An alternative implementation of APPS is being developed in which there is exactly one agent per node, with the single agent managing multiple evaluations of the objective function. As part of this alternative implementation, the ability to dynamically add new hosts as they become available (or to re-add previously failed hosts) also will be incorporated.

Another improvement to the implementation will be the addition of a cache to store the values of the function at all the points visited by the search in order to avoid reevaluating the same point more than once. The challenges are to make the recovery of this information fast and to decide when two points are actually equal. The latter is especially difficult when we do not know the sensitivity of the function to changes in each variable.

The importance of positive bases in the pattern also raises several interesting questions. In general, we might consider the best way to generate the starting basis. The analysis of pattern search makes clear that the "conditioning" of the positive basis has an effect on the amount of decrease that may be realized [16]. Our numerical studies have indicated that the quality of the positive basis can, indeed, affect the progress of the search. Thus, explicitly monitoring the conditioning of the positive basis, which changes dynamically, could improve the overall performance of APPS. Further, supposing that enough failures have occurred so that there is no longer a positive basis, we may ask if we can easily determine the smallest number of vectors to add to once again have a positive basis. Our current implementation simply restarts all failed APPS agents (see Figure 4.1). In general, we desire a pattern that maximizes the probability of maintaining a well-conditioned positive basis in the event of failures, without requiring us to keep a large number of processes active when it is neither necessary nor convenient to do so.

Finally, although the engineering examples used in this work have bound constraints, the current version of APPS does not handle constraints in a rigorous fashion. The poor results on the SPICE problem with faults on 50 processors may well be attributed to this fact since several constraints are active at a known solution. The analysis for pattern search suggests several algorithmic options we could pursue [17, 18, 19], but the challenge is to do so in a way that works effectively within the asynchronous framework we have devised. Future work will explore robust extensions for handling constraints.

Acknowledgments. We thank Jim Kohl, Ken Marx, and Juan Meza for helpful comments and advice in the implementation of APPS and the test problems. We also thank the referees for their careful reading of an earlier draft of this manuscript. Their suggestions considerably improved the presentation of this work.

REFERENCES

[1] M. Beck, J. J. Dongarra, G. E. Fagg, G. A. Geist, P. Gray, J. Kohl, M. Migliardi, K. Moore, T. Moore, P. Papadopoulous, S. L. Scott, and V. Sunderam, *HARNESS: A next generation distributed virtual machine*, Future Generation Computer Systems, 15 (1999), pp. 571–582.

[2] L. S. Blackford, A. Cleary, A. Petitet, R. C. Whaley, J. Demmel, I. Dhillon, H. Ren, K. Stanley, J. Dongarra, and S. Hammarling, *Practical experience in the numerical dangers of heterogeneous computing*, ACM Trans. Math. Software, 23 (1997), pp. 133–147.

[3] Q. Chen and M. C. Ferris, *FATCOP: A fault tolerant Condor-PVM mixed integer programming solver*, SIAM J. Optim., 11 (2001), pp. 1019–1036.

[4] D. Conforti and R. Musmanno, *Convergence and numerical results for a parallel asynchronous quasi-Newton method*, J. Optim. Theory Appl., 84 (1995), pp. 293–310.

[5] A. R. Conn, N. I. M. Gould, and P. L. Toint, *Testing a class of methods for solving minimization problems with simple bounds on the variables*, Math. Comp., 50 (1988), pp. 399–430.

[6] C. Davis, *Theory of positive linear dependence*, Amer. J. Math., 76 (1954), pp. 733–746.

[7] J. E. Dennis, Jr., and V. Torczon, *Direct search methods on parallel machines*, SIAM J. Optim., 1 (1991), pp. 448–474.

[8] E. D. Dolan, *Pattern Search Behavior in Nonlinear Optimization*, Honors thesis, Department of Computer Science, College of William and Mary, Williamsburg, VA, 1999.

[9] E. D. Dolan, R. M. Lewis, and V. J. Torczon, *On the Local Convergence Properties of Pattern Search*, Tech. report 2000–36, Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, VA, 2000. SIAM J. Optim., submitted.

[10] H. Fischer and K. Ritter, *An asynchronous parallel Newton method*, Math. Program., 42 (1988), pp. 363–374.

[11] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. S. Sunderam, *PVM: Parallel Virtual Machine: A Users' Guide and Tutorial for Network Parallel Computing*, MIT Press, Cambridge, MA, 1994.

[12] W. G. Houf, J. F. Grcar, and W. G. Breiland, *A model for low pressure chemical vapor deposition in a hot-wall tubular reactor*, Materials Science Engineering B, Solid State Materials for Advanced Technology, 17 (1993), pp. 163–171.

[13] T. G. Kolda and V. Torczon, *On the Convergence of Asynchronous Parallel Direct Search*, in preparation.

[14] C. L. Lawson and R. J. Hanson, *Solving Least Squares Problems*, Classics Appl. Math. 15, SIAM, Philadelphia, 1995.

[15] R. M. Lewis, V. Torczon, and M. W. Trosset, *Why pattern search works*, Optima, 59 (1998), pp. 1–7.

[16] R. M. Lewis and V. J. Torczon, *Rank Ordering and Positive Bases in Pattern Search Algorithms*, Tech. report 96–71, Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, VA, 1996.

[17] R. M. Lewis and V. J. Torczon, *A Globally Convergent Augmented Lagrangian Pattern Search Algorithm for Optimization with General Constraints and Simple Bounds*, Tech. report 98–31, Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, VA, 1998. SIAM J. Optim., submitted.

[18] R. M. Lewis and V. J. Torczon, *Pattern search algorithms for bound constrained minimization*, SIAM J. Optim., 9 (1999), pp. 1082–1099.

[19] R. M. Lewis and V. J. Torczon, *Pattern search methods for linearly constrained minimization*, SIAM J. Optim., 10 (2000), pp. 917–941.

[20] J. J. Moré, B. S. Garbow, and K. E. Hillstrom, *Testing unconstrained optimization software*, ACM Trans. Math. Software, 7 (1981), pp. 17–41.

[21] J. S. Plank, Y. Kim, and J. Dongarra, *Fault tolerant matrix operations for networks of workstations using diskless checkpointing*, J. Parallel Distrib. Comput., 43 (1997), pp. 125–138.

[22] T. Quarles, A. R. Newton, D. O. Perderson, and A. Sangiovanni-Vincentelli, *SPICE*3 *Version* 3f3 *User's Manual*, Tech. report, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA, 1993.

[23] S. L. Smith, E. Eskow, and R. B. Schnabel, *Large adaptive, asynchronous stochastic global optimization algorithms for sequential and parallel computation*, in Large-Scale Numerical Optimization, T. F. Coleman and Y. Li, eds., SIAM, Philadelphia, 1990, pp. 207–227.

[24] T. L. Sterling, J. Salmon, D. J. Becker, and D. F. Savarese, *How to Build a Beowulf: A Guide to the Implementation and Application of PC Clusters*, MIT Press, Cambridge, MA, 1999.

[25] V. Torczon, *PDS: Direct Search Methods for Unconstrained Optimization on Either Sequential or Parallel Machines*, Tech. report TR92–09, Department of Computational and Applied Mathematics, Rice University, Houston, TX, 1992.

[26] V. Torczon, *On the convergence of pattern search algorithms*, SIAM J. Optim., 7 (1997), pp. 1–25.

[27] S. E. Wright, *A note on positively spanning sets*, Amer. Math. Monthly, 107 (2000), pp. 364–366.

SIAM J. SCI. COMPUT.
Vol. 23, No. 1, pp. 157–169

ON BLOCK PRECONDITIONERS FOR NONSYMMETRIC SADDLE POINT PROBLEMS*

PIOTR KRZYŻANOWSKI†

Abstract. We discuss a class of preconditioning methods for an iterative solution of algebraic nonsymmetric saddle point problems arising from a mixed finite element discretization of partial differential equations, in particular the Navier–Stokes equation. We prove that block diagonal and block triangular preconditioners based on *symmetric, positive definite* blocks guarantee that the convergence rate of the method is independent of the mesh parameter h.

Key words. saddle point problems, nonsymmetric, block preconditioners

AMS subject classifications. 65F10, 65N22, 65N30

PII. S1064827599360406

1. Introduction.

In many applications one needs to solve a discrete system of linear equations with a block matrix

$$\mathcal{M}\begin{pmatrix} u \\ p \end{pmatrix} \equiv \begin{pmatrix} A & B^T \\ B & -C \end{pmatrix}\begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}, \tag{1.1}$$

where the matrix block A is positive definite, yet not necessarily symmetric, and C is nonnegative. Our motivation for considering this kind of problem comes from computational fluid dynamics, precisely, from the linearized discrete mixed finite element Navier–Stokes equations, where, except in some very simple cases, A is nonsymmetric, while usually it is reasonable to assume that it is positive definite [10]. C can be zero or positive semidefinite when a stabilized method is applied. Since (1.1) is ill conditioned with respect to the mesh parameter h, our aim in this paper is to design and/or analyze block preconditioners for this system, for which an iterative method converges independently of h.

Block preconditioners for a symmetric matrix $\mathcal{M}$ have been considered by many authors, for example, Bramble and Pasciak [1], [3], Rusten and Winther [27], Silvester and Wathen [29], and Klawonn [20], [19]. A symmetrized approach has been considered by D'yakonov [7] and Bramble and Pasciak [2]. Block preconditioned Uzawa-type methods have also attracted research interest; see, for example, papers by Elman and Golub [12] or Bramble, Pasciak, and Vassilev [4]. These preconditioners have been verified in computational tests (see the references listed above and also [9]).

Elman and Silvester [10] were probably the first to analyze how block preconditioners do work in the nonsymmetric case. They analyzed discretizations of the Oseen equations

$$\begin{cases} -\nu\Delta u + (k\cdot\nabla)u + \nabla p = f, \\ \operatorname{div} u = 0, \end{cases}$$

*Received by the editors August 26, 1999; accepted for publication (in revised form) January 16, 2001; published electronically June 8, 2001. This work has been supported in part through Polish Scientific Research Committee grant KBN 8T11F01209. Results from this paper have been presented in restricted form in *Proceedings of the 11th International Conference on Domain Decomposition Methods*, Greenwich, UK, 1998.

http://www.siam.org/journals/sisc/23-1/36040.html

†Institute of Applied Mathematics, Warsaw University, Banacha 2, 02–097 Warszawa, Poland (piotr.krzyzanowski@mimuw.edu.pl).

which serve as a model of the linearized Navier–Stokes equations and lead to a nonsymmetric system. In papers by Elman, Silvester, and Wathen [13], [14], Elman [11], Silvester et al. [30], and Golub and Wathen [17], a range of block preconditioning methods for this kind of problem has been introduced and analyzed. Some of these methods show very good convergence properties with respect to both h and ν. Of these, [10], [11], and [30] address diagonal/triangular preconditioners using also inexact nonsymmetric solves, while [17] develops a method based on Stokes-like solves with inexact blocks. Recently, Bramble, Pasciak, and Vassilev studied the inexact Uzawa algorithm for nonsymmetric saddle point problems [5], exploiting preconditioners for the symmetric part of A.

Independently, Klawonn and Starke [23] provided a field-of-values analysis of block triangular preconditioners for the Oseen problem. Their analysis is valid for triangular block preconditioner with upper diagonal block that is "sufficiently close" to A.

In this paper, based on [24] and [26], we present a new mathematical analysis of block preconditioning algorithms and propose other approaches as well. Our analysis is valid for inexact *and* symmetric block solvers and shows that for diffusion-dominated Oseen equations, a sufficient preconditioner can be based just on the symmetric part of the diagonal blocks of $\mathcal{M}$. We perform our analysis in a discrete $H^1 \times L^2$-like norm (which is natural for the problem), using a custom inner product derived from the preconditioner being used.

We focus on block preconditioners built up from preconditioners for *symmetric* parts of diagonal blocks of (1.1) and we consider two preconditioning strategies, using block diagonal or block triangular matrices. The first approach is to symmetrize the problem. We prove that the resulting system spectral condition number is independent of the dimension of (1.1), so that the PCG method [16] converges uniformly in h. However, the numerical experiments that we report on show that this method is quite computationally intensive (because it uses two or more solves per iteration). Thus, our second approach relies on applying the GMRES [28] iterative method to system (1.1) preconditioned by a block triangular matrix. We prove that, under minor assumptions, the block triangular preconditioner guarantees the GMRES to converge independently of h.

In the case of the linearized Navier–Stokes equations, block preconditioners discussed in the paper can be based on symmetric and positive definite preconditioners for the discrete Laplacian and mass matrices. That gives an application programmer a great opportunity to reuse, in an efficient way, existing very powerful methods (or software) like domain decomposition [31] or multigrid [18] methods for these simpler problems. Using symmetric solvers can also lead to some computational savings. We believe that in certain cases this may be a robust alternative to custom domain decomposition preconditioners, such as those proposed by Klawonn and Pavarino [21], developed for the whole saddle point problem (for a comparison between a 2-level Schwarz method and block preconditioners based also on 2-level Schwarz methods, see another paper by Klawonn and Pavarino [22]).

A general drawback of the methods being analyzed in the paper is that their convergence rate deteriorates when the ellipticity constant of A decreases. Clearly, in such a case, the symmetric part holds too little information about A. This limits the application of these methods to flows with reasonably small Reynolds number.

The plan of the remaining sections is as follows. In section 2, we introduce the framework in which we shall analyze our algorithms. Some useful estimates are also provided there. Section 3 is devoted to the analysis of preconditioning methods which lead to symmetric positive definite systems. In section 4, we analyze a block triangular preconditioner for the GMRES method. We conclude with section 5, which contains the results of numerical experiments for the Oseen equation.

2. General assumptions.

We consider the saddle point problem (1.1) with matrix $\mathcal{M}$, which arises from a mixed finite element discretization of partial differential equations, so our framework should reflect the dependence of (1.1) on the mesh parameter h. For theoretical convenience, we shall carry our analysis in terms of linear operators rather than in terms of matrices.

Let $\bar{V}, \bar{W}$ be real Hilbert spaces with scalar products denoted by $((\cdot,\cdot))$ and $(\cdot,\cdot)$, respectively. The norms in these spaces, induced by the inner products, will be denoted by $||\cdot||$ and $|\cdot|$. We consider a family of finite-dimensional subspaces indexed by the parameter $h \in (0,1)$: $V_h \subset \bar{V}$, $W_h \subset \bar{W}$. If V_h, W_h come from the finite element approximation, the dimension of these subspaces increases for decreasing h.

Let us introduce three continuous bilinear forms, $a : \bar{V} \times \bar{V} \to R$, $b : \bar{V} \times \bar{W} \to R$, $c : \bar{W} \times \bar{W} \to R$, and assume that $a(\cdot,\,\cdot)$ is $\bar{V}$-elliptic, i.e.,

$$\exists 0 < \nu \le 1 \quad \forall v \in \bar{V}, \qquad a(v,\,v) \ge \nu ||v||^2 \tag{2.1}$$

(by analogy to the Oseen equations, we shall call ν the viscosity parameter), and that $c(\cdot,\,\cdot)$ satisfies

$$\exists \gamma \ge 0 \quad \forall p \in \bar{W}, \qquad c(p,\,p) \ge \gamma |p|^2 \tag{2.2}$$

(we allow $\gamma = 0$). Notice that $a(\cdot,\cdot)$ does not need to be symmetric. We shall also assume that V_h and W_h satisfy the uniform LBB condition (see [15])

$$\exists \beta > 0 \quad \forall h \in (0,1), \quad \forall p \in W_h, \qquad \beta |p| \le \sup_{v \in V_h, v \ne 0} \frac{b(v,\,p)}{||v||}. \tag{2.3}$$

In what follows we consider preconditioners for a family of finite-dimensional problems (we drop the subscript h for simplicity of notation).

Problem 2.1. *Find $(u,p) \in V \times W$ such that*

$$\mathcal{M}\begin{pmatrix} u \\ p \end{pmatrix} \equiv \begin{pmatrix} A & B^* \\ B & -C \end{pmatrix}\begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} F \\ G \end{pmatrix}. \tag{2.4}$$

The operators in (2.4) are

$$\begin{aligned} &A : V \to V, \quad ((Au,v)) = a(u,\,v) \quad \forall u,v \in V,\\ &B : V \to W, \quad (Bu,p) = b(u,\,p) \quad \forall u \in V,\ p \in W,\\ &C : W \to W, \quad (Cp,q) = c(p,\,q) \quad \forall p,q \in W, \end{aligned}$$

while the right-hand side $F \in V, G \in W$ is defined through $((F,v)) \equiv \langle\langle f,v \rangle\rangle$ and $(G,w) \equiv \langle g,w \rangle$, where f,g are given continuous functionals on $\bar{V}$, $\bar{W}$, and $\langle\langle\cdot,\cdot\rangle\rangle$, $\langle\cdot,\cdot\rangle$ denote the duality pairing in $\bar{V}$, $\bar{W}$, respectively. B^* denotes the formal adjoint operator to B, i.e., $(Bu,p) = ((u,B^*p))\ \forall u \in V,\ p \in W$.

We introduce two more operators, $A_0 : V \to V$ and $J_0 : W \to W$. We assume that they are self-adjoint, their inverses are easy to apply, and there exist positive constants a_0, a_1, b_0, b_1, which are independent of h and ν, such that

$$a_0((u,u)) \le ((A_0u,u)) \le a_1((u,u)) \qquad \forall u \in V, \tag{2.5}$$

$$b_0(p,p) \le (J_0p,p) \le b_1(p,p) \qquad \forall p \in W. \tag{2.6}$$

In other words, we shall always assume that A_0 and J_0 define good preconditioners for the Grammian matrices for the chosen bases in V and W, respectively. For example, in the case of the linearized Navier–Stokes equations, A_0 may be a preconditioner for the discrete Laplace operator, while J_0 may be a pressure mass matrix operator preconditioner. Thus, the A_0 preconditioner may be constructed using very efficient domain decomposition or multigrid techniques; for J_0, instead of domain decomposition, one can also use very cheap diagonal scaling [33].

For $f \in V$, $g \in W$, the energetic norms defined by $|||f|||_{A_0^{-1}} = ((A_0^{-1}f, f))^{1/2}$ and $|||g|||_{J_0^{-1}} = (J_0^{-1}g, g)^{1/2}$ are equivalent, with constants independent of h and ν, to $||f||$ and $|g|$, respectively. This fact follows directly from (2.5) and (2.6). The product space $V \times W$ is equipped with the natural scalar product $\langle \cdot, \cdot \rangle$,

$$\left\langle \begin{pmatrix} u \\ p \end{pmatrix}, \begin{pmatrix} v \\ q \end{pmatrix} \right\rangle = ((u, v)) + (p, q);$$

however, we are going to analyze the preconditioned problem using a custom inner product $[\cdot, \cdot]$, which is dependent on the preconditioners being used:

$$\left[\begin{pmatrix} u \\ p \end{pmatrix}, \begin{pmatrix} v \\ q \end{pmatrix} \right] = ((A_0 u, v)) + (J_0 p, q) = \left\langle \begin{pmatrix} A_0 & 0 \\ 0 & J_0 \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix}, \begin{pmatrix} v \\ q \end{pmatrix} \right\rangle . \tag{2.7}$$

Again, the norms generated by both products are equivalent to one another, with constants independent of h and ν.

The generic constant "const," which we shall use later in the paper, is independent of both the mesh parameter h and the viscosity parameter ν. The following estimates hold for the operators involved in Problem 2.1.

LEMMA 2.1. *The norms of A, B, C, A_0, J_0, $\mathcal{M}$ operators and their adjoints in adequate spaces are bounded independently of h and ν:*

$$\begin{aligned} &||A||_{V\to V}, \quad ||B||_{V\to W}, \quad ||C||_{W\to W}, \\ &||A^*||_{V\to V}, \quad ||B^*||_{W\to V}, \quad ||C^*||_{W\to W}, \\ &||A_0||_{V\to V}, \quad ||J_0||_{W\to W}, \quad ||\mathcal{M}||_{V\times W\to V\times W}, \quad ||\mathcal{M}^*||_{V\times W\to V\times W} \le \text{const}\,. \end{aligned}$$

Moreover,

$$\begin{aligned} &||A_0^{-1}||_{V\to V}, \quad ||J_0^{-1}||_{W\to W} \le \text{const}, \\ &||\mathcal{M}^{-1}||_{V\times W\to V\times W} \le \frac{1}{\nu}\,\text{const}\,. \end{aligned}$$

Proof. The norm estimates for $A, B, C, A_0, J_0, A_0^{-1}, J_0^{-1}$ and their adjoints follow from our assumptions on the corresponding bilinear forms and from (2.5), (2.6). Then the estimate for $\mathcal{M}$ and its adjoint follows. The estimate $||\mathcal{M}^{-1}||_{V\times W\to V\times W} \le \frac{1}{\nu}$ const is another statement of the stability result for saddle point problem solutions [6]: there exists const > 0, independent of h and ν, such that any solution $(u, p) \in V \times W$ of Problem 2.1 satisfies $||u|| + |p| \le \frac{1}{\nu}\,\text{const}(||F|| + |G|)$. □

3. Preconditioners leading to symmetric, positive definite problems. In this section we are going to extend the results obtained previously for symmetric saddle point problems, e.g., in [7] and [2]. This approach is of normal equations type, which influences the convergence speed and the computational complexity of the method, though it has certain interesting advantages, too. The transformed system allows for using the conjugate gradients (PCG) method, which is less memory consuming than, e.g., GMRES; for fixed ν, the condition number of transformed system is independent of h. Moreover, the analysis remains valid for A only V^0-elliptic; see Remark 3.1. As the condition number grows like $O(\nu^{-2})$, the use of these preconditioners is practically limited only to diffusion-dominated flows.

3.1. Block diagonal preconditioner. Using the block diagonal preconditioner

$$\mathcal{M}_D = \begin{pmatrix} A_0 & 0 \\ 0 & J_0 \end{pmatrix},$$

we transform system (2.4) into the following one.

PROBLEM 3.1. *Find $(u,p) \in V \times W$ such that*

$$\mathcal{M}_D^{-1}\mathcal{M}^*\mathcal{M}_D^{-1}\mathcal{M}\begin{pmatrix} u \\ p \end{pmatrix} = \mathcal{M}_D^{-1}\mathcal{M}^*\mathcal{M}_D^{-1}\begin{pmatrix} F \\ G \end{pmatrix}. \tag{3.1}$$

Clearly, the operator $\mathcal{P} = \mathcal{M}_D^{-1}\mathcal{M}^*\mathcal{M}_D^{-1}\mathcal{M}$ is self-adjoint with respect to the inner product $[\cdot,\cdot]$. The following theorem guarantees $\mathcal{P}$ is also well conditioned.

THEOREM 3.1. *There exist positive constants m_0, m_1, independent of h, ν, such that*

$$m_0 \cdot \nu^2 \cdot \left[\begin{pmatrix} u \\ p \end{pmatrix}, \begin{pmatrix} u \\ p \end{pmatrix}\right] \le \left[\mathcal{P}\begin{pmatrix} u \\ p \end{pmatrix}, \begin{pmatrix} u \\ p \end{pmatrix}\right] \le m_1 \cdot \left[\begin{pmatrix} u \\ p \end{pmatrix}, \begin{pmatrix} u \\ p \end{pmatrix}\right]. \tag{3.2}$$

Proof. Observe that

(3.3)
$$\begin{aligned}\left[\mathcal{P}\begin{pmatrix} u \\ p \end{pmatrix}, \begin{pmatrix} u \\ p \end{pmatrix}\right] &= ((A_0^{-1}(Au + B^*p), Au + B^*p)) + (J_0^{-1}(Bu - Cp), Bu - Cp) \\ &= |||Au + B^*p|||^2_{A_0^{-1}} + |||Bu - Cp|||^2_{J_0^{-1}}.\end{aligned}$$

In what follows we will use the fact that for $a, b \ge 0$, there holds $a^2 + b^2 \le (a+b)^2 \le 2(a^2 + b^2)$. Defining $\tilde{f} = Au + B^*p$ and $\tilde{g} = Bu - Cp$, we obviously have that (u,p) satisfies

$$\begin{cases} Au + B^*p &= \tilde{f}, \\ Bu - Cp &= \tilde{g}, \end{cases} \tag{3.4}$$

so from Lemma 2.1 we obtain

$$\left[\mathcal{P}\begin{pmatrix} u \\ p \end{pmatrix}, \begin{pmatrix} u \\ p \end{pmatrix}\right] = |||\tilde{f}|||^2_{A_0^{-1}} + |||\tilde{g}|||^2_{J_0^{-1}} \ge \text{const}\, \nu^2 \left(||u||^2 + |p|^2\right),$$

which yields the lower bound in (3.2). In order to prove the upper bound, we return to (3.3) and use Lemma 2.1 again.

$$\begin{aligned}\left[\mathcal{P}\begin{pmatrix} u \\ p \end{pmatrix}, \begin{pmatrix} u \\ p \end{pmatrix}\right] &= \left\langle \mathcal{M}^*\mathcal{M}_D^{-1}\mathcal{M}\begin{pmatrix} u \\ p \end{pmatrix}, \begin{pmatrix} u \\ p \end{pmatrix}\right\rangle \\ &\le ||\mathcal{M}^*|| \cdot ||\mathcal{M}_D^{-1}|| \cdot ||\mathcal{M}|| \cdot \left\langle \begin{pmatrix} u \\ p \end{pmatrix}, \begin{pmatrix} u \\ p \end{pmatrix}\right\rangle \\ &\le \text{const}\left[\begin{pmatrix} u \\ p \end{pmatrix}, \begin{pmatrix} u \\ p \end{pmatrix}\right],\end{aligned}$$

since from Lemma 2.1 it follows that $||\mathcal{M}_D^{-1}||_{V\times W \to V\times W} \le \text{const}$. □

Remark 3.1. *From a theoretical point of view it is worth noticing that the V-ellipticity assumption in Theorem* 3.1 *on A may be weakened. Indeed, all that is needed in the proof is that the saddle point problem* (3.4) *be uniquely solvable and that the stability estimate holds. According to* [15], *a sufficient condition is to assume the LBB condition* (2.3) *and the V^0-ellipticity of A:*

$$a(u,\, u) \geq \nu ||u||^2 \qquad \forall v \in V^0, \tag{3.5}$$

where $V^0 = \{v \in V : b(v,\, p) = 0 \quad \forall p \in W\}$.

The above preconditioning technique has been previously investigated in [25] in the context of micropolar equations.

3.2. Block triangular preconditioner. It is also possible to construct a symmetric preconditioner for the operator $\mathcal{M}$, which is based on a block lower triangular matrix:

$$\mathcal{M}_T = \begin{pmatrix} A_0 & 0 \\ B & J_0 \end{pmatrix}. \tag{3.6}$$

We transform system (2.4) into an equivalent one,

$$\mathcal{P}\begin{pmatrix} u \\ p \end{pmatrix} \equiv \mathcal{M}^*\mathcal{L}_T^*\mathcal{K}_0^{-1}\mathcal{L}_T\mathcal{M}\begin{pmatrix} u \\ p \end{pmatrix} = \mathcal{M}^*\mathcal{L}_T^*\mathcal{K}_0^{-1}\mathcal{L}_T\begin{pmatrix} F \\ G \end{pmatrix},$$

where

$$\mathcal{L}_T^*\mathcal{K}_0^{-1}\mathcal{L}_T \equiv \begin{pmatrix} I & A_0^{-1}B^* \\ 0 & -I \end{pmatrix}\begin{pmatrix} A_0^{-1} & 0 \\ 0 & J_0^{-1} \end{pmatrix}\begin{pmatrix} I & 0 \\ BA_0^{-1} & -I \end{pmatrix}.$$

Observe that $\mathcal{M}_D^{-1}\mathcal{L}_T$ is nothing but $\mathcal{M}_T^{-1}$ and that $\mathcal{P}$ is symmetric with respect to the scalar product $\langle\cdot,\, \cdot\rangle$.

Theorem 3.2. *There exist constants $0 < c_0 \leq c_1$ independent of h and ν, such that*

$$c_0 \cdot \nu^2 \left\langle \begin{pmatrix} u \\ p \end{pmatrix}, \begin{pmatrix} u \\ p \end{pmatrix}\right\rangle \leq \left\langle \mathcal{P}\begin{pmatrix} u \\ p \end{pmatrix}, \begin{pmatrix} u \\ p \end{pmatrix}\right\rangle \leq c_1 \left\langle \begin{pmatrix} u \\ p \end{pmatrix}, \begin{pmatrix} u \\ p \end{pmatrix}\right\rangle$$

$\forall (u,p) \in V \times W$.

Proof. First, we shall prove the lower bound. As in the previous section, the upper bound will be easy to derive from the boundedness assumptions (see Lemma 2.1). For $(u,p) \in V \times W$ we have

$$\left\langle \mathcal{P}\begin{pmatrix} u \\ p \end{pmatrix}, \begin{pmatrix} u \\ p \end{pmatrix}\right\rangle = \left\langle \mathcal{M}_D^{-1}\mathcal{L}_T\mathcal{M}\begin{pmatrix} u \\ p \end{pmatrix}, \mathcal{L}_T\mathcal{M}\begin{pmatrix} u \\ p \end{pmatrix}\right\rangle \tag{3.7}$$

and

$$\mathcal{L}_T\mathcal{M} = \begin{pmatrix} A & B^* \\ B(A_0^{-1}A - I) & BA_0^{-1}B^* + C \end{pmatrix}.$$

Therefore,

$$\left\langle \mathcal{P}\begin{pmatrix} u \\ p \end{pmatrix}, \begin{pmatrix} u \\ p \end{pmatrix}\right\rangle = |||Au + B^*p|||^2_{A_0^{-1}} + |||BA_0^{-1}(A - A_0)u + (BA_0^{-1}B^* + C)p|||^2_{J_0^{-1}}. \tag{3.8}$$

Defining this time $\tilde{f} = Au + B^*p$ and $\tilde{g} = BA_0^{-1}(A - A_0)u + (BA_0^{-1}B^* + C)p$ we obviously have that the pair (u, p) satisfies the system

$$\left\{ \begin{array}{l} Au + B^*p = \tilde{f}, \\ Cp - Bu = \tilde{g} - BA_0^{-1}(Au + B^*p) = \tilde{g} - BA_0^{-1}\tilde{f}. \end{array} \right. \tag{3.9}$$

From stability estimates for the saddle point problem (see Lemma 2.1), it follows that the solution (u, p) of (3.9) satisfies

$$\nu \cdot (||u|| + |p|) \leq \text{const} \cdot (||\tilde{f}|| + |BA_0^{-1}\tilde{f} - \tilde{g}|) \leq \text{const} \cdot (|||\tilde{f}|||_{A_0^{-1}} + |||\tilde{g}|||_{J_0^{-1}}). \tag{3.10}$$

This estimate yields the lower bound on the spectrum of $\mathcal{P}$. To get the upper bound we return to (3.8) and use the estimates from section 2. □

REMARK 3.2. *Observe that the preconditioner discussed here is more costly to evaluate than the diagonal one: it requires three applications of A_0^{-1} per inner product, compared to only one A_0^{-1} for the diagonal preconditioner analyzed in the previous subsection.*

REMARK 3.3. *Again, the ellipticity assumption on A in Theorem* 3.2 *may be weakened. V^0-ellipticity of A (see* (3.5)*) is sufficient.*

4. Preconditioner for the GMRES method. It is well known that after symmetrizing the system, its condition number increases; moreover, the symmetrized preconditioned matrix is quite costly to apply, as it takes at least two preconditioner solves. This fact is also indicated by numerical experiments (see section 5). Therefore, rather than symmetrizing the system, we may use an efficiently preconditioned GMRES method for our problem. If the preconditioner works well enough so that satisfactory approximation is obtained after few iterations, then the GMRES memory consumption is less painful, and we can benefit from its better convergence properties.

For $k > 0$ to be specified later, let us consider the block triangular preconditioner,

$$\mathcal{M}_T = \begin{pmatrix} k \cdot A_0 & 0 \\ B & -k \cdot J_0 \end{pmatrix},$$

which was previously discussed (with another choice of scaling parameters), e.g., in [1], [10], [13] and [19], [23]. The convergence rate of the GMRES method for a system

$$\mathcal{M}_T^{-1}\mathcal{M}\begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} F \\ G \end{pmatrix} = \mathcal{M}_T^{-1}\begin{pmatrix} f \\ g \end{pmatrix} \tag{4.1}$$

can be estimated by means of two parameters (see [8]),

$$\underline{c} = \inf_{(u,p)\neq 0} \frac{\left[\mathcal{M}_T^{-1}\mathcal{M}\begin{pmatrix} u \\ p \end{pmatrix}, \begin{pmatrix} u \\ p \end{pmatrix}\right]}{\left[\begin{pmatrix} u \\ p \end{pmatrix}, \begin{pmatrix} u \\ p \end{pmatrix}\right]}, \quad \bar{c} = \sup_{(u,p)\neq 0} \frac{\left[\mathcal{M}_T^{-1}\mathcal{M}\begin{pmatrix} u \\ p \end{pmatrix}, \mathcal{M}_T^{-1}\mathcal{M}\begin{pmatrix} u \\ p \end{pmatrix}\right]}{\left[\begin{pmatrix} u \\ p \end{pmatrix}, \begin{pmatrix} u \\ p \end{pmatrix}\right]}. \tag{4.2}$$

Recall that $[\cdot, \cdot]$ is defined by (2.7) and generates the norm in which the residual for GMRES is measured.

We shall prove that these parameters are bounded independently of h. Since they are strongly dependent on ν, the preconditioner use is practically limited to diffusion dominated flows.

LEMMA 4.1. *There exists $k > 0$ such that the symmetric part of A, $A_{symm} = (A + A^*)/2$, together with $A^* A_0^{-1} A$, satisfies*

$$\left(\left(\left(A_{symm} - \frac{1}{2k} A^* A_0^{-1} A\right) u,\ u\right)\right) \geq \delta((A_0 u,\ u)) \qquad \forall u \in V \tag{4.3}$$

for some constant $\delta \geq \text{const} \cdot \nu$. The scaling parameter k can be chosen independently of h, precisely, $k \geq \text{const} \cdot \nu^{-1}$.

Proof. Let λ be the smallest eigenvalue of the following generalized eigenproblem:

$$A_{symm} u = \lambda A_0 u,$$

and let μ be the largest eigenvalue of another generalized eigenproblem,

$$A^* A_0^{-1} A u = \mu A_0 u.$$

From (2.1) it follows that $\lambda \geq \text{const}\, \nu$, while from Lemma 2.1 we have $\mu \leq \text{const}$. Then

$$\left(\left(\left(A_{symm} - \frac{1}{2k} A^* A_0^{-1} A\right) u,\ u\right)\right) \geq \left(\lambda - \frac{\mu}{2k}\right) ((A_0 u,\ u)),$$

and for $k \geq \text{const} \cdot \nu^{-1} \geq \frac{\mu}{\lambda}$, (4.3) holds with $\delta = \frac{\lambda}{2}$. This gives the estimate $\delta \geq \text{const} \cdot \nu$. □

REMARK 4.1. *The scaling requirement is similar to that of Bramble and Pasciak* [1] *for the triangular preconditioner in the symmetric case. In the general case, the scaling parameter k would need to be computed from (rough) estimates of the above eigenvalues. In practice, however, the scaling is usually not essential for the convergence. For example, in our numerical experiments (see section 5) there was no significant difference between scaled and unscaled preconditioner convergence.*

THEOREM 4.2. *For any fixed $\nu > 0$ and for k as in Lemma 4.1, the convergence rate of the GMRES for (4.1) is independent of h, precisely,*

$$\underline{c} \geq \text{const} \cdot \nu^2, \qquad \bar{c} \leq \text{const}\,.$$

Proof. Our aim is to estimate the quantities $\underline{c}, \bar{c}$ in (4.2) independently of h. Since

$$\begin{pmatrix} kA_0 & 0 \\ 0 & kJ_0 \end{pmatrix} \cdot \mathcal{M}_T^{-1} = \begin{pmatrix} I & 0 \\ \frac{1}{k} B A_0^{-1} & -I \end{pmatrix},$$

then

$$\left[\mathcal{M}_T^{-1} \mathcal{M} \begin{pmatrix} u \\ p \end{pmatrix},\ \begin{pmatrix} u \\ p \end{pmatrix}\right] = \frac{1}{k}((Au,\ u)) + \frac{1}{k^2}((A_0^{-1} Au,\ B^* p)) + \frac{1}{k^2}((A_0^{-1} B^* p,\ B^* p)) + (Cp,\ p).$$

From (2.2) we estimate

$$\left[\mathcal{M}_T^{-1} \mathcal{M} \begin{pmatrix} u \\ p \end{pmatrix},\ \begin{pmatrix} u \\ p \end{pmatrix}\right] \geq \frac{1}{2k^2} |||Au + B^* p|||^2_{A_0^{-1}} + \frac{1}{k}\left(\left(\left(A_{symm} - \frac{1}{2k} A^* A_0^{-1} A\right) u,\ u\right)\right) + \frac{1}{2k^2} |||B^* p|||^2_{A_0^{-1}}.$$

Obviously, $|||Au + B^*p|||^2_{A_0^{-1}} \geq 0$ and $|||B^*p|||^2_{A_0^{-1}} = (BA_0^{-1}B^*p,\, p) \geq \mathrm{const}(J_0p,\, p)$, the latter from (2.6) and the LBB condition; see [1]. Therefore, for k and δ as in Lemma 4.1, we can conclude that

$$\left[\mathcal{M}_T^{-1}\mathcal{M}\begin{pmatrix}u\\p\end{pmatrix}, \begin{pmatrix}u\\p\end{pmatrix}\right] \geq \mathrm{const}\,\nu^2 \left[\begin{pmatrix}u\\p\end{pmatrix}, \begin{pmatrix}u\\p\end{pmatrix}\right].$$

In order to estimate $\bar{c}$, observe that

$$\begin{aligned}\left[\mathcal{M}_T^{-1}\mathcal{M}\begin{pmatrix}u\\p\end{pmatrix}, \mathcal{M}_T^{-1}\mathcal{M}\begin{pmatrix}u\\p\end{pmatrix}\right] &= \left\langle\begin{pmatrix}A_0 & \\ & J_0\end{pmatrix}\mathcal{M}_T^{-1}\mathcal{M}\begin{pmatrix}u\\p\end{pmatrix}, \mathcal{M}_T^{-1}\mathcal{M}\begin{pmatrix}u\\p\end{pmatrix}\right\rangle\\ &= \left\langle\mathcal{M}^*(\mathcal{M}_T^{-1})^*\mathcal{M}_D\mathcal{M}_T^{-1}\mathcal{M}\begin{pmatrix}u\\p\end{pmatrix}, \begin{pmatrix}u\\p\end{pmatrix}\right\rangle\\ &= ||\mathcal{M}^*|| \cdot ||(\mathcal{M}_T^{-1})^*|| \cdot ||\mathcal{M}_D|| \cdot ||\mathcal{M}_T^{-1}|| \cdot ||\mathcal{M}|| \cdot \left\langle\begin{pmatrix}u\\p\end{pmatrix}, \begin{pmatrix}u\\p\end{pmatrix}\right\rangle.\end{aligned}$$

As $k \geq \mathrm{const}\cdot\nu^{-1}$, then using estimates from section 2 and the factorization

$$\mathcal{M}_T^{-1} = \begin{pmatrix}I & \\ & \frac{1}{k}J_0^{-1}\end{pmatrix}\begin{pmatrix}I & 0\\ B & -I\end{pmatrix}\begin{pmatrix}\frac{1}{k}A_0^{-1} & \\ & I\end{pmatrix},$$

we have that $||\mathcal{M}_T^{-1}||, ||(\mathcal{M}_T^{-1})^*|| \leq \max\{1, \frac{1}{k^2}\}\cdot\mathrm{const} \leq \mathrm{const}$, and finally

$$\left[\mathcal{M}_T^{-1}\mathcal{M}\begin{pmatrix}u\\p\end{pmatrix}, \mathcal{M}_T^{-1}\mathcal{M}\begin{pmatrix}u\\p\end{pmatrix}\right] \leq \mathrm{const}\left[\begin{pmatrix}u\\p\end{pmatrix}, \begin{pmatrix}u\\p\end{pmatrix}\right],$$

which completes the proof. □

5. Numerical experiments. We implemented certain preconditioners considered in this paper and examined their applicability in the iterative solution of a nonsymmetric saddle point problem. The test problem was to solve the Oseen equations in a rectangle $\Omega = (-1, 1) \times (-1, 1)$

$$\begin{cases}-\nu\Delta u + (k\cdot\nabla)u + \nabla p = f,\\ \operatorname{div} u = 0,\end{cases}$$

with homogeneous Dirichlet boundary conditions on u. The function $k(\cdot,\cdot)$ was defined as in [10], [23], as a simple vortex,

$$k(x, y) = \begin{pmatrix}2y(1-x^2)\\ -2x(1-y^2)\end{pmatrix}.$$

For the right-hand side vector in our experiments, we always took a random vector with elements uniformly distributed between $(-1, 1)$. We discretized the equation using popular Q_2–Q_1 Taylor–Hood rectangular finite elements; see [6]. The mesh was uniform in both directions, with nx inner pressure nodes along the x-axis and ny inner pressure nodes in y-direction, so there were approximately twice as many inner nodes in each direction for each component of the discrete velocity. In all experiments presented below, $nx = ny$. To keep the pressure in L_0^2, we added to the system a discretized condition $\int p = 0$ using the Lagrange multipliers.

All algorithms were implemented on an unloaded Linux PC, using PETSc 2.0.29 library subroutines for the iterative solvers and additive Schwarz preconditioners [32]. We performed our tests in three different solver/preconditioner configurations.

Table 5.1

Iteration counts for different solvers and preconditioning strategies: results for A_0 = Laplacian, J_0 = mass matrix. Wall clock time (seconds) in parentheses.

nx	DOF[1]	GMRES/diag	GMRES/triang	CG/symm
10	1202	38 (0.4)	21 (0.2)	45 (0.7)
20	4182	39 (1.5)	21 (0.9)	46 (2.8)
40	15542	40 (8.0)	21 (3.8)	49 (12.8)
80	59862	39 (47.3)	21 (17.7)	49 (56.1)

Table 5.2

Timings relative to GMRES/triang for different solver/preconditioner strategies reported in Tables 5.1 *and* 5.3.

	nx	GMRES/diag	GMRES/triang	CG/symm
"best"	20	1.7	1.0	3.1
	40	2.1	1.0	3.4
	80	2.7	1.0	3.2
ASM	20	1.4	1.0	2.3
	40	1.4	1.0	1.9
	80	1.7	1.0	1.6

1. CG/symm: the symmetrized preconditioned conjugate gradient method with the block diagonal preconditioner; see section 3.1.
2. GMRES/triang: the GMRES method with the block triangular preconditioner; see section 4.
3. GMRES/diag: the GMRES method with the block diagonal preconditioner.

To compare the overall efficiency of the preconditioners, we report both the iteration counts and the wall clock times (in seconds) needed to reduce the initial residual by a factor of 10^6. We used a restarted version of GMRES, with restart after every 30 iterations.

Two simplifications to GMRES algorithms have been made, which, as it turns out in the preliminary set of experiments, hardly influences the convergence rate of the methods. Our GMRES algorithm minimizes the Euclidean norm of the residual, instead of the energy norm analyzed in section 4. For the same reason we always used $k = 1$ (see Theorem 4.2). All this makes the GMRES algorithm cheaper.

Except for the last set of experiments, we restrict ourselves to the case where the viscosity parameter $\nu = 1$. The influence of ν on the convergence rate will be reported at the end of this section. As it has been indicated in the above theorems and will be confirmed in Table 5.5 as well, the performance of our preconditioner deteriorates for small values of the viscosity parameter ν. This is the reason that we focus mainly on diffusion dominated flow.

First, we conducted experiments under the best available circumstances, that is, we used as the preconditioners A_0 = Laplacian, J_0 = mass matrix. Notice that, in contrast to [11] and [23], we restricted ourselves only to *symmetric* preconditioners. We refer to the above choice of preconditioners as the "best" preconditioners, since in this case we exactly solve the symmetric part of the relevant diagonal operators. Our results are presented in Tables 5.1 and 5.2 and Figure 5.1.

Next, we tested the same methods using inexact preconditioners for A_0, J_0, namely, the 1-level additive Schwarz method (ASM) with standard black-box decomposition

[1] DOF = the total number of degrees of freedom.

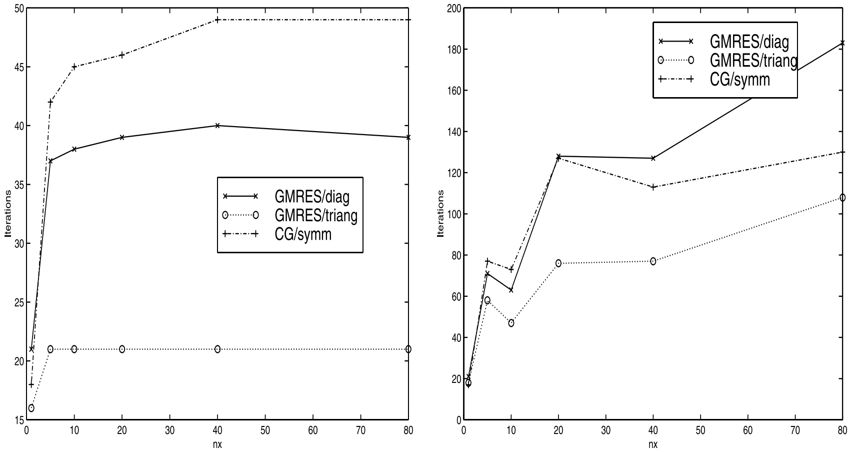


Fig. 5.1. *The growth of iterations as a function of the problem size for different solvers and preconditioning strategies. Left:* A_0 = *Laplacian,* J_0 = *mass matrix. Right: additive Schwarz preconditioners (fixed number of subdomains, small overlap, no coarse grid).*

Table 5.3

Iteration counts for different solvers and preconditioning strategies: The results for additive Schwarz preconditioners. (Timings in parentheses.)

nx	DOF	GMRES/diag	GMRES/triang	CG/symm
10	1202	63 (1.0)	51 (0.7)	73 (1.7)
20	4182	128 (6.0)	76 (4.2)	128 (9.8)
40	15542	127 (23.1)	77 (16.3)	113 (31.6)
80	59862	183 (155.0)	108 (91.8)	130 (146.7)

into 6 rectangular subdomains with small and fixed overlap (2 nodes) provided by PETSc. The results are presented in Tables 5.2 and 5.3 and Figure 5.1.

The experiments confirm theoretical results, showing that the number of iterations reflects the quality of the preconditioning blocks being used. This may be seen very clearly from the experiments with "best" solvers and implicitly from those for the inexact solvers (Figure 5.1). For $\nu = 1$, the number of iterations is moderate and, with "best" solvers, virtually independent of the mesh size. As one could expect, the triangular preconditioner is most effective, being usually more than two times faster than diagonal preconditioners. The CG/symm combination is even slower than GMRES/diag.

According to [31], for the ASM preconditioning blocks, with fixed number of subdomains, small overlap, and no coarse grid, the number of iterations in Table 5.3 should increase as the square root of nx, and this is approximately what we actually see. Theorems 3.1 and 4.2 ensure that the convergence rate would be fully independent of nx if a 2-level ASM were used.

Another experiment (Table 5.4) shows the performance of our preconditioners when the number of subdomains is scaled with the problem size, so that $\sqrt{N}/nx$ remains constant. As compared to [21, Table 1], the number of iterations of the GMRES/triang method is quite similar to a 1-level overlapping ASM for the whole system (1.1), despite the fact that our decomposition might suffer from subdomain bad aspect ratios (we did not have control over the shape of the subdomains). This indicates that a block 1-level ASM preconditioner may be an alternative to 1-level

Table 5.4

Iteration counts for different solvers and preconditioning strategies: the results for additive Schwarz preconditioners with number of subdomains scaled with problem size, so that $\sqrt{N}/nx = 4$. (Note: full GMRES used.)

$\sqrt{N}$	nx	DOF	GMRES/diag	GMRES/triang	CG/symm
1	4	279	34	21	40
2	8	823	66	51	96
4	16	2775	95	62	108
8	32	10135	165	124	139
10	40	15543	221	147	184

Table 5.5

Iteration counts for varying mesh size nx and viscosity ν with preconditioning blocks A_0 = Laplacian, J_0 = mass matrix.

	GMRES/diag			GMRES/triang			CG/symm		
ν	1.0	0.1	0.02	1.0	0.1	0.02	1.0	0.1	0.02
nx									
16	38	111	> 300	21	76	255	43	42	116
32	40	102	> 300	22	73	> 300	48	41	117
64	40	105	> 300	22	73	> 300	48	37	115

ASM designed in [21] for the full system, although it seems that 1-level ASM for the full system is still a bit cheaper. More comparisons of this type have recently been made in a paper by Klawonn and Pavarino [22].

Finally, we examined the behavior of our methods for decreasing values of the viscosity parameter ν. They confirm that symmetric-block-based preconditioners are competitive to those using nonsymmetric A solves only if the symmetric part dominates in A; for small ν, the convergence rate deteriorates very quickly, as indicated in Table 5.5.

Acknowledgments. The author wishes to thank Professor Maksymilian Dryja and Professor Krzysztof Moszyński for their valuable remarks. The comments of two anonymous referees also significantly helped to improve the paper.

REFERENCES

[1] J. Bramble and J. Pasciak, *A preconditioning technique for indefinite problems resulting from mixed approximation of elliptic problems*, Math. Comp., 50 (1988), pp. 1–17.

[2] J. Bramble and J. Pasciak, *A domain decomposition technique for Stokes problems*, Appl. Numer. Math., 6 (1990), pp. 251–261.

[3] J. Bramble and J. Pasciak, *Iterative techniques for time dependent Stokes problems. Approximation theory and applications*, Comput. Math. Appl., 33 (1997), pp. 13–30.

[4] J. H. Bramble, J. E. Pasciak, and A. T. Vassilev, *Analysis of the inexact Uzawa algorithm for saddle point problems*, SIAM J. Numer. Anal., 34 (1997), pp. 1072–1092.

[5] J. H. Bramble, J. E. Pasciak, and A. T. Vassilev, *Uzawa type algorithms for nonsymmetric saddle point problems*, Math. Comp., 69 (2000), pp. 667–689.

[6] F. Brezzi and M. Fortin, *Mixed and Hybrid Finite Element Methods*, Springer-Verlag, New York, Berlin, Heidelberg, London, Paris, Tokyo, Hong Kong, Barcelona, 1991.

[7] E. D'yakonov, *On iterative methods with saddle operators*, Soviet Math. Dokl., 35 (1987), pp. 166–170.

[8] S. C. Eisenstat, H. C. Elman, and M. H. Schultz, *Variational iterative methods for nonsymmetric systems of linear equations*, SIAM J. Numer. Anal., 20 (1983), pp. 345–357.

[9] H. Elman, *Multigrid and Krylov subspace methods for the discrete Stokes equations*, Internat. J. Numer. Methods Fluids, 22 (1996), pp. 755–770.

[10] H. Elman and D. Silvester, *Fast nonsymmetric iterations and preconditioning for Navier–Stokes equations*, SIAM J. Sci. Comput., 17 (1996), pp. 33–46.

[11] H. C. Elman, *Preconditioning for the steady-state Navier–Stokes equations with low viscosity*, SIAM J. Sci. Comput., 20 (1999), pp. 1299–1316.

[12] H. C. Elman and G. H. Golub, *Inexact and preconditioned Uzawa algorithms for saddle point problems*, SIAM J. Numer. Anal., 31 (1994), pp. 1645–1661.

[13] H. C. Elman, D. J. Silvester, and A. J. Wathen, *Iterative methods for problems in computational fluid dynamics*, in Iterative Methods in Scientific Computing (Hong Kong, 1995), Springer-Verlag, Singapore, 1997, pp. 271–327.

[14] H. C. Elman, D. J. Silvester, and A. J. Wathen, *Performance and Analysis of Saddle Point Preconditioners for the Discrete Steady-State Navier–Stokes Equations*, Tech. report CS-TR 4164, Department of Computer Science, University of Maryland, 2000.

[15] V. Girault and P. Raviart, *Finite Element Method for Navier–Stokes Equations. Theory and Algorithms*, Springer-Verlag, Berlin, Heidelberg, New York, 1986.

[16] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed., Johns Hopkins Stud. Math. Sci., Johns Hopkins University Press, Baltimore, MD, 1996.

[17] G. H. Golub and A. J. Wathen, *An iteration for indefinite systems and its application to the Navier–Stokes equations*, SIAM J. Sci. Comput., 19 (1998), pp. 530–539.

[18] W. Hackbusch, *Multigrid Methods and Applications*, Springer-Verlag, Berlin, 1985.

[19] A. Klawonn, *Block-triangular preconditioners for saddle point problems with a penalty term*, SIAM J. Sci. Comput., 19 (1998), pp. 172–184.

[20] A. Klawonn, *An optimal preconditioner for a class of saddle point problems with a penalty term*, SIAM J. Sci. Comput., 19 (1998), pp. 540–552.

[21] A. Klawonn and L. F. Pavarino, *Overlapping Schwarz methods for mixed linear elasticity and Stokes problems*, Comput. Methods Appl. Mech. Engrg., 165 (1998), pp. 233–245.

[22] A. Klawonn and L. F. Pavarino, *A comparison of overlapping Schwarz methods and block preconditioners for saddle point problems*, Numer. Linear Algebra Appl., 7 (2000), pp. 1–25.

[23] A. Klawonn and G. Starke, *Block triangular preconditioners for nonsymmetric saddle point problems: Field-of-values analysis*, Numer. Math., 81 (1999), pp. 577–594.

[24] P. Krzyżanowski, *A Note on Preconditioning Nonsymmetric Saddle Point Problems*, Tech. report RW 97-09, Institute of Applied Mathematics and Mechanics, Warsaw University, Warsaw, Poland, 1997.

[25] P. Krzyżanowski, *Domain decomposition method for finite element micropolar equations*, in Proceedings of the 9th International Symposium on Domain Decomposition Methods, Ullensvang, Norway, P. E. Bjørstad, M. Espedal, and D. Keyes, eds., DDM.org, 1998. Also available electronically at http://www.ddm.org/DD9/krzyzanowski.ps.gz.

[26] P. Krzyżanowski, *Mathematical and Numerical Analysis of Problems in Microcontinuum Mechanics*, Ph.D. thesis, Warsaw University, Poland, 1998.

[27] T. Rusten and R. Winther, *A preconditioned iterative method for saddle point problems*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 887–904.

[28] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[29] D. Silvester and A. Wathen, *Fast iterative solution of stabilized Stokes systems.* II. *Using general block preconditioners*, SIAM J. Numer. Anal., 31 (1994), pp. 1352–1367.

[30] D. J. Silvester, H. C. Elman, D. Kay, and A. J. Wathen, *Efficient Preconditioning of the Linearized Navier–Stokes Equations*, Tech. report CS-TR-4073, Department of Computer Science, University of Maryland, 1999.

[31] B. Smith, P. Bjorstad, and W. Gropp, *Domain Decomposition. Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge University Press, Cambridge, UK, 1996.

[32] B. Smith, W. Gropp, and L. McInnes, *PETSc* 2.0 *Users Manual*, Tech. report ANL-95/11, Argonne National Laboratory, Argonne, IL, 1997. Also available via ftp://www.mcs.anl/pub/petsc/manual.ps.

[33] A. J. Wathen, *Realistic eigenvalue bounds for the Galerkin mass matrix*, IMA J. Numer. Anal., 7 (1987), pp. 449–457.

SIAM J. Sci. Comput.
Vol. 23, No. 1, pp. 170–192

VORTICITY-PRESERVING LAX–WENDROFF-TYPE SCHEMES FOR THE SYSTEM WAVE EQUATION[*]

K. W. MORTON[†] AND P. L. ROE[‡]

Abstract. In numerical solutions of fluid flow, vorticity can be generated by truncation errors. We analyze this phenomenon for linearized equations and give conditions for preventing it. The Lax–Wendroff method that meets these constraints is essentially unique, although there are two distinct interpretations, and also turns out to have optimal properties regarding stability and truncation error. The extension of the scheme to unstructured grids is given, together with some discussion of practical problems to which these schemes might bring improvement.

Key words. vorticity-preserving, system wave equation, discrete Kelvin's theorem

AMS subject classifications. 35B05, 35F05, 35L05, 35L65, 65M06, 76N10

PII. S106482759935914X

1. Introduction.

Much of computational fluid dynamics (CFD) is based on the analysis of simple "model problems." To an astonishing extent, the solution of problems governed by hyperbolic equations is based on analysis of the one-dimensional linear advection equation $u_t+au_x = 0$. However, neither this nor the multidimensional $u_t+\vec{a}\cdot\nabla u = 0$ really reflects the richness of fluid behavior, capturing merely advection and some aspects of wave propagation but ignoring, for example, all phenomena associated with vorticity, which are of vital importance in many three-dimensional situations.

The simplest model problem to combine wave propagation and vorticity is the system wave equation. We will write this in two space dimensions in the matrix form, using a notation corresponding to acoustic waves in a fluid that is stationary in the mean, with pressure p^* and velocity $\vec{u}^* = (u^*, v^*)$; thus

$$\partial_t \mathbf{u} + cL\mathbf{u} = 0. \tag{1}$$

Here $\mathbf{u} = (p^*/(\rho c^2), u^*/c, v^*/c) \equiv (p, u, v,)$, c is the sound speed in the mean flow, and

$$L = \begin{bmatrix} 0 & \partial_x & \partial_y \\ \partial_x & 0 & 0 \\ \partial_y & 0 & 0 \end{bmatrix}. \tag{2}$$

Restriction to two dimensions is merely for economy of notation; all of the analysis at the PDE level extends very straightforwardly to three dimensions, as does the numerical analysis on Cartesian grids. We study the wave equation in system rather than scalar ($\partial_{tt}u = c^2\nabla^2 u$) form for two reasons: first, because this is the form of

[*] Received by the editors July 22, 1999; accepted for publication (in revised form) January 16, 2001; published electronically June 19, 2001.
http://www.siam.org/journals/sisc/23-1/35914.html

[†] Department of Mathematical Sciences, University of Bath, Bath BA2 7AY, UK and Oxford University Computing Laboratory, Oxford, UK (Bill.Morton@comlab.ox.ac.uk).

[‡] Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI and Centre pour Mathematiques et Leurs Applications, Ecole Normale Supérieure de Cachan, France (philroe@engin.umich.edu).

the wave equation that is hidden inside the Euler equations, whether in their two-dimensional unsteady or in their three-dimensional supersonic steady forms [1], and second, because the scalar form automatically implies vanishing vorticity, whereas the interaction of the waves with vorticity is one of the aspects we want to study.

Here the interaction is very simple, as befits a model problem. We easily deduce from (1), (2), if c is a constant, that

$$\partial_t \zeta = 0, \qquad \text{where} \qquad \zeta = \partial_x v - \partial_y u.$$

In other words, there is no interaction, and any initial distribution of vorticity is preserved. Maintaining this independence at the discrete level will be one of our objectives. In section 6, however, we consider more general, variable coefficient, problems, for which vorticity is created by the interaction of waves with density gradients. In this case, there is a very satisfying discrete parallel.

It is a trivial modification to change the notation so that (1), (2) describe Maxwell's equations, and the constraint of invariant vorticity becomes one of invariant (and vanishing) divergence of the magnetic field. The two cases differ, however, when nonconstant coefficients are involved, a case that will be treated in due course. The analysis also applies to the divergence constraint in magnetohydrodynamics and to maintaining the div-curl identity when using a velocity-vorticity formulation of the Navier–Stokes equations.

We study fully discrete schemes because of our personal conviction that transient hyperbolic problems are most naturally treated by such methods. More specifically, we study Lax–Wendroff schemes because the Taylor expansion of the evolution operator seems to be the most general technique available, applicable to the wave equation for any order of accuracy. However, Lax–Wendroff schemes are ambiguous in more than one dimension. For example, if we look to discretize $u_t + \vec{a} \cdot \nabla u = 0$ to second-order accuracy on the standard nine-point stencil, we have only six constraints on the nine coefficients and hence three degrees of freedom. There are even more degrees of freedom in discretizing a system of equations. One reason for the rise of finite-element and finite-volume methodologies is that by working within a more disciplined framework some of the ambiguity is removed, hopefully without at the same time losing valuable options.

In the early stages of the present analysis we consider mainly regular rectangular grids with uniform spacing h in both x and y. We do not impose any particular interpretation on the discrete solution $u^n_{i,j}$: the values could represent cell-averaged quantities as in a cell-centered finite-volume scheme, or nodal values as in a vertex-centered finite-difference scheme. One point of interest is that by initially taking a simple finite-difference viewpoint, the formulae that emerge as having distinguished properties are precisely those that have dual interpretations as each of the above. Subsequently, however, we find that only the finite-volume interpretation generalizes to unstructured grids but that the particular form of finite-volume scheme to emerge is not the one most commonly encountered.

In section 2 of the paper, we recall some simple formulae of the finite-difference calculus and some elementary properties of the wave equation. In section 3 we place constraints on the scheme such that some discrete measure of vorticity is preserved and in section 4 show that requiring an adjoint property on a minimal stencil results in a unique version of the Lax–Wendroff scheme, although one having two distinct implementations. In section 5 we analyze the stability and truncation error of this scheme. In section 6 we show that if the waves are being propagated in a uniformly

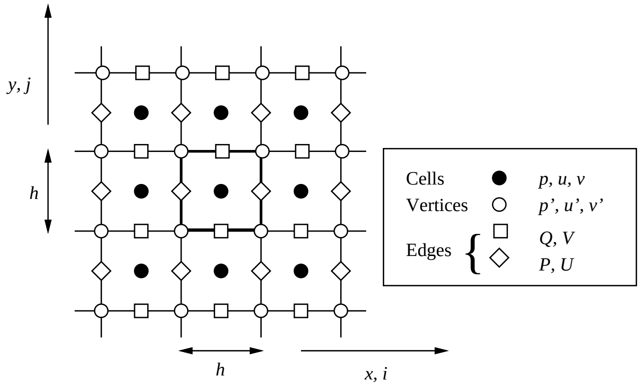


FIG. 1. *Grid definition.*

moving medium, then any vorticity present in the initial conditions will be propagated numerically according to the scalar version of our Lax–Wendroff method.

In section 7 we turn to irregular grids, and we find that there is still a measure of vorticity that is conserved, and in section 8 generalize further to the case of a problem with variable coefficients, although still linear. In this case the argument can be made global to yield a discrete Kelvin theorem.

2. Properties of the exact and discrete solution operators.

2.1. Discrete notation. As stated in the introduction, we concentrate at first on simple finite-difference formulations on uniform grids. These can be manipulated using a calculus that is almost as transparent as that of the differential operators. We exploit this simplicity to remove ambiguity from the finite-difference formulae by requiring certain algebraic properties. Then we ask what interpretations are compatible with the formulae.

We adopt a uniform square grid, such that the spacing in the x and y directions is h and the time step is Δt, with $u^n_{i,j}$ a discrete approximation located at $(x, y, t) = (ih, jh, n\Delta t)$. The standard discrete differencing and averaging operators are defined by

$$\delta_x()_{\cdot,\cdot} = ()_{\cdot+\frac{1}{2},\cdot} - ()_{\cdot-\frac{1}{2},\cdot}, \qquad \mu_x()_{\cdot,\cdot} = \tfrac{1}{2}\{()_{\cdot+\frac{1}{2},\cdot} + ()_{\cdot-\frac{1}{2},\cdot}\},$$

$$\delta_y()_{\cdot,\cdot} = ()_{\cdot,\cdot+\frac{1}{2}} - ()_{\cdot,\cdot-\frac{1}{2}}, \qquad \mu_y()_{\cdot,\cdot} = \tfrac{1}{2}\{()_{\cdot,\cdot+\frac{1}{2}} + ()_{\cdot,\cdot-\frac{1}{2}}\},$$

where it is understood that the result of any operator is located halfway between the two input points. Then the product $\Delta_{ox}()_{i,j} \equiv \mu_x\delta_x()_{i,j}$ is a central difference $\frac{1}{2}\{()_{i+1,j} - ()_{i-1,j}\}$ located at the grid point i, j, and the product $\mu_y\delta_x()_{i,j}$, which features frequently in what follows, involves four points $(i \pm \frac{1}{2}, j \pm \frac{1}{2})$ of a square centered at i, j.

The points with integer coordinates will be called *cells*, those with one integer and one half-integer coordinate *edges*, and those with two half-integer coordinates *vertices*. The variables stored at cells will be (p, u, v). The same variables stored at vertices will be distinguished by primes where necessary. The variables stored at vertical edges will be (P, U) and those stored at horizontal edges (Q, V), where both P and Q are approximations to the pressure. In a finite-volume interpretation the edge quantities are the fluxes (see Figure 1).

If it is recognized that each application of the above operators changes the centering of the mesh values to a different set of grid points, then arbitrarily long products of operators are allowed and all multiplications commute. It is simple to prove analogues of differential identities. For example, using $\langle\cdot,\cdot\rangle$ to denote an inner product,

$$\begin{aligned}\langle p, \delta_x U\rangle &\equiv \sum_{i,j} h^2 p_{i,j}\left(U_{i+\frac{1}{2},j} - U_{i-\frac{1}{2},j}\right)\\ &= -\sum_{i,j} h^2\left(p_{i+1,j} - p_{i,j}\right)U_{i+\frac{1}{2},j} \quad\equiv\quad -\langle\delta_x p, U\rangle\end{aligned}$$

and a similar argument leads to $\langle p, \mu_x U\rangle = \langle\mu_x p, U\rangle$. Thence we have

$$\langle p, \mu_x\delta_x u\rangle \equiv \langle p, \Delta_{ox} u\rangle = -\langle\Delta_{ox} p, u\rangle, \text{ etc.}$$

Also we can combine cell and vertex values; for example,

$$\begin{aligned}\langle p, \mu_y\delta_x u'\rangle &\equiv \sum_{i,j} \tfrac{1}{2}h^2 p_{i,j}(u'_{i+\frac{1}{2},j+\frac{1}{2}} - u'_{i-\frac{1}{2},j+\frac{1}{2}} + u'_{i+\frac{1}{2},j-\frac{1}{2}} - u'_{i-\frac{1}{2},j-\frac{1}{2}})\\ = -\sum_{i,j} \tfrac{1}{2}h^2&\left(p_{i+1,j+1} - p_{i,j+1} + p_{i+1,j} - p_{i,j}\right)u'_{i+\frac{1}{2},j+\frac{1}{2}} = -\langle\mu_y\delta_x p, u'\rangle.\end{aligned}$$

Thus inner products are meaningful provided each term has the same centering. Such products may be called compatible. Matrix-valued operators will be a useful way to describe schemes, and obey the usual rules of matrix multiplication, provided each matrix has compatible entries.

2.2. Symmetry of solutions. An obvious property of (1), (2) is that because it involves only the divergence and gradient operators, then any solution remains a solution under an arbitrary translation or rotation of the (x, y)-plane. Clearly this cannot be true of the discrete solution, but we should insist on the weaker condition that the solution remains a solution under any translation or rotation that maps the grid onto itself.

This will ensure that all truncation errors are symmetric functions of the wave numbers k_x, k_y; it would also be a desirable property (minimizing the anisotropy of the scheme) if the leading terms depended only on $(k_x^2 + k_y^2)$.

2.3. Power series form. A formal solution to the initial-value problem for (1) is

$$\mathbf{u}(t) = \mathrm{e}^{-cLt}\mathbf{u}(0). \tag{3}$$

Let e^{-cLt} be represented by its power series, separating the odd and even terms,

$$\mathrm{e}^{-cLt} = 1 - \sum_{p=1}^{p=\infty}\frac{(cLt)^{2p-1}}{(2p-1)!} + \sum_{q=1}^{q=\infty}\frac{(cLt)^{2q}}{(2q)!}.$$

We can easily verify that

$$L(L^2 - \nabla^2) = 0,$$

which implies that

$$L^{p+2} = \nabla^2 L^p, \qquad p \geq 1. \tag{4}$$

Then we can write

$$\mathbf{u}(t) = \left[1 - (cLt)\sum_{p=0}^{p=\infty}\frac{t^{2p}(c^2\nabla^2)^p}{(2p+1)!} + (cLt)^2\sum_{q=0}^{q=\infty}\frac{t^{2q}(c^2\nabla^2)^q}{(2q+2)!}\right]\mathbf{u}(0). \tag{5}$$

Truncating each sum at its first term will lead to a second-order Lax–Wendroff method. For extension to higher order it is useful to note that each term beyond the third merely represents continued applications of the Laplacian, either to L or to L^2. The structure of L is that its first row represents a divergence and its first column a gradient. As for L^2, we have

$$L^2 = \begin{bmatrix} \partial_x^2 + \partial_y^2 & 0 & 0 \\ 0 & \partial_x^2 & \partial_x\partial_y \\ 0 & \partial_x\partial_y & \partial_y^2 \end{bmatrix},$$

which displays the Laplacian and the grad-div operator. Most of these operators are ambiguous on the usual nine-point stencil of Lax–Wendroff schemes, and it is that ambiguity we seek to resolve.

2.4. Conservation form. In the cell-centered finite-volume method, discrete conservation is ensured by drawing a control volume around the grid point of interest and writing the update as an integral around this volume. In the generic case of a vector $\mathbf{U}$ of conserved variables, with fluxes $\mathbf{F}, \mathbf{G}$ in the (x, y)-directions, respectively, one has

$$h^2\left[\mathbf{U}^{n+1} - \mathbf{U}^n\right] + h\Delta t\left[\delta_x\mathbf{F}^* + \delta_y\mathbf{G}^*\right] = 0,$$

where $\mathbf{F}^*, \mathbf{G}^*$ are numerical fluxes evaluated from some formula to be determined. In the present case we can write, with $\nu = c\Delta t/h$ and following the notation of Figure 1,

$$p^{n+1} - p^n + \nu[\delta_x U + \delta_y V] = 0, \tag{6}$$

$$u^{n+1} - u^n + \nu\delta_x P = 0, \tag{7}$$

$$v^{n+1} - v^n + \nu\delta_y Q = 0. \tag{8}$$

It will usefully restrict the schemes to require that they can be written in this form. A second-order scheme of the Lax–Wendroff type follows from taking U, V, P, Q to be estimates halfway through the time step. However, we will find subsequently that it is a rather special type of conservation form that emerges from the analysis.

3. Preservation of discrete vorticity.

We will now require that some discrete measure of vorticity is preserved during a time step. There are two simple measures.

3.1. Centered vorticity. First define the "centered vorticity"

$$\zeta_{\Delta_1} = \mu_x\delta_x v - \mu_y\delta_y u = Z_1\mathbf{u}, \tag{9}$$

where $Z_1 = [0, -\mu_y\delta_y, \mu_x\delta_x]$, and require that $\zeta_{\Delta_1}^{n+1} = \zeta_{\Delta_1}^n$, so that

$$\begin{aligned} 0 &= Z_1(\mathbf{u}^{n+1} - \mathbf{u}^n) \\ &= -\nu Z_1[\delta_x U + \delta_y V, \delta_x P, \delta_y Q] \\ &= \nu\delta_x\delta_y(\mu_y P - \mu_x Q). \end{aligned}$$

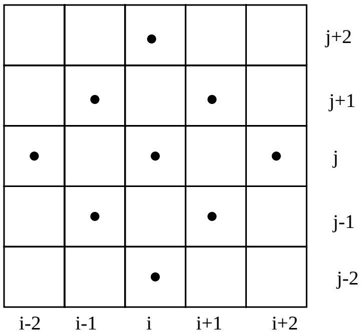


FIG. 2. *The stencil for Richtmyer's version of Lax–Wendroff.*

This measure of vorticity will therefore be preserved if the pressures assigned to the edges have the property that $\mu_y P = \mu_x Q$, and this is ensured if we take $P = \mu_x r, Q = \mu_y r$, where r is some quantity defined in cells. For consistency, it should be some local mean of the pressure, and to obey the symmetry principle (section 2.2) it must be evaluated by a symmetrical operator such as $[1 + \alpha(\delta_x^2 + \delta_y^2) + \beta\delta_x^2\delta_y^2]$. The smallest possible stencil comes from taking simply $r = p$, and hence

$$P = \mu_x r = \mu_x p, \qquad Q = \mu_y r = \mu_y p,$$

which corresponds to the simplest form of central differencing used in cell-centered finite-volume schemes, for example, [2]. However, this would lead to an unconditionally unstable scheme. Averaging just in the coordinate directions, as for the well-known Lax–Friedrichs difference scheme, leads to taking $\alpha = \frac{1}{4}, \beta = 0$.

To obtain second-order accuracy P, Q should be evaluated halfway through the time step, and to preserve vorticity we need to derive them from an r that has been evaluated halfway through the time step, so that $r = p + \frac{1}{2}\Delta t \partial_t p = p - \frac{1}{2}c\Delta t \mathrm{div}\vec{u}$.

On the most compact available stencil, and maintaining stability as for the Lax–Friedrichs scheme, this leads to

$$r = [1 + \tfrac{1}{4}(\delta_x^2 + \delta_y^2)]p - \tfrac{1}{2}\nu[\mu_x\delta_x u + \mu_y\delta_y v]$$

and hence, with $P = \mu_x r, Q = \mu_y r$, (7), (8) become

$$u^{n+1} = u^n - \nu\mu_x\delta_x\{[1 + \tfrac{1}{4}(\delta_x^2 + \delta_y^2)]p - \tfrac{1}{2}\nu[\mu_x\delta_x u + \mu_y\delta_y v]\}, \tag{10}$$

$$v^{n+1} = u^n - \nu\mu_y\delta_y\{[1 + \tfrac{1}{4}(\delta_x^2 + \delta_y^2)]p - \tfrac{1}{2}\nu[\mu_x\delta_x u + \mu_y\delta_y v]\}. \tag{11}$$

If we update the pressure with the same standard central-differences, together with a four-point averaging of p to give stability, we recover Richtmyer's form of the Lax–Wendroff method, usually written as a two-step scheme [3] (see also [4, pp. 360–365]). Its vorticity-preserving property does not seem to have been previously noticed. However, in other respects, it is less desirable. The stencil is shown in Figure 2 and can be seen to involve only points of one "parity" (value of $(i+j)(\mathrm{mod} 2)$). Therefore the method suffers from "odd-even decoupling." We do not consider this method further but turn to an alternative definition of discrete vorticity.

3.2. Compact vorticity. Next let us define the "compact vorticity"

$$\zeta_{\Delta_2} = \mu_y\delta_x v - \mu_x\delta_y u = Z_2\mathbf{u}. \tag{12}$$

Following the previous argument, we can preserve this if

$$\begin{aligned} 0 &= Z_2(\mathbf{u}^{n+1} - \mathbf{u}^n) \\ &= -\nu Z_2[\delta_x U + \delta_y V, \delta_x P, \delta_y Q] \\ &= \nu \delta_x \delta_y (\mu_x P - \mu_y Q), \end{aligned}$$

and the condition $\mu_x P = \mu_y Q$ will be met if we take $P = \mu_y r', Q = \mu_x r'$, where r' is some quantity defined at vertices. The only way to define a consistent local pressure while retaining a nine-point stencil is to take

$$r' = \mu_x \mu_y p. \tag{13}$$

In that case we have

$$P = \mu_x \mu_y^2 p, \qquad Q = \mu_x^2 \mu_y p. \tag{14}$$

To obtain second-order accuracy, r' must now be updated to halfway through the time step. The simple formula

$$r' = \mu_x \mu_y p - \tfrac{1}{2}\nu[\mu_y \delta_x u + \mu_x \delta_y v]$$

is the unique symmetrical formula to achieve this without enlarging the stencil, leading to

$$P = \mu_y r' = \mu_x \mu_y^2 p - \tfrac{1}{2}\nu[\mu_y^2 \delta_x u + \mu_x \mu_y \delta_y v], \tag{15}$$

$$Q = \mu_x r' = \mu_x^2 \mu_y p - \tfrac{1}{2}\nu[\mu_y \mu_x \delta_x u + \mu_x^2 \delta_y v]. \tag{16}$$

3.3. Two remarks on implementation. We insert here an important observation relating to the construction of nonlinear "limited" schemes that avoid nonphysical overshoots [5]. We do not attempt a thorough discussion of this issue in the present paper, because the objectives for such schemes are still unclear. For example, there is no maximum principle because waves may focus and increase in strength. However, we do note that any limiting applied to the intermediate quantity r' will still leave the vorticity exactly preserved. The limiter for such a scheme must, however, be centered on a vertex and therefore depend on values in the four neighboring cells. Schemes that are based, like most upwind schemes in current use, on one-dimensional reconstruction and interpolation cannot preserve vorticity in any of the above senses.

Our second remark deals with the application of boundary conditions, for example, to simulate the generation of acoustic waves by a moving boundary. In most finite-volume schemes one would derive from the boundary condition some expression for the unknown pressure at an interface (say, P, in Figure 1) in terms of the known normal velocity (say, U, for a vertical boundary). This would be incorrect. The correct procedure is to obtain vertex pressure p' in terms of given vertex velocities u'. If the condition is applied in this way one can easily show that vorticity at nodes adjacent to the wall is preserved; for any other procedure a vortical layer will be produced.

4. Construction and properties of evolution operator. We collect the results of section 3.2 into a prescription for the matrix operator that will update the solution, so that if

$$\mathbf{u}^{n+1} = \mathbf{u}^n - M_\Delta \mathbf{u}^n, \tag{17}$$

certain elements of the matrix M_Δ are already uniquely determined by insisting that the velocities are updated with second-order accuracy while preserving the discrete vorticity (12). We have in fact, from (7), (8), (15), (16),

$$M_\Delta = \begin{bmatrix} ?? & ?? & ?? \\ \nu\mu_x\mu_y^2\delta_x & \frac{1}{2}\nu^2\mu_y^2\delta_x^2 & \frac{1}{2}\nu^2\mu_x\mu_y\delta_x\delta_y \\ \nu\mu_x^2\mu_y\delta_y & \frac{1}{2}\nu^2\mu_x\mu_y\delta_x\delta_y & \frac{1}{2}\nu^2\mu_x^2\delta_y^2 \end{bmatrix}.$$

The adjoint property $\mathrm{div}_h = -\mathrm{grad}_h^*$ requires that this matrix be symmetric, hence

$$M_\Delta = \begin{bmatrix} ?? & \nu\mu_x\mu_y^2\delta_x & \nu\mu_x^2\mu_y\delta_y \\ \nu\mu_x\mu_y^2\delta_x & \frac{1}{2}\nu^2\mu_y^2\delta_x^2 & \frac{1}{2}\nu^2\mu_x\mu_y\delta_x\delta_y \\ \nu\mu_x^2\mu_y\delta_y & \frac{1}{2}\nu^2\mu_x\mu_y\delta_x\delta_y & \frac{1}{2}\nu^2\mu_x^2\delta_y^2 \end{bmatrix}$$

and only the $1,1$ element remains open. This can be determined by noting that the flux U, say, is implied by the above formula to be

$$U = \mu_x\mu_y^2 u = \mu_y(\mu_x\mu_y u) = \mu_y u',$$

where $u' = \mu_x\mu_y u$ is an average evaluated at the vertices, as in (13). To update this to $(n+\frac{1}{2})\Delta t$ we need to add a term $-\frac{1}{2}\Delta t\partial_x p$ which can only be $-\frac{1}{2}\nu\mu_y\delta_x p$. By considering V also, we finally arrive at

$$M_\Delta = \begin{bmatrix} \frac{1}{2}\nu^2(\mu_y^2\delta_x^2 + \mu_x^2\delta_y^2) & \nu\mu_x\mu_y^2\delta_x & \nu\mu_x^2\mu_y\delta_y \\ \nu\mu_x\mu_y^2\delta_x & \frac{1}{2}\nu^2\mu_y^2\delta_x^2 & \frac{1}{2}\nu^2\mu_x\mu_y\delta_x\delta_y \\ \nu\mu_x^2\mu_y\delta_y & \frac{1}{2}\nu^2\mu_x\mu_y\delta_x\delta_y & \frac{1}{2}\nu^2\mu_x^2\delta_y^2 \end{bmatrix}. \tag{18}$$

The scheme represented by this matrix has been uniquely determined by the requirements of conservation, vorticity preservation, symmetry of the solution under grid transformations, adjoint symmetry of the discrete operator, and second-order accuracy. Again, however, it is not a new scheme. It can be recognized by noting that M_Δ can be factored as

$$M_\Delta = \nu L_\Delta[\mu_x\mu_y I - \tfrac{1}{2}\nu L_\Delta], \tag{19}$$

where

$$L_\Delta = \begin{bmatrix} 0 & \mu_y\delta_x & \mu_x\delta_y \\ \mu_y\delta_x & 0 & 0 \\ \mu_x\delta_y & 0 & 0 \end{bmatrix}, \tag{20}$$

and therefore can be written as a two-step scheme. The operation

$$\mathbf{u}' = [\mu_x\mu_y I - \tfrac{1}{2}\nu L_\Delta]\mathbf{u}^n \tag{21}$$

gives a provisional solution at the vertices. The operation

$$\mathbf{u}^{n+1} = \mathbf{u}^n - \nu L_\Delta \mathbf{u}' \tag{22}$$

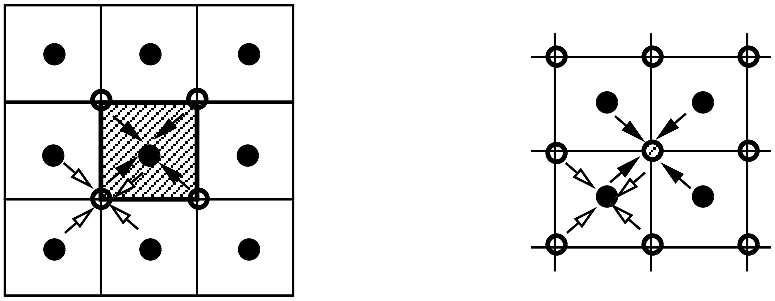


FIG. 3. *(left) The rotated Richtmyer scheme. In the first step, symbolized by white arrows, data from the cells are used to create a half time-step solution at the vertices. In the second step (black arrows) integration round the vertices updates the central cell. (right) Ni's cell-vertex scheme. In the first step (white arrows) we integrate around the cells to obtain a "cell-residual." In the second step (black arrows) these are distributed to the vertices.*

completes the update by integrating around the vertices. This is in fact the version of Lax–Wendroff known as the rotated Richtmyer scheme (see, e.g., [5, p. 125]). It is shown schematically on the left of Figure 3; comparing with Figure 2 we see the reason for the name. The original motivations for this scheme were compactness, computational economy, and stability. In the nonlinear case, as in all two-step Lax–Wendroff schemes, one avoids any multiplication by the Jacobian matrices. The vorticity-preserving property does not seem to have been previously noticed.

The second step of the method is of course a leapfrog scheme. Usually the dissipative first step is desirable to enhance stability, especially in the nonlinear case. However, we could, if we wished, construct the "pure" leapfrog method,

$$\mathbf{u}^{n+1} = \mathbf{u}^{n-1} - 2\nu L_\Delta \mathbf{u}^n, \tag{23}$$

and it is clear that this scheme would also be vorticity-preserving. So indeed would a multistage method of Runge–Kutta type [2] in which each stage employed the operator L_Δ.

4.1. Duality. Since both factors of M_Δ depend only on L_Δ they commute, and so the scheme may also be written as

$$\mathbf{u}^{n+1} - \mathbf{u}^n = -\nu[\mu_x\mu_y I - \tfrac{1}{2}\nu L_\Delta]L_\Delta \mathbf{u}^n. \tag{24}$$

In this form it is Ni's cell-vertex scheme [7], in which the variables are usually thought of as located at vertices of a grid, defining a bilinear interpolant over a square element. The first step L_Δ is to integrate $\partial_x\mathbf{F} + \partial_y\mathbf{G}$ over this element and the second step is to distribute this to the nodes of the element. It is easy to check that if some cell variable q is distributed to the nodes in the four ways shown in Figure 4, the updates to the vertices are, respectively,

$$\mu_x\mu_y q, \qquad \mu_y\delta_x q, \qquad \mu_x\delta_y q, \qquad \delta_x\delta_y q.$$

Ni's scheme was devised in the context of solving steady-state aerodynamic problems. If the transients are not required to be accurately resolved, one may consider variants of the form

$$\mathbf{u}^{n+1} - \mathbf{u}^n = -\nu[\mu_x\mu_y I - \tfrac{1}{2}\nu' L'_\Delta]L_\Delta \mathbf{u}^n,$$

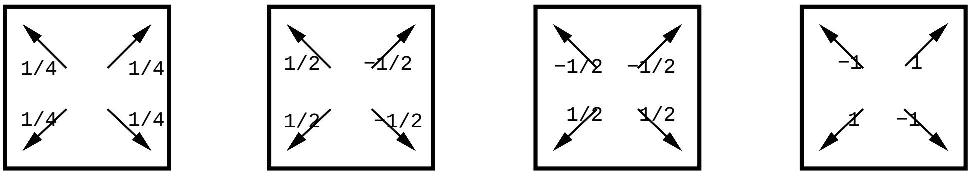


FIG. 4. *Distribution strategies.*

where ν' and L'_Δ may not be the same as ν and L_Δ. Such a scheme will not in general preserve vorticity.

However, vorticity is preserved by those schemes in which $L'_\Delta = L_\Delta$; smoother solutions are obtained by taking ν small and ν' large such that $\nu\nu' \leq 1$ (see [8] and the next section).

4.2. Energy conservation and stability. At the PDE level, we have in two dimensions the classical energy invariance

$$\begin{aligned}
\frac{d}{dt}\iint \tfrac{1}{2}(p^2+u^2+v^2)dxdy &= \iint (p\partial_t p + u\partial_t u + v\partial_t v)dxdy \\
&= -\langle p, \mathrm{div}(u,v)\rangle - \langle (u,v)^T, \mathrm{grad}p\rangle \\
&= -\langle 1, \mathrm{div}(pu,pv)\rangle \\
&= 0.
\end{aligned}$$

This proof turns on the relationship div $= -\text{grad}^*$, where $()^*$ denotes the adjoint operator.

To obtain stability in the discrete case we will require analogous discrete properties. In fact, the choices for $\mathbf{F}^*, \mathbf{G}^*$ derived above utilize discrete definitions $(\mathrm{div})_h$ and $(\mathrm{grad})_h$ which actually possess an adjoint relationship through the summation-by-parts formulae in section 2.1. In the following section we use this fact to prove energy stability of the scheme.

5. Stability and truncation error.

5.1. Stability. In this section, we consider a slight generalization of the update operator, taking

$$M_\Delta = \nu\mu_x\mu_y L_\Delta - \tfrac{1}{2}qL_\Delta^2, \tag{25}$$

where q is a parameter controlling the numerical dissipation. All schemes of this family preserve vorticity in the sense of section 3.2. In general they are first-order accurate unless $q = \nu^2$. Denoting Fourier transforms of $\mathbf{u}, M_\Delta, L_\Delta$ by $\widehat{\mathbf{u}}, \widehat{M_\Delta}, \widehat{L}_\Delta$ we write the stability condition as

$$\widehat{\mathbf{u}}^*[I - \widehat{M_\Delta^*}][I - \widehat{M_\Delta}]\widehat{\mathbf{u}} \leq \widehat{\mathbf{u}}^*\widehat{\mathbf{u}},$$

where $()^*$ denotes the Hermitian operator. We write

$$\widehat{L}_\Delta = 2\mathrm{i}\begin{bmatrix} 0 & \alpha & \beta \\ \alpha & 0 & 0 \\ \beta & 0 & 0 \end{bmatrix},$$

where $\alpha = \cos(\frac{1}{2}k_y h)\sin(\frac{1}{2}k_x h), \beta = \cos(\frac{1}{2}k_x h)\sin(\frac{1}{2}k_y h)$, and k_x, k_y are the wave numbers; note that $\widehat{L}_\Delta^* = -\widehat{L}_\Delta$. Then, abbreviating also $c_x = \cos(\frac{1}{2}k_x h), c_y = \cos(\frac{1}{2}k_y h)$ so that $\hat{\mu}_x\hat{\mu}_y = c_x c_y$, we have

$$\begin{aligned}[I - \widehat{M}_\Delta^*][I - \widehat{M}_\Delta] &= [I + \nu c_x c_y \widehat{L}_\Delta + \tfrac{1}{2}q\widehat{L}_\Delta^2][I - \nu c_x c_y \widehat{L}_\Delta + \tfrac{1}{2}q\widehat{L}_\Delta^2] \\ &= [I + (q - \nu^2 c_x^2 c_y^2)\widehat{L}_\Delta^2 + \tfrac{1}{4}q^2\widehat{L}_\Delta^4].\end{aligned}$$

After carrying out the multiplications, we have

$$\widehat{\mathbf{u}}^*[I - \widehat{M}_\Delta^*][I - \widehat{M}_\Delta]\widehat{\mathbf{u}} = \|\widehat{\mathbf{u}}\|^2 - 4[(q - \nu^2 c_x^2 c_y^2) - q^2(\alpha^2 + \beta^2)][(\alpha^2 + \beta^2)|\hat{p}|^2 + |\alpha\hat{u} + \beta\hat{v}|^2]$$

so that the stability condition is

$$(\alpha^2 + \beta^2)q^2 \le q - \nu^2 c_x^2 c_y^2 \tag{26}$$

for all modes.

Lemma 1. *The scheme is stable if*

$$\nu^2 \le q \le 1. \tag{27}$$

Proof. Necessity of the left inequality follows from considering $k_x = k_y = 0$, and necessity of the right inequality from $k_x h = \pi, k_y = 0$. Sufficiency follows from noting

$$\alpha^2 + \beta^2 = 1 - (1 - c_x^2)(1 - c_y^2) - c_x^2 c_y^2 \le 1 - c_x^2 c_y^2$$

so that

$$\begin{aligned}(\alpha^2 + \beta^2)q^2 - q + \nu^2 c_x^2 c_y^2 &\le (1 - c_x^2 c_y^2)q^2 - q + \nu^2 c_x^2 c_y^2 \\ &= (\nu^2 - q)c_x^2 c_y^2 + q(q - 1)(1 - c_x^2 c_y^2) \le 0. \qquad \square\end{aligned}$$

As well as the special choice $q = \nu^2$ leading to the second-order scheme, one may note other stable choices. Taking $q = 1$ gives a Lax–Friedrichs scheme, and $q = \nu$ gives a scheme that reduces to the first-order upwind scheme if the data is one-dimensional. It may be conjectured that this latter scheme will have an important role to play in the development of nonlinear, “limited” methods. Finally, the parameter q may be identified with the product $\nu\nu'$ mentioned in the previous section.

In the appendix we use Fourier analysis to estimate the rate of vorticity production by comparable schemes which have not been designed to preserve it. In particular, an expression is given for the vorticity production by the standard one-step Lax–Wendroff method.

5.2. Truncation error. The amplification factor that results from applying the operator L_Δ to an arbitrary Fourier mode is, using the above notation, $2\mathrm{i}\sqrt{\alpha^2 + \beta^2}$ and hence the amplification factor of the operator $1 - M_\Delta$ is, for $q = \nu^2$,

$$g = 1 - 2\mathrm{i}\nu\hat{\mu}_x\hat{\mu}_y\sqrt{\alpha^2 + \beta^2} - 2\nu^2(\alpha^2 + \beta^2).$$

This has a modulus given by

$$|g|^2 = 1 - 4\nu^2(1 - c_x^2 c_y^2)(\alpha^2 + \beta^2) + 4\nu^4(\alpha^2 + \beta^2)^2.$$

Expanding this as a Taylor series in k_x, k_y leads to

$$|g| = 1 - \frac{h^4}{8}\nu^2(1 - \nu^2)(k_x^2 + k_y^2)^2 + \mathcal{O}(h^6), \tag{28}$$

which is isotropic to leading order. The amplification error over a number of time steps proportional to h^{-1} (that is, over some fixed time independent of the grid) is $\mathcal{O}(h^3)$.

The phase error is given by the ratio of numerical to exact propagation speeds

$$\frac{\arg g}{\frac{1}{2}\sqrt{k_x^2+k_y^2 h}} = 1 - \frac{h^2}{8}(1-\nu^2)(k_x^2+k_y^2) - \frac{h^2}{24}\frac{k_x^2k_y^2}{k_x^2+k_y^2} + \mathcal{O}(h^4), \tag{29}$$

which is $\mathcal{O}(h^2)$ and is not isotropic.

6. Including advection. If the linearization is not with respect to a stationary flow but with respect to a uniform flow with velocity (u_0, v_0), the differential operator becomes

$$L^Q = \frac{1}{c}\begin{bmatrix} u_0\partial_x + v_0\partial_y & c\partial_x & c\partial_y \\ c\partial_x & u_0\partial_x + v_0\partial_y & 0 \\ c\partial_y & 0 & u_0\partial_x + v_0\partial_y \end{bmatrix}, \tag{30}$$

from which we may easily deduce that vorticity is a convected quantity,

$$(\partial_t + u_0\partial_x + v_0\partial_y)(\partial_x v - \partial_y u) = 0. \tag{31}$$

6.1. Discrete vorticity advection. This is very simply achieved. Define

$$L_\Delta^Q = \begin{bmatrix} Q_\Delta & \mu_y\delta_x & \mu_x\delta_y \\ \mu_y\delta_x & Q_\Delta & 0 \\ \mu_x\delta_y & 0 & Q_\Delta \end{bmatrix}, \tag{32}$$

where Q_Δ is any discrete operator consistent with $(u_0\partial_x + v_0\partial_y)/c$ that maps between the cell center and cell vertex meshes. Then the evolution of vorticity ζ_{Δ_2} under a first-order application of this operator is given by

$$\begin{aligned} \zeta_{\Delta_2}^{n+1} - \zeta_{\Delta_2}^{n} &= Z_2\mathbf{u}^{n+1} - Z_2\mathbf{u}^n \\ &= -\nu Z_2[Q_\Delta I + L_\Delta]\mathbf{u}' \\ &= -\nu Z_2 Q_\Delta \mathbf{u}' \\ &= -\nu Q_\Delta Z_2\mathbf{u}' \\ &= -\nu Q_\Delta \zeta'_{\Delta_2}. \end{aligned} \tag{33}$$

A Lax–Wendroff scheme defined by

$$\mathbf{u}^{n+1} - \mathbf{u}^n = -\nu L_\Delta^Q[\mu_x\mu_y I - \tfrac{1}{2}\nu L_\Delta^Q]\mathbf{u}^n$$

will lead to

$$\zeta_{\Delta_2}^{n+1} - \zeta_{\Delta_2}^{n} = -\nu Q_\Delta[\mu_x\mu_y I - \tfrac{1}{2}\nu Q_\Delta]\zeta_{\Delta_2}^n, \tag{34}$$

so that ζ_{Δ_2} is transported by the Lax–Wendroff version of Q_Δ.

6.2. Stability with advection. For consistency in our choice of difference operators we will take Q_Δ above to be

$$Q_\Delta = \frac{1}{c}(u_0 \mu_y \delta_x + v_0 \mu_x \delta_y) \tag{35}$$

so that the Fourier symbol of the update operator becomes

$$\widehat{L}^Q_\Delta = 2\mathrm{i} \begin{bmatrix} \gamma & \alpha & \beta \\ \alpha & \gamma & 0 \\ \beta & 0 & \gamma \end{bmatrix}, \tag{36}$$

where $\gamma = M_x \alpha + M_y \beta$ and $M_x = u_0/c, M_y = u_0/c$ are Mach numbers of the undisturbed flow in the (x, y)-directions.

In this section we use an alternative analysis in terms of the eigenvalues of the evolution operator, exploiting the fact that only powers of the first-order operator are involved. Suppose that $\widehat{L}^Q_\Delta$ has an eigenvalue $2\mathrm{i}\kappa$. Then the Fourier symbol of the second-order update operator

$$I - \nu \mu_x \mu_y L^Q_\Delta + \tfrac{1}{2} q (L^Q_\Delta)^2$$

has an eigenvalue

$$1 - 2\mathrm{i}\nu \hat{\mu}_x \hat{\mu}_y \kappa - 2q\kappa^2,$$

and this has modulus less than unity if

$$\kappa^2 q^2 \leq q - \nu^2 c_x^2 c_y^2. \tag{37}$$

Now, κ is actually equal to either γ or $\gamma \pm \sqrt{\alpha^2 + \beta^2}$ so that in either case

$$\kappa^2 \leq \gamma^2 + 2|\gamma|\sqrt{\alpha^2 + \beta^2} + \alpha^2 + \beta^2.$$

Using the fact that $|\gamma| \leq |M|\sqrt{\alpha^2 + \beta^2}$ we obtain

$$\kappa^2 \leq (|M| + 1)^2 (\alpha^2 + \beta^2)$$

so that from (37) the stability condition becomes

$$(1 + |M|)^2 (\alpha^2 + \beta^2) q^2 \leq q - \nu^2 c_x^2 c_y^2 \tag{38}$$

for all modes, which may be compared with (26). From the analysis of that equation it follows that the necessary and sufficient condition for stability of all modes in the presence of advection is

$$\nu^2 \leq q \leq \left(\frac{1}{1 + |M|} \right)^2. \tag{39}$$

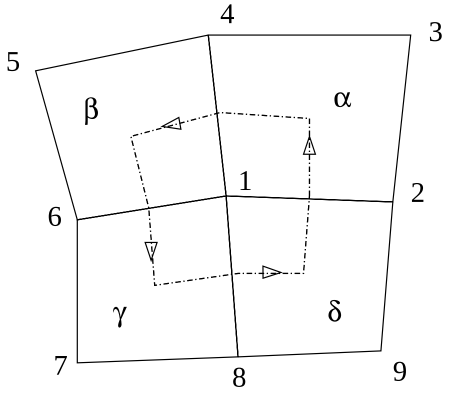


FIG. 5. *Cell and vertex numbering for a quadrilateral mesh.*

7. Irregular grids. We now turn our attention to irregular grids. Our aim is to find some measure of vorticity that is exactly preserved. More significantly, if that measure can be expressed as a line integral around some small control volume, and if many such control volumes can be united to make a large control volume, there will be a global conservation principle for circulation.

This is where the two interpretations of our scheme diverge. In the cell-centered finite-volume approach the velocities are piecewise constant within each cell so that circulation will be evaluated by a rectangle rule. In a vertex-centered finite-difference interpretation, velocities vary bilinearly within each element so that circulation will be evaluated by a trapezium rule. Even on a re-entrant path on a square grid these are equivalent so that both interpretations lead to a circulation-preserving scheme. On irregular grids the measures of circulation are different and it is not obvious that any correspondence exists.

In fact, we have only been able to prove preservation of circulation for the cell-centered finite-volume scheme, and have come to suspect rather strongly that there is no such result for the vertex-centered finite-difference method.

7.1. Quadrilateral cells. Suppose the square cells of Figure 1 are distorted to give the quadrilaterals of Figure 5, and the solution is updated by a two-step scheme as in (21), (22) with the intermediate quantities $\mathbf{u}'$ held at the vertices of the quadrilaterals. To calculate the update we suppose that $\mathbf{u}'$ is bilinearly interpolated over each quadrilateral from the vertex values. Then we show that a natural generalization of the discrete vorticity $Z_2\mathbf{u}$ is exactly preserved.

With the cell and vertex numbering of Figure 5, the update to the cell average $\mathbf{u}_\alpha$ is obtained by integrating $L\mathbf{u}'$ over the quadrilateral Ω_α of measure V_α and by applying Gauss's theorem to obtain a line integral around the perimeter of the cell. Thus for u we obtain

$$\begin{aligned} u_\alpha^{n+1} - u_\alpha^n &= -\frac{c\Delta t}{V_\alpha}\int_{\Omega_\alpha} \partial_x p' d\Omega = -\frac{c\Delta t}{V_\alpha}\oint_{\partial\Omega_\alpha} p' dy \\ &= -\frac{c\Delta t}{2V_\alpha}[(p'_1 - p'_3)(y_2 - y_4) + (p'_2 - p'_4)(y_3 - y_1)], \end{aligned} \tag{40}$$

where the integrals along each edge are given exactly by use of the trapezoidal rule. Similarly for v we obtain

$$\begin{aligned} v_\alpha^{n+1} - v_\alpha^n &= -\frac{c\Delta t}{V_\alpha}\int_{\Omega_\alpha} \partial_y p' d\Omega = \frac{c\Delta t}{V_\alpha}\oint_{\partial\Omega_\alpha} p' dx \\ &= \frac{c\Delta t}{2V_\alpha}[(p'_1 - p'_3)(x_2 - x_4) + (p'_2 - p'_4)(x_3 - x_1)]. \end{aligned} \tag{41}$$

The change in discrete vorticity at vertex 1 is then obtained by integrating the change in $\partial_x v - \partial_y u$ over an appropriate region and again turning this into a line integral of the form

$$\Delta\Gamma = \oint [(v^{n+1} - v^n)dy + (u^{n+1} - u^n)dx], \tag{42}$$

where (u, v) are constant in each quadrilateral and the line integral represents the circulation. Preservation of vorticity can be ensured by appropriately choosing a path for the line integral. We choose the part of the line in cell Ω_α to run from the midpoint of the side $(1, 2)$ to the centroid and then to the midpoint of side $(1, 4)$. In fact, since (u, v) are constant along this path, the integral is the same for any path joining the midpoints of the sides, but making it go through the centroid ensures that the control volumes can be joined together. The contribution to (42) at vertex 1 from (40) and (41) is then, after some crucial cancellation,

$$\begin{aligned} \Delta\Gamma_{1,\alpha} &= \tfrac{1}{2}[(v_\alpha^{n+1} - v_\alpha^n)(y_4 - y_2) + (u_\alpha^{n+1} - u_\alpha^n)(x_4 - x_2)] \\ &= \frac{c\Delta t}{4V_\alpha}(p'_2 - p'_4)[(y_4 - y_2)(x_3 - x_1) - (x_4 - x_2)(y_3 - y_1)] \\ &= \tfrac{1}{2}c\Delta t(p'_2 - p'_4), \end{aligned} \tag{43}$$

because the quantity in square brackets on the line above (43) is just $2V_\alpha$. It is now clear that as we add the contributions from cells $\Omega_\alpha, \Omega_\beta, \Omega_\gamma$, and Ω_δ, which meet at the vertex 1, they all cancel. Thus there is no change in the circulation around the contour chosen, and so the discrete vorticity defined by (42) and (43) is exactly preserved.

7.2. Mixed meshes. Suppose we have a mesh in which any combination of triangles and quadrilaterals meets at every vertex. Again we define the vorticity at that vertex through (42). Then, with the numbering of Figure 6, the line integrals for the update in triangular cell Ω_α can be computed as in (40), (41) to give

$$\begin{aligned} \begin{pmatrix} u \\ v \end{pmatrix}_\alpha^{n+1} - \begin{pmatrix} u \\ v \end{pmatrix}_\alpha^{n} &= -\frac{c\Delta t}{2V_\alpha}\left[p'_1\begin{pmatrix} y_2 - y_3 \\ x_3 - x_2 \end{pmatrix} + p'_2\begin{pmatrix} y_3 - y_1 \\ x_1 - x_3 \end{pmatrix} + p'_3\begin{pmatrix} y_1 - y_2 \\ x_2 - x_1 \end{pmatrix}\right] \\ &= \frac{c\Delta t}{2V_\alpha}\left[(p'_2 - p'_1)\begin{pmatrix} y_1 - y_3 \\ x_3 - x_1 \end{pmatrix} + (p'_3 - p'_1)\begin{pmatrix} y_2 - y_1 \\ x_1 - x_2 \end{pmatrix}\right]. \end{aligned} \tag{44}$$

It is also clear that the area of the triangle is given by

$$\begin{aligned} V_\alpha &= \tfrac{1}{2}[x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2)] \\ &= -\tfrac{1}{2}[y_1(x_2 - x_3) + y_2(x_3 - x_1) + y_3(x_1 - x_2)]. \end{aligned} \tag{45}$$

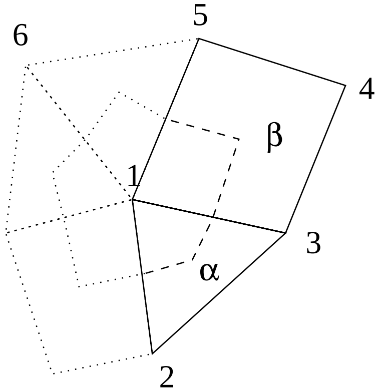


FIG. 6. *Cell and vertex numbering for a mixed mesh.*

Hence the vorticity change computed from all the triangular cells meeting at vertex 1 is in the form of a line integral,

$$\begin{aligned}&\oint [(v^{n+1}-v^n)dy+(u^{n+1}-u^n)dx]\\ &= c\Delta t \sum_{(\alpha)} \frac{1}{2V_\alpha}[(v_\alpha^{n+1}-v_\alpha^n)\Delta y_\alpha+(u_\alpha^{n+1}-u_\alpha^n)\Delta x_\alpha],\end{aligned} \tag{46}$$

into which the u and v updates are substituted from (44).

As in the quadrilateral case, we choose the line integrals to go from the midpoints of edges to the centroids of each triangle. Thus in cell Ω_α we have $\Delta x_\alpha = \frac{1}{2}(x_3+x_1)-\frac{1}{2}(x_2+x_1)=\frac{1}{2}(x_3-x_2)$. As a result, the contribution from Ω_α to the update of circulation around vertex 1 may be written

$$\begin{aligned}\Delta\Gamma_{1,\alpha} &= \frac{c\Delta t}{4V_\alpha}\left[\left\{(p_2'-p_1')(y_3-y_1)+(p_3'-p_1')(y_1-y_2)\right\}(x_3-x_2)\right.\\ &\qquad \left.+\left\{(p_2'-p_1')(x_1-x_3)+(p_3'-p_1')(x_2-x_1)\right\}(y_3-y_2)\right]\\ &= \frac{c\Delta t}{4V_\alpha}\left[p_2'\left\{(y_3-y_1)(x_3-x_2)+(x_1-x_3)(y_3-y_2)\right\}\right.\\ &\qquad \left.+p_3'\left\{(y_1-y_2)(x_3-x_2)+(x_2-x_1)(y_3-y_2)\right\}\right]\\ &= \frac{c\Delta t}{2}(p_2'-p_3').\end{aligned} \tag{47}$$

Since this is of exactly the same form as (43) the total update to the circulation around vertex 1 vanishes, and since the control volumes for any set of contiguous vertices can be merged into one large control volume as in Figure 7, the circulation around that remains unchanged also.

If cells with more than four vertices are considered, their contribution involves the values of p' at vertices that are not common to two cells and which therefore do not cancel. It seems that meshes made from a mixture of quadrilaterals and triangles are the most general for which this particular result holds.

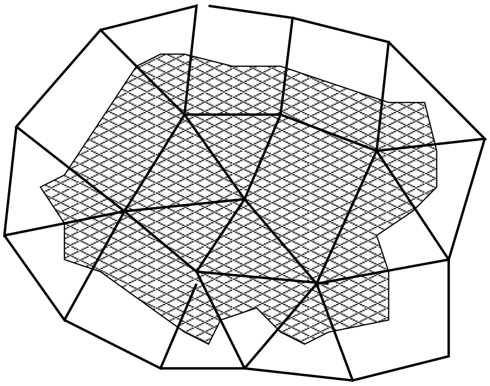


FIG. 7. *Contour for defining circulation.*

7.3. Three-dimensional unstructured grids. In three dimensions the circulation around some closed surface is the vectorial quantity

$$\vec{\Gamma} = \int\!\!\int \vec{u} \times d\vec{n}, \tag{48}$$

which evolves, for the three-dimensional version of (1), according to

$$\partial_t \vec{\Gamma} = -\int\!\!\int c\,\nabla p \times d\vec{n}, \tag{49}$$

and this vanishes if c is constant.

For finite-volume schemes, it is possible to show that a discrete version of (48) remains invariant for grids that are arbitrary combinations of tetrahedra and hexahedra if a suitable control volume is chosen around each vertex. This is constructed as shown in Figure 8 . For tetrahedral cells, one constructs the six planes defined by one edge and the midpoint of the opposite edge, thereby dissecting the tetrahedron into four hexahedra, one associated with each vertex. For hexahedral cells, one creates eight small hexahedra by constructing the three bilinear surfaces that bisect pairs of

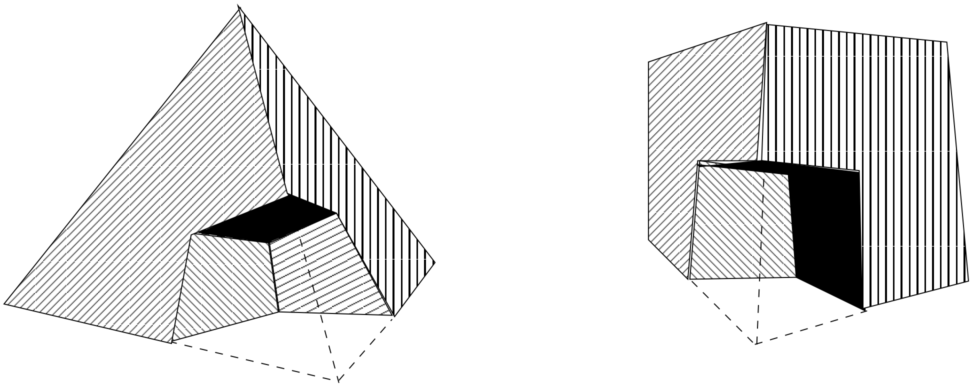


FIG. 8. *Elements of a mixed three-dimensional unstructured grid, showing a tetrahedral cell (left) and a hexahedral cell (right) with the subvolume corresponding to one vertex cut away.*

opposite faces. The control volume around any given vertex is the union of subcells so constructed. An appropriate measure for the circulation around this control volume is

$$\vec{\Gamma}_V = \sum \vec{u}_{\text{cell}} \vec{n}_{\text{cell}}, \tag{50}$$

where the sum is over all cells meeting at V, $\vec{u}_{\text{cell}}$ is the constant velocity associated with that cell, and $\vec{n}_{\text{cell}}$ is the "integrated normal" $\int\int \vec{n} dS$ over the boundary between the subvolume of the cell associated with V and the remainder of the cell. (Recall that $\vec{n}_{\text{cell}}$ depends only on the perimeter of that boundary and not on the surface spanning it; we choose a particular surface only to ensure that the control volumes fill the whole space.) We now sketch a proof that this circulation remains constant.

After some rather lengthy algebra, the contribution made by some cell α to the circulation around V can be written as a sum over a certain set of edges. Think of the cells meeting at V as defining some irregular polyhedron. The edges of this polyhedron that divide α from its neighbors may be called the boundary of α with respect to V and for the kind of grid considered will be either a triangle or a (probably nonplanar) hexahedron. Suppose that it is a hexahedron whose vertices have position vectors $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \mathbf{r}_4, \mathbf{r}_5, \mathbf{r}_6$. The relationship is

$$\Delta \vec{\Gamma}_{V,\alpha} = \frac{c\Delta t}{4} \sum_{j=1}^{j=6} (p_j + p_{j+1})(\mathbf{r}_{j+1} - \mathbf{r}_j). \tag{51}$$

From this equation it is clear that if c is a constant, the sum over the boundaries of all cells surrounding V will vanish, because every edge appears twice with opposite sign. If c is not a constant, then the sum becomes a consistent discretization of $-\int\int c\nabla p \times \vec{n}_{\text{cell}}$, and hence the circulation around any assembly of these control volumes depends only on a surface summation. Noting that the formulae remain valid when the hexahedra are degenerate, in particular when they degenerate to tetrahedra, completes the outline of the proof.

8. Variable coefficients. As an example of how the techniques of the present paper may be extended to problems with variable coefficients, we consider the case of acoustic waves in a fluid of variable density ρ. Then the system (1) is replaced by

$$\partial_t p + \rho a^2(\partial_x u + \partial_y v) = 0, \;\; \partial_t u + (1/\rho)\partial_x p = 0, \;\; \partial_t v + (1/\rho)\partial_y p = 0, \tag{52}$$

and the change in vorticity, $\zeta = \partial_x v - \partial_y u$, is given by

$$\begin{aligned} \partial_t \zeta &= -\partial_x (1/\rho)\partial_y p + \partial_y (1/\rho)\partial_x p \\ &= [\rho_x p_y - \rho_y p_x]/\rho^2. \end{aligned} \tag{53}$$

This represents the barotropic effect through which an inviscid flow may acquire vorticity, unless the pressure and density gradients are parallel. The integral version of this effect is Kelvin's theorem,

$$\partial_t \Gamma = \oint \frac{dp}{\rho}, \tag{54}$$

where $\Gamma = \oint \vec{u} \cdot \vec{ds}$ is the circulation around some given contour.

Now let us consider the update of the velocities on the quadrilateral mesh of Figure 5 with an appropriate modification of (40) and (41). In fact, let us take R_α as the constant value of $1/\rho$ in cell Ω_α so that these update formulae need to be modified only by a factor R_α. The resulting change in the discrete vorticity $Z_2\mathbf{u}$ at vertex 1 is given by replacing c by R_α in (43) and summing over the cells. The result is

$$\begin{aligned} Z_2(\mathbf{u}^{n+1}-\mathbf{u}^n) = \tfrac{1}{2}\Delta t[&R_\alpha(p'_2-p'_4)+R_\beta(p'_4-p'_6)\\ &+R_\gamma(p'_6-p'_8)+R_\delta(p'_8-p'_2)]. \end{aligned} \tag{55}$$

Moreover, it is clear that the same argument applies on the mixed grid of Figure 6, and that if we merge any set of contiguous control volumes as in Figure 7, then a discrete version of Kelvin's theorem applies to such circuits of arbitrary size.

To relate this to the local formula (53), we note that (55) can be rearranged to give

$$\begin{aligned} &\tfrac{1}{4}\Delta t\{(R_\alpha-R_\gamma)[(p'_2-p'_4)+(p'_8-p'_6)]+(R_\beta-R_\delta)[(p'_4-p'_6)+(p'_2-p'_8)]\\ &\qquad+(R_\alpha-R_\beta+R_\gamma-R_\delta)(p'_2-p'_4+p'_6-p'_8)\}. \end{aligned} \tag{56}$$

Now if the mesh in Figure 5 is roughly rectangular, $R_\alpha - R_\gamma$ is a difference of R in the NW direction while $(p'_2-p'_4)+(p'_8-p'_6)$ is an averaged difference of p' in the roughly orthogonal NE direction; and $R_\delta - R_\beta$ is a difference of R in the NE direction while $(p'_4-p'_6)+(p'_2-p'_8)$ is an averaged difference of p' in the NW direction. Thus the first line of (56) will be close to zero when the gradients of R and p' are parallel, matching the behavior of (53). The second line of (56) is a product of crossed second derivatives of R and p' and will normally be two orders smaller.

This example demonstrates how properties of vorticity preservation in the differential system, even those that hold in the presence of variable coefficients, can still be reflected in our discrete schemes. With appropriate control volumes they still hold for three-dimensional unstructured grids, as in section 7.3. The case of variable coefficients in Maxwell's equations is actually easier, because the coefficients appear inside the derivative, meaning that the magnetic field remains curl-free in an arbitrary nonresistive medium. Our method will mimic this precisely.

9. Concluding discussion.

A great step forward in CFD was the casting of discrete representations into conservation form by Lax and Wendroff [9]. This ensured that any shock waves captured by the scheme would be in some sense "correct" and greatly extended the range of application of numerical methods. Conservation of derived quantities seems to have received much less attention since the early work of Arakawa [10] and Jesperson [11] on schemes that conserve total energy and enstrophy. In particular, there seems to have been little exploration of the conservation of local derivative quantities such as vorticity which is the object of study in this paper.

By starting with the simplest case of linear problems on square grids treated by Lax–Wendroff methods, we have been able to reduce the number of candidate schemes. In fact, insisting on conservation of the primary quantities and the derivatives, together with symmetry and compactness properties, makes the scheme essentially unique. We have analyzed the stability of the scheme and find that either with or without advective terms it is stable up to the largest time steps that the CFL condition allows.

Expressing the update operator as the product of two commuting factors allows just two interpretations, depending on the order of their execution. One interpretation is as a cell-centered finite-volume scheme, whereas the other is as a vertex-centered finite-difference scheme. On irregular grids the operators no longer commute, and the two interpretations are different. We show that the finite-volume interpretation continues to preserve vorticity, but it is a finite-volume scheme of a somewhat nontraditional kind. There is a unique flux through each interface, but it may not be evaluated by reference only to the pair of cells that it separates; it must be found by averaging the fluxes at its two end points and therefore involves four cell states. Apparently this is the price to be paid for incorporating the additional conservation properties.

Whether this price will buy worthwhile benefits remains to be seen. In [12] a number of alternative nine-point schemes are given, some of which have attractive properties, but they are not vorticity-preserving. These schemes are designed by approximating the exact evolution operator for the PDE (using the bicharacteristics of the wave equation in this case) and then projecting onto piecewise constant functions. Unfortunately these approximations to the evolution operator destroy the vorticity preservation. On the other hand the schemes generated in the present paper can be regarded as evolution Galerkin methods in which Taylor expansions in time are used to approximate the evolution operator; this has the advantage of preserving vorticity. For the schemes in [12] the changes in vorticity can be estimated and shown to be quite substantial relative to other truncation error terms. Vorticity production by a general class of schemes, which includes the standard one-step Lax–Wendroff method, is presented briefly in the appendix.

Moreover, there is one rather substantial practical problem to which the current approach might be relevant. This is the appearance in many shock-capturing codes of anomalous solutions, such as the "carbuncle" that often appears ahead of blunt bodies in supersonic flow computations, first reported in [13] and recently investigated thoroughly in [14]. Closely related to this is "Quirk's phenomenon" [15] that produces an odd-even decoupling in response to small mesh perturbations, and the kinks that sometimes appear in reflected Mach stems [15, 16]. It seems very probable that these are truly weak solutions, even entropy-satisfying weak solutions, of the Euler equations, even though not usually observed in practice.[1]

Very often, perhaps always, these anomalous solutions are marked by the presence of nonphysical vorticity; in the carbuncle this accumulates into two massive counter-rotating vortices. Robinet et al. [17] note a marked correspondence between the Quirk phenomenon and a previously overlooked form of shock instability involving resonance between vortical and acoustic modes. It is therefore a reasonable speculation that a method based to some extent on the control of vorticity might be less prone to producing anomalous solutions, and might even avoid them altogether. At the moment they are usually eliminated by adding numerical dissipation in a somewhat indiscriminate manner.

But as stated earlier, we regard the present paper merely as a beginning and hope to present various extensions, both theoretical and practical, in future publications. Some extensions are very easy, such as the consideration of Maxwell's equations or ideal MHD, where the differential constraint involves the divergence rather than the curl. We have begun to consider higher-order extensions and consistent finite-element versions. We have not yet seriously thought about fully nonlinear problems or moving meshes, both of which are necessary steps to take if these ideas are indeed to have

[1] The carbuncle phenomenon can be created in a wind tunnel. One inserts a thin splitter-plate ahead of the tip of the body. A Schlieren photograph of the resulting flow appears as Plate 272 in [18]. Quite possibly the vorticity created by the boundary layer on the plate is responsible, but the high Reynolds number limit of this flow is a valid solution of the Euler equations. We conjecture that anomalous Euler solutions result from the systematic creation of vorticity due to truncation error.

practical impact on complex problems. Neither have we ourselves yet conducted any numerical experiments, although we plan on doing so soon. In the meantime we are grateful to Professor Alain Lerat who has had some experiments carried out in his laboratory when the second author was visiting there [21]: these confirm the predictions of the analysis. For initial data representing a stationary vortex they show substantial vorticity losses in a variety of alternative schemes applied to the linear acoustic system. For the present scheme, such initial data are of course perfectly preserved. Moreover, when the fully nonlinear Euler equations are used, the initial data are almost perfectly preserved by the rotated Richtmyer scheme, which is a natural generalization of our scheme to this problem.

As a final remark, we wish to draw to the attention of our readers the work of Hyman and Shashkov (see [19, 20] and the references cited therein) with which our analysis has many points of similarity. They construct discrete operators on logically rectangular meshes that obey discrete analogues of the major vector identities, such as $CURL \times GRAD = 0$, which is another way of expressing what we do here. The starting points, and the style, of the two analyses are quite different, and although our results are less specific in one sense, being limited to the single identity above, they are more general in the sense of extending to (some) unstructured and multidimensional grids. An extended comparison of the two approaches may well prove profitable.

Appendix. Vorticity production by a general scheme. We consider a class of schemes that include Lax–Wendroff schemes and compute the vorticity that they generate. The schemes are written as (only the velocity updates are needed)

$$u^{n+1} = u^n - \nu\delta_x P - \nu\delta_y Q', \tag{57}$$

$$v^{n+1} = v^n - \nu\delta_x P' - \nu\delta_y Q, \tag{58}$$

where

$$\begin{aligned} P &= [a + (1-a)\mu_y^2]\mu_x p^n - \tfrac{1}{2}\nu[b + c\mu_y^2]\delta_x u^n - \tfrac{1}{2}\nu d\mu_x\mu_y\delta_y v^n, \\ Q &= [a + (1-a)\mu_x^2]\mu_y p^n - \tfrac{1}{2}\nu[b + c\mu_x^2]\delta_y v^n - \tfrac{1}{2}\nu d\mu_x\mu_y\delta_x u^n, \\ P' &= -e\delta_x v^n, \\ Q' &= -e\delta_y u^n. \end{aligned}$$

Such a scheme is properly centered in time, and hence second-order accurate, if

$$b + c = 1, \qquad d = 1, \qquad e = 0;$$

a particular case is $a = 1, c = 0$, which corresponds to the standard one-step Lax–Wendroff scheme. For the compact vorticity of (12), in general we have

$$\begin{aligned} \zeta^{n+1} - \zeta^n &= \nu\left\{\mu_x\delta_x\delta_y P - \mu_y\delta_x\delta_y Q + \mu_x\delta_y^2 Q' - \mu_y\delta_x^2 P'\right\} \\ &= \nu\left\{\mu_x^2\delta_x\delta_y[a + (1-a)\mu_y^2]p^n - \mu_y^2\delta_x\delta_y[a + (1-a)\mu_x^2]p^n\right. \\ &\quad -\tfrac{1}{2}\nu[b + c\mu_y^2]\mu_x\delta_x^2\delta_y u^n + \tfrac{1}{2}\nu d\mu_x\mu_y^2\delta_x^2\delta_y u^n \\ &\quad -\tfrac{1}{2}\nu d\mu_x^2\mu_y\delta_x\delta_y^2 v^n + \tfrac{1}{2}\nu[b + c\mu_x^2]\mu_y\delta_x\delta_y^2 v^n \\ &\quad \left.-e\mu_x\delta_y^3 u^n + e\mu_y\delta_x^3 v^n\right\} \\ &= \nu\left\{a(\mu_x^2 - \mu_y^2)\delta_x\delta_y p^n - \tfrac{1}{2}\nu b\delta_x\delta_y(\mu_x\delta_x u^n - \mu_y\delta_y v^n)\right. \\ &\quad \left.+\tfrac{1}{2}\nu(d-c)\mu_x\mu_y\delta_x\delta_y(\mu_y\delta_x u^n - \mu_x\delta_y v^n) - e(\mu_x\delta_y^3 u^n - \mu_y\delta_x^3 v^n)\right\}. \end{aligned}$$

This formula displays four different mechanisms for vorticity production. For all of their contributions to vanish, we require

$$a = 0, \qquad b = 0, \qquad c = d, \qquad e = 0,$$

and the unique second-order scheme meeting these constraints has

$$a = b = e = 0, \qquad c = d = 1. \tag{59}$$

If we now take Fourier transforms, we have, with the notation of section 5.1,

$$\begin{aligned}\hat{\zeta}^{n+1} - \hat{\zeta}^n = \nu \{ & a(s_y^2 - s_x^2)(-4s_x s_y)\hat{p}^n \\ & -\tfrac{1}{2}\nu b(-4s_x s_y)(2\mathrm{i}c_x s_x \hat{u}^n - 2\mathrm{i}c_y s_y \hat{v}^n) \\ & +\tfrac{1}{2}\nu(d-c)(-4s_x s_y)c_x c_y(2\mathrm{i}c_y s_x \hat{u}^n - 2\mathrm{i}c_x s_y \hat{v}^n) \\ & -e(-8\mathrm{i}c_x s_y^3 \hat{u}^n + 8\mathrm{i}c_y s_x^3 \hat{v}^n)\}.\end{aligned}$$

As the mesh size h approaches zero, the left-hand side tends to h times the vorticity, $c_x, c_y \to 1, s_x \to k_x h/2, s_y \to k_y h/2$, and hence

$$\begin{aligned}\frac{\hat{\zeta}^{n+1} - \hat{\zeta}^n}{h} \simeq {} & \mathrm{i}\frac{h^2}{2}\left[\nu^2(b+c-d)k_x k_y(k_x \hat{u}^n - k_y \hat{v}^n) + e\nu(k_x^3 \hat{u}^n - k_y^3 \hat{v}^n)\right] \\ & + \frac{h^3}{4} a\nu(k_x^2 - k_y^2)k_x k_y \hat{p}^n.\end{aligned} \tag{60}$$

This shows that most first-order members of the family produce vorticity at a rate proportional to h^2, unless $b + c = d$ and $e = 0$. It also shows that most second-order members will produce vorticity at a rate proportional to h^3 unless $a = 0$; note that the standard Lax–Wendroff scheme has $a = 1$. Finally, if also $b = 0$ and $c = d$, there is no vorticity production. This corresponds to the single-parameter family studied in section 5.1.

REFERENCES

[1] P. L. Roe and E. Turkel, *The quest for diagonalization of differential systems*, in Proceedings of the Workshop on Barriers and Challenges in Computational Fluid Dynamics, 1996, ICASE/LaRC Interdiscip. Ser. Sci. Eng. 6, Kluwer, Dordrecht, The Netherlands, 1998, pp. 351–369.

[2] A. Jameson and T. J. Baker, *Multigrid Solution of the Euler Equations for Aircraft Configurations*, AIAA paper 84-0093, Reno, NV, 1984.

[3] R. D. Richtmyer, *A Survey of Difference Methods for Non-Steady Fluid Dynamics*, NCAR Tech. note 63-2, National Center for Atmospheric Research, Boulder, CO, 1962.

[4] R. D. Richtmyer and K. W. Morton, *Difference Methods for Initial-Value Problems*, Interscience, New York, 1967.

[5] P. K. Sweby, *High resolution schemes using flux limiters for hyperbolic conservation laws*, SIAM J. Numer. Anal., 21 (1984), pp. 955–1011.

[6] K. W. Morton and D. F. Mayers, *Numerical Solution of Partial Differential Equations: An Introduction*, Cambridge University Press, Cambridge, UK, New York, 1994.

[7] R.-H. Ni, *Multiple-grid scheme for solving the Euler equations*, AIAA J., 20 (1982), pp. 1565–1571.

[8] P. I. Crumpton, J. S. Mackenzie, and K. W. Morton, *Cell vertex methods for the compressible Navier-Stokes equations*, J. Comput. Phys., 109 (1993), pp. 1–15.

[9] P. D. Lax and B. Wendroff, *Systems of conservation laws*, Comm. Pure Appl. Math., 13 (1960), pp. 217–237.

[10] A. Arakawa, *A computational design for the long-term integration of the equations of atmospheric motion*, J. Comput. Phys., 1 (1966), pp. 119–143.

[11] D. C. Jesperson, *Arakawa's method is a finite-element method*, J. Comput. Phys., 16 (1974), pp. 383–390.

[12] M. Lukáčová-Medvid'ová, K. W. Morton, and G. Warnecke, *Evolution-Galerkin methods for hyperbolic systems in two space dimensions*, Math. Comp., 69 (2000), pp. 1355–1384.

[13] K. M. Peery and S. T. Imlay, *Blunt Body Flow Simulations*, AIAA paper 88-2924, 1988.

[14] M. Pandolfi and D. D'Ambrosio, *Numerical instabilities in upwind methods: Analysis and cures for the "carbuncle phenomenon,"* J. Comput. Phys., 166 (2001), pp. 271–301.

[15] J. J. Quirk, *A contribution to the great Riemann solver debate*, Internat. J. Numer. Methods Fluids, 18 (1994), pp. 555–574.

[16] J. Gressier and J.-M. Moschetta, *On the Pathological Behaviour of Upwind Schemes*, AIAA paper 98-0110, Reno, NV, 1998.

[17] J.-Ch. Robinet, J. Gressier, G. Casalis, and J.-M. Moschetta, *Shock wave instability and carbuncle phenomenon: Same intrinsic origin?*, J. Fluid Mech., 417 (2000), pp. 237–263.

[18] M. van Dyke, *An Album of Fluid Motion*, Parabolic Press, Stanford, CA, 1982.

[19] J. M. Hyman and M. Shashkov, *Natural discretizations for the divergence, gradient and curl on logically rectangular grids*, Comput. Math. Appl., 33 (1997), pp. 81–104.

[20] J. M. Hyman and M. Shashkov, *Adjoint operators for the natural discretizations of the divergence, gradient and curl on logically rectangular grids*, Appl. Numer. Math., 25 (1997), pp. 413–442.

[21] E. Douay and T. Lefebvre, *Schema de calcul numerique preservant la vorticite*, Internal report, Ecole Nationale Supérieure des Arts et Métiers, Paris, 2000.

SIAM J. SCI. COMPUT.
Vol. 23, No. 1, pp. 193–218

ALGEBRAIC MESH QUALITY METRICS[*]

PATRICK M. KNUPP[†]

Abstract. Quality metrics for structured and unstructured mesh generation are placed within an algebraic framework to form a mathematical theory of mesh quality metrics. The theory, based on the Jacobian and related matrices, provides a means of constructing, classifying, and evaluating mesh quality metrics. The Jacobian matrix is factored into geometrically meaningful parts. A nodally invariant Jacobian matrix can be defined for simplicial elements using a weight matrix derived from the Jacobian matrix of an ideal reference element. Scale and orientation-invariant algebraic mesh quality metrics are defined. The singular value decomposition is used to study relationships between metrics. Equivalence of the element condition number and mean ratio metrics is proved. The condition number is shown to measure the distance of an element to the set of degenerate elements. Algebraic measures for skew, length ratio, shape, volume, and orientation are defined abstractly, with specific examples given. Two combined metrics, shape-volume and shape-volume orientation, are algebraically defined and examples of such metrics are given. Algebraic mesh quality metrics are extended to nonsimplicial elements. A series of numerical tests verifies the theoretical properties of the metrics defined.

Key words. unstructured mesh generation, mesh quality metrics, condition number, shape measures

AMS subject classification. 65M50

PII. S1064827500371499

1. Introduction.

Mesh quality metrics for assessing the results of a meshing process have been in use almost since the beginning of meshing. Metrics are or can be used in a number of ways. First, metrics can serve as mesh *requirement specifications* prior to mesh creation. Element volume, shape, and orientation in various parts of the geometric domain can be specified in advance of meshing to enable the mesh generator to select proper algorithms and concentrate on the most difficult areas. Second, mesh improvement techniques such as *smoothing, optimization,* and *edge swapping* depend heavily on the use of quality metrics. Third, metrics often serve as a *quality control* mechanism. Given a mesh, is it of sufficient quality that it can be passed on to the consumer? Nonadaptive, a priori meshing of complex geometries is difficult, especially with nonsimplicial elements. As a result, mesh quality is not assured. Consumers of meshes for adaptive purposes should also be interested in quality metrics because h-adaptive mesh refinement will rarely improve initial mesh quality. R-type adaptive procedures, in which mesh nodes are moved, can also make good use of mesh quality metrics. Given these uses, mesh quality metrics will be needed for the foreseeable future.

[*] Received by the editors May 1, 2000; accepted for publication (in revised form) December 19, 2000; published electronically June 19, 2001. This work was funded by the Department of Energy's Mathematics, Information and Computational Sciences Program (SC-31) and was performed at Sandia National Laboratories. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the U.S. Department of Energy under contract DE-ACO4-94AL85000. This work was performed by an employee of the U.S. Government or under U.S. Government contract. The U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes. Copyright is owned by SIAM to the extent not limited by these rights.

http://www.siam.org/journals/sisc/23-1/37149.html

[†] Parallel Computing Sciences Department, Sandia National Laboratories, M/S 0847, P.O. Box 5800, Albuquerque, NM 87185-0847 (pknupp@sandia.gov).

For the most part, mesh quality metrics are based on geometric criteria. For example, does a given element possess positive volume and a good shape? Element volume, aspect ratio, skew, angles, stretching, and orientation are common geometric quality metrics. Surprisingly, a mathematical theory of geometric mesh quality metrics has not been developed until now. Such a theory should include a discussion of what a mesh quality metric is, what properties it should possess, a capability for analyzing and classifying various metrics, including a way to show how metrics are related and a means of identifying redundant metrics. This attempt at such a theory is based on element Jacobian matrices and an algebraic framework that uses matrix norm, trace, and determinant. A crucial feature introduced in this theory is the idea that metrics don't exist in a vacuum but need to be referenced to an ideal element. The metric then measures the deviation from the ideal. The ideal may vary from one application to another. For example, some applications can do well with isotropic elements while others may need anisotropic elements with particular orientations. We thus construct our theory for arbitrary reference elements.

We do not attempt a comprehensive survey of all the work that has been done on metrics but refer the reader to the early work of Robinson on quality metrics for quadrilaterals [22], [23], [24], the distortion measure of Oddy [20], the "flatness" measure of Ives [11], the summary of tetrahedral measures in [6], [21], the measures in Canann, Tristano, and Staten [3], and the paper [7]. The work reported here is an extension of the ideas of the author presented in [13], [14], [8], [15], and [16] in which the use of matrices, norms, and the condition number for mesh quality measures were introduced.

2. Preliminary observations. For both structured and unstructured meshes we can refer to mesh *nodes* and mesh *elements*. A mesh element is a geometric object topologically equivalent to some geometrically regular object such as a cube/square, tetrahedron/triangle, wedge, or pyramid. The boundary of the element is defined in terms of mesh nodes with given spatial coordinates.[1] Given a mesh element we define an element quality metric as follows.

Definition. *An element* quality metric *is a scalar function of node positions that measures some geometric property of the element.*

If a three-dimensional element has K nodes with coordinates $x_k \in R^3$, $k = 0, 1, \ldots, K-1$, then we denote a mesh quality metric by $\hat{f} \mid R^{3K} \to R$.

A host of mesh quality metrics have been defined over the years. Many of the metrics are redundant. Others may lack one or more of the following properties of quality metrics (also given in Table 1).

Definitions. *A metric is* dimension-free *if its definition in three dimensions is an unambiguous, natural generalization of its definition in two dimensions; otherwise it is* dimension-specific*. Example: Volume metrics are dimension-free while angle metrics are dimension-specific.*

A metric is element-free *if its definition on one element type is an unambiguous, natural generalization of its definition on another element type; otherwise it is* element-specific*. Example: Maximum angle is element-free on two-dimensional elements while the ratio of quadrilateral diagonal lengths is element-specific.*

A metric on a fixed element type is domain-general *if it is meaningful over a wide range of possible shapes of the element; otherwise it is* domain-specific*. Example: Aspect ratio is domain specific. Although aspect ratio may be defined for any quadri-*

[1] In this paper attention is restricted to linear elements having no midside nodes.

lateral [24] *it is not meaningful for any shape of quadrilateral. The minimum angle of a quadrilateral is domain-general.*

A metric on a fixed element type is versatile *if it is sensitive to more than one distortion mode (e.g., skew and aspect ratio); otherwise it is* specialized. *Example: Tetrahedral shape measures are versatile while skew is specialized. Versatile metrics are useful when one does not need to know the specific mode of distortion.*

A metric is scale-free *if its value does not depend on the volume of the element; otherwise it is* scale-sensitive. *A metric is* orientation-free *if its value does not depend on the orientation of the element in space; otherwise it is* orientation-sensitive. *Example: Rectangle aspect ratio is scale-free and orientation-free. Volume is orientation-free, but scale-sensitive.*

A metric is unitless *if it has no units. Example: Aspect ratio and skew are unitless while volume is not. Unitless measures do not depend upon the physical units of the problem (such as length in feet vs. meters).*

A metric is referenced *if it incorporates an explicit comparison to a reference element, which may determine volume, shape, or orientation; otherwise it is unreferenced. Example: Aspect ratio $h/(sw)$ is referenced to a rectangle with aspect ratio $s > 0$. By necessity, referenced metrics are unitless.*

Table 1
Quality metric property summary.

Property	Restricted metric	General metric
Dimension ($n = 2$ vs. $n = 3$)	dimension-specific (e.g., only applies to $n = 2$)	dimension-free (applies to both $n = 2$ and $n = 3$)
Element type (e.g., tri or quad)	element-specific (e.g., only defined for quad)	element-free (e.g., both tri and quad)
Domain (e.g., shape of quad)	domain-specific (e.g., rectangles only)	domain-general (e.g., all quads)
Versatility (# qualities measured)	specialized (only one)	versatile (e.g., volume-shape orientation)
Element size (or volume)	scale-sensitive (size-dependent)	scale-free (size-invariant)
Orientation	orientation-sensitive (orientation-dependent)	orientation-free (orientation-invariant)
Units (of metric)	has-units (dimensional)	unitless (nondimensional)
Reference (ideal element)	unreferenced (implicit ideal)	referenced (explicit ideal)

Before proceeding, we make a few general comments. First, many of the propositions noted have trivial proofs, which are omitted. Proofs are given for less straightforward results. Second, although many of the ideas presented in this paper can be generalized, we prefer to remain concrete since the meshing application demands it. Accordingly, we work over the field of real numbers, with objects in two or three dimensions ($n = 2$ or $n = 3$). We will work primarily with simplicial elements in mind and concentrate on three dimensions since this is more difficult than two dimensions. Most results we present hold in both two and three dimensions, even though only one case or the other is presented. Differences are noted. Extension of our results to nonsimplicial elements is given in section 14. We rely heavily on results from linear algebra to develop the theory of metrics. It is important to keep in mind that our emphasis differs from that of numerical linear algebra. The matrices with which we work are 2×2 or 3×3, so efficiency of computation is not the main issue. Instead, the issue is to define algebraic metrics having the desired properties and to show how they are related.

Various sets of matrices are used extensively in our presentation. Let $\mathcal{M}_n$ be the set of all $n \times n$ real matrices. Let $\mathcal{M}_n^+$ be the set of all $n \times n$ real matrices with positive determinant. The boundary of this latter set is $\partial\mathcal{M}_n^+$, the set of all $n \times n$ singular matrices. Let $\mathcal{I}_n$ be the $n \times n$ identity matrix and $\mathcal{O}$ the $n \times n$ zero-matrix. Let $\mathcal{Z}(n)$ be the set of all matrices in $\mathcal{M}_n^+$ whose determinant is unity. Let $\mathcal{SO}(n)$ be the set of all $n \times n$ orthogonal matrices with determinant 1. Let $\mathcal{D}(n)$ be the set of all $n \times n$ nonsingular diagonal matrices and $\mathcal{U}(n)$ the set of all $n \times n$ nonsingular upper triangular matrices. Let $\mathcal{SR}(n)$ be the set of all $n \times n$ nonsingular matrices of the form $\rho\,\Theta$, where $\rho > 0$ and $\Theta \in \mathcal{SO}(n)$. Each of these sets forms one of the classical matrix groups. Recognition of these groups is important because we rely heavily upon the closure, identity, and inverse properties of these matrix groups throughout this exposition.

Consider the affine map associated with a tetrahedron. Let $x_k \in R^3$, $k = 0, 1, 2, 3$, be the coordinates of the four vertices of the tetrahedron in physical space. Let ξ_k, with $0 \le \xi_k \le 1$ and $\xi_0 + \xi_1 + \xi_2 + \xi_3 = 1$, be four logical space coordinates and define the mapping from logical space to physical space by

$$x(\xi) = \sum \xi_k \, x_k$$

with $x \in R^3$.

This can be explicitly written as

$$x = (1 - \xi_1 - \xi_2 - \xi_3)\, x_0 + \xi_1\, x_1 + \xi_2\, x_2 + \xi_3\, x_3,$$

giving

$$x = A_0\, u_0 + x_0$$

with $x = (x, y, z)^t$, $u_0 = (\xi_1, \xi_2, \xi_3)^t$, and

$$A_0 = \begin{pmatrix} x_1 - x_0 & x_2 - x_0 & x_3 - x_0 \\ y_1 - y_0 & y_2 - y_0 & y_3 - y_0 \\ z_1 - z_0 & z_2 - z_0 & z_3 - z_0 \end{pmatrix}.$$

Written in this form, one sees that x is an affine map which takes points u_0 in the right tetrahedron with node coordinates $(0,0,0)$, $(1,0,0)$, $(0,1,0)$, and $(0,0,1)$ to points in a tetrahedron in physical space with the four nodes x_k. The vector x_0 controls translation of the element while the matrix A_0 controls volume, shape, and orientation of the element. We refer to A_0 as the Jacobian matrix because the columns $x_k - x_0$ of the matrix form the Jacobian of the affine map with respect to the logical variables, i.e., $A_{ij} = dx_i/d\xi_j$. The Jacobian matrix has units of length and is, in general, nonsymmetric. The formulation above also applies to triangular elements on a surface, provided the surface has a well-defined normal at every point.

3. Geometric significance of the Jacobian matrix. The Jacobian matrix of an element is important because it is well-defined for both $n = 2$ and $n = 3$. Basing element metrics on the Jacobian matrix thus makes it easy to devise metrics that are dimension-free. Furthermore, the Jacobian matrix contains information relating to the volume, shape, and orientation of an element. This can be understood more clearly by performing the QR factorization of the Jacobian matrix. The factorization decomposes the Jacobian matrix into several matrices with clear geometric interpretations. These matrices will be used to build mesh quality metrics in sections 9, 10, 11, 12, and 14.

Let A be the Jacobian matrix and $\lambda_{ij} = [A^t A]_{ij}$ be the elements of the "metric tensor." Let $\alpha = \det(A)$. It is assumed that $0 <\mid A \mid< \infty$ and $\alpha \geq 0$. Elements with $\alpha < 0$ are *inverted* and will not be considered.

PROPOSITION 3.1. *Let A be the $n = 2$ or $n = 3$ Jacobian matrix. Then one can decompose A as follows:*

$$A = RU = \mu RS = \mu RQD = RQ\Delta,$$

where

- *$R \in \mathcal{SO}(n)$ defines "orientation,"*
- *$U = \mu S$ with $U \in \mathcal{U}(n)$ and $U_{ii} > 0$,*
- *μ is a nonnegative scalar,*
- *$S = QD$ with $S \in \mathcal{U}(n)$, $S_{11} = 1$, and S defines "shape,"*
- *$Q \in \mathcal{U}(n)$ has unit column vectors, and defines "skew,"*
- *$D, \Delta \in \mathcal{D}(n)$ defines "aspect ratio."*

Proof. We explicitly construct the factorizations for $n = 2$ and $n = 3$, as they are needed for computation of the various metrics to be defined later. For $n = 2$,

$$R = \frac{1}{\sqrt{\lambda_{11}}} \begin{pmatrix} A_{11} & -A_{21} \\ A_{21} & A_{11} \end{pmatrix},$$

$$U = \begin{pmatrix} \sqrt{\lambda_{11}} & \lambda_{12}/\sqrt{\lambda_{11}} \\ 0 & \alpha/\sqrt{\lambda_{11}} \end{pmatrix},$$

$$\mu = \sqrt{\lambda_{11}},$$

$$S = \begin{pmatrix} 1 & \lambda_{12}/\lambda_{11} \\ 0 & \alpha/\lambda_{11} \end{pmatrix},$$

$$Q = \begin{pmatrix} 1 & \lambda_{12}/\sqrt{\lambda_{11}\lambda_{22}} \\ 0 & \alpha/\sqrt{\lambda_{11}\lambda_{22}} \end{pmatrix},$$

$$D = \begin{pmatrix} 1 & 0 \\ 0 & \sqrt{\lambda_{22}/\lambda_{11}} \end{pmatrix},$$

$$\Delta = \begin{pmatrix} \sqrt{\lambda_{11}} & 0 \\ 0 & \sqrt{\lambda_{22}} \end{pmatrix}.$$

For $n = 3$, let x_{ξ_k}, $k = 1, 2, 3$, be the kth column vector of A.

$$R = \left(\frac{x_{\xi_1}}{\sqrt{\lambda_{11}}}, \frac{\lambda_{11} x_{\xi_2} - \lambda_{12} x_{\xi_1}}{\sqrt{\lambda_{11}} \mid x_{\xi_1} \times x_{\xi_2} \mid}, \frac{x_{\xi_1} \times x_{\xi_2}}{\mid x_{\xi_1} \times x_{\xi_2} \mid} \right),$$

$$U = \begin{pmatrix} \sqrt{\lambda_{11}} & \frac{\lambda_{12}}{\sqrt{\lambda_{11}}} & \frac{\lambda_{13}}{\sqrt{\lambda_{11}}} \\ 0 & \frac{|x_{\xi_1} \times x_{\xi_2}|}{\sqrt{\lambda_{11}}} & \frac{\lambda_{11}\lambda_{23}-\lambda_{12}\lambda_{13}}{\sqrt{\lambda_{11}}|x_{\xi_1} \times x_{\xi_2}|} \\ 0 & 0 & \frac{\alpha}{|x_{\xi_1} \times x_{\xi_2}|} \end{pmatrix},$$

$$\mu = \sqrt{\lambda_{11}},$$

$$S = \begin{pmatrix} 1 & \frac{\lambda_{12}}{\lambda_{11}} & \frac{\lambda_{13}}{\lambda_{11}} \\ 0 & \frac{|x_{\xi_1} \times x_{\xi_2}|}{\lambda_{11}} & \frac{\lambda_{11}\lambda_{23}-\lambda_{12}\lambda_{13}}{\lambda_{11}|x_{\xi_1} \times x_{\xi_2}|} \\ 0 & 0 & \frac{\alpha}{\sqrt{\lambda_{11}}|x_{\xi_1} \times x_{\xi_2}|} \end{pmatrix},$$

$$Q = \begin{pmatrix} 1 & \frac{\lambda_{12}}{\sqrt{\lambda_{11}\lambda_{22}}} & \frac{\lambda_{13}}{\sqrt{\lambda_{11}\lambda_{33}}} \\ 0 & \frac{|x_{\xi_1} \times x_{\xi_2}|}{\sqrt{\lambda_{11}\lambda_{22}}} & \frac{\lambda_{11}\lambda_{23}-\lambda_{12}\lambda_{13}}{\sqrt{\lambda_{11}\lambda_{33}}|x_{\xi_1} \times x_{\xi_2}|} \\ 0 & 0 & \frac{\alpha}{\sqrt{\lambda_{33}}|x_{\xi_1} \times x_{\xi_2}|} \end{pmatrix},$$

$$D = \operatorname{diag}\left(1, \frac{\sqrt{\lambda_{22}}}{\sqrt{\lambda_{11}}}, \frac{\sqrt{\lambda_{33}}}{\sqrt{\lambda_{11}}}\right),$$

$$\Delta = \operatorname{diag}(\sqrt{\lambda_{11}}, \sqrt{\lambda_{22}}, \sqrt{\lambda_{33}}). \qquad \square$$

The *orientation matrix R* rotates the first column vector of A to the x-axis (and, for $n = 3$, rotates the second column vector to the x-y plane). The *volume matrix U* contains volume and shape information about the element, but not orientation. The *scale factor* μ is the length of the first column vector in the Jacobian matrix. The *shape matrix S* contains length ratio and skew information. The *length ratio matrix D* gives the ratio of element edge lengths while the *skew* matrix Q contains information about the angles in the element. The matrices *R, S, Q,* and D have units of $(length)^0$ while U, Δ, and μ have units of $(length)^1$.

Orientation, volume, shape, length ratio, and skew are a complete list of the element properties embodied in the Jacobian matrix. Other properties such as curvature or relationships between adjacent elements are not contained in this matrix.

4. Multiple Jacobian matrices. To obtain the Jacobian matrix A_0 of the affine map in the previous section we replaced ξ_0 with $1 - \xi_1 - \xi_2 - \xi_3$. A_0 is thus referenced to the node at x_0. One could just as well refer to any of the four nodes in the tetrahedron, giving four Jacobian matrices per tetrahedral element. Let $k = 0, 1, 2, 3$ and let

$$A_k = (-1)^k \begin{pmatrix} e_{k+1,k} & e_{k+2,k} & e_{k+3,k} \end{pmatrix}$$

be the kth Jacobian matrix, where $e_{k,\ell} = \mathbf{x}_k - \mathbf{x}_\ell$ with $k \neq \ell$ and $\ell = 0, 1, 2, 3$ (note that $e_{\ell,k} = -e_{k,\ell}$). Node k has three attached edge vectors, $e_{k+1,k}$, $e_{k+2,k}$, and $e_{k+3,k}$, where the indices are taken modulo four. The $(-1)^k$ factor ensures that $\alpha_k > 0$ according to the right-hand-rule.

Definitions. Let x_k be the nodes of a simplicial element ε_n. Let the centroid of the element be

$$x_c = \frac{1}{n+1}\sum_k x_k.$$

Element translation. Let $\tilde{x}_k$ be the corresponding nodes of the element translated in space by a vector b. Then $\tilde{x}_k = x_k + b$ and the centroid of the translated element, $\tilde{x}_c$, is $\tilde{x}_c = x_c + b$.

Element scaling. Let $\tilde{x}_k$ be the corresponding nodes of an element uniformly scaled by $\rho > 0$ about the centroid. Then $\tilde{x}_k = x_c + \rho(x_k - x_c)$ and the centroid is $\tilde{x}_c = x_c$.

Element rotation. Let $\tilde{x}_k$ be the corresponding nodes of the element rotated about its centroid. Then, if the rotation is given by $\Theta \in \mathcal{SO}(n)$, $\tilde{x}_k = x_c + \Theta\,(x_k - x_c)$ and the centroid of the rotated element is $\tilde{x}_c = x_c$.

Element scaling and rotation. Let $\tilde{x}_k$ be the corresponding nodes of the scaled and rotated element. Then $\tilde{x}_k = x_c + B(x_k - x_c)$ where $B \in \mathcal{SR}(n)$ and the centroid is preserved.

Proposition 4.1. *The Jacobian matrices A_k transform under element translation, scaling, rotation, or both scaling and rotation as $\tilde{A}_k = A_k$ (translation), $\tilde{A}_k = \rho A_k$ (uniform scaling), $\tilde{A}_k = \Theta A_k$ (rotation), and $\tilde{A}_k = BA_k$ (scaling and rotation).*

Proposition 4.2. *If A_k is given, the nodal coordinates are known relative to one another but not relative to the origin of the coordinate system.*

The fact that A_k is not invariant to k would appear to be a serious obstacle to using the Jacobian matrix as a basis for measuring element quality because metrics based on A_k will vary with k.[2] This difficulty will be addressed in the next section but first we show how the four Jacobian matrices are related.

Let $M \in \mathcal{Z}(n)$ be the following constant matrix

$$M = \begin{pmatrix} 1 & 1 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \end{pmatrix}.$$

Proposition 4.3. *The set $\mathcal{I}_3, M, M^2, M^3$ is a cyclic group under matrix multiplication.*

Proposition 4.4. *The four Jacobian matrices are related to one another by $A_k = A_0\, M^k$. This can be verified by a direct calculation.*[3]

Proposition 4.5. *The Jacobian determinant α_k is invariant to k.*

Proof. This follows directly from Proposition 4.4 since the determinant of M equals 1. □

The result in Proposition 4.5 is to be expected since the volume of a tetrahedron is one-sixth of the Jacobian determinant [9], and hence α_k cannot depend on k.

[2] The matrices R, U, S, Q, D, and Δ in the factorization of A are also not invariant to k.

[3] For $n = 2$,

$$M = \begin{pmatrix} -1 & -1 \\ 1 & 0 \end{pmatrix}.$$

5. A nodally invariant Jacobian matrix. In this section we exhibit a weighted Jacobian matrix that is nodally invariant.[4] We consider linear transformations between certain simplicial elements. Figure 1 illustrates the situation in two dimensions. Three triangular elements are shown in the figure: the logical triangle, the reference triangle, and the physical triangle. The physical triangle is the triangle defined by an element of the mesh. The reference triangle is the ideal triangle one wants to obtain (for example, an equilateral triangle). The logical triangle is constructed by placing one node at the origin and the other nodes at unit lengths along Cartesian axes. The physical triangle has three Jacobian matrices A_k, defined in the previous section. Similarly, the reference triangle has three Jacobian matrices W_k, computed in the same manner. The logical triangle also has three Jacobian matrices, $\mathcal{I}_n$, M, and M^2, corresponding to $k = 0, 1, 2$. The three triangles can be related via the three matrices W_k, $T_k = A_k W_k^{-1}$, and A_k. The matrix W_k is taken to have the same units as A_k (length); therefore T_k is unitless.

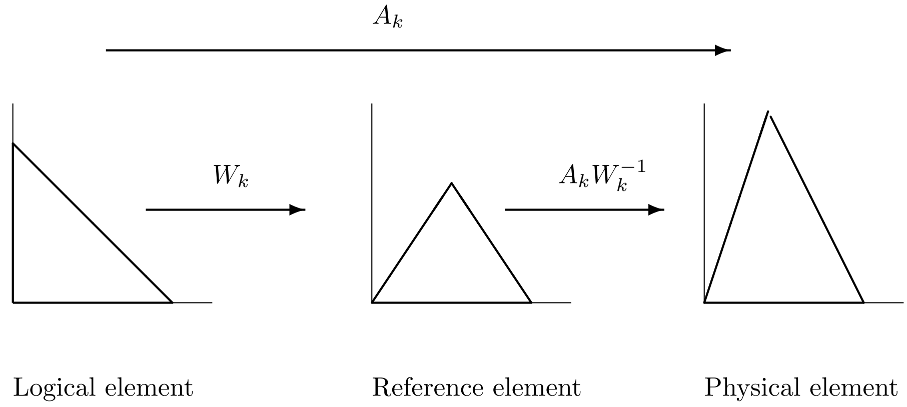


FIG. 1.

PROPOSITION 5.1. *Given any tetrahedron with Jacobian matrices A_k, $k = 0, 1, 2, 3$, let T_k be the linear transformation that takes W_k to A_k. Assume* $\det(W_k) \neq 0$. *Then $T_k = A_0\, W_0^{-1}$, that is, T_k is independent of k.*

Proof. By definition, $T_k W_k = A_k$. Proposition 4.4 applies to the matrices W_k. Thus $W_k = W_0\, M^k$. Since $A_k = A_0\, M^k$, we have the stated result. □

The matrix $T = AW^{-1}$ between the reference and physical elements does not depend on which node one chooses to compute; therefore one may use T (instead of A) to define nodally invariant element quality measures.

A consequence of the nodal invariance of T is that, unlike geometrically based tetrahedral metrics [21], we do not use all the edges of the tetrahedron, but only three (however, we also use three edges of the reference element).

From here on, then, we suppress the subscripts k, with the understanding that A and W must be computed with respect to the same node. This implies a one-to-one correspondence between the nodes of the reference element and the physical element. The matrix W is not only useful for making T nodally invariant but, as will be seen, it permits the construction of *referenced* quality metrics. Because W is derived from

[4] Some of the results of this section were foreshadowed in [19].

an ideal reference element, it is reasonable to assume that $w = \det(W) > 0$.

The following associated derived matrices are useful in a theory of quality metrics: T^t, T^{-1}, T^{-t}, $adj(T)$, T^tT (the metric matrix), $(T^tT)^{-1}$, and TT^tT.

6. Algebraic mesh quality measures. We have shown that, given W, the nodally invariant Jacobian matrix T can be computed using any node of a simplicial element. The Jacobian matrix A was factored into four matrices controlling orientation, volume, skew, and length ratio. We now turn to the question of how to build mesh quality metrics from these matrices. Determinant, trace, and norm are the most useful means to convert matrices to scalar quantities.

DEFINITION. *Let* $\tau = \det(T) = \det(AW^{-1}) = \det(A)\det(W^{-1}) = \alpha/w$.

PROPOSITION 6.1. α *and* τ *are invariant to element rotation because* $\det(\Theta T) = \det(T)$.

Another useful means to convert a matrix to a scalar is the trace function.

DEFINITION.

$$trace(T) = \sum_i T_{ii}.$$

PROPOSITION 6.2. $trace(T)$ *is a linear map from* $\mathcal{M}_n$ *to the real numbers, i.e.,* $trace(\rho T) = \rho\, trace(T)$ *and* $trace(T_1 + T_2) = trace(T_1) + trace(T_2)$.

The matrix inner product $B \cdot C$, defined in terms of the trace, is $trace(B^tC)$. For example, $A^t \cdot W^{-1} = trace(T)$. The inner product leads to the Frobenius matrix norm

$$\mid T \mid^2 = trace(T^tT).$$

The Frobenius norm is the sum of the squares of the matrix elements. The Frobenius norm is preferred for mesh quality metrics because (1) it is less expensive to compute than the p-norms and (2) many well-known mesh quality measures can be written in terms of the Frobenius norm. For some of the results in this paper it is necessary to use the 2-norm, which we will denote by $\mid T \mid_2$. The 2-norm of T is the square-root of the maximum eigenvalue of T^tT.

DEFINITION. *Let* $f \mid B_i \in \mathcal{M}_n, i = 0, \ldots, I \to R$, *be a continuous function from sets of real matrices to the real numbers. Then* f *is an* algebraic mesh quality metric *if* (1) *the matrices* B_i *are constructed from* A_k, W_k, *or factorizations thereof;* (2) *the matrices* B_i *are converted to scalars by means of the matrix norm, determinant, or trace; and* (3) f *is invariant to the element node at which the matrices are computed. The algebraic metric* f *is* referenced *if the domain of* f *is restricted to weighted matrices that make use of* W *or factorizations thereof.*

Let $\mathcal{A}$ be the set of all algebraic mesh quality metrics.

Examples of algebraic metrics are given in Table 2. They are inspired by the sources cited, but these sources did not pose the metrics in terms of the Jacobian matrix, nor were they explicitly referenced.

Algebraic mesh quality metrics are, in general, no more expensive to compute than geometrically based metrics, especially if the Frobenius norm is used.

An advantage of the algebraic metrics is that, using matrix theory and linear algebra, they are in general easier to analyze than are nonalgebraic metrics.

PROPOSITION 6.3. *Assume* $\det(W)$ *and* $\mid W \mid > 0$. *Then* (1) $\tau = 0$ *if and only if* $\alpha = 0$, (2) $\tau > 0$ *if and only if* $\alpha > 0$, (3) $\mid T \mid = 0$ *if and only if* $\mid A \mid = 0$, *and* (4) $\mid T \mid > 0$ *if and only if* $\mid A \mid > 0$.

Let $f(\{B_i\})$ be shorthand for $f(B_1, B_2, \ldots, B_I)$.

Table 2
Examples of algebraic mesh quality metrics.

Algebraic metric	Comments/source
$trace(T)$	
$T \cdot B$	with B some constant matrix
$\mid T \mid^2$	Laplace
$\mid T^t T \mid^2$	Liao [17]
τ^2	Volume [4]
$\mid adj(T) \mid^2$	Jacquotte and Cabello [12]
$\tau \mid T^{-1} \mid^2$	Winslow [2]
$\mid T \mid^2 / \tau$	Tinico-Ruiz and Barrera-Sanchez [1]
$\tau^{-4/3}\{\mid T^t T \mid^2 -(1/3) \mid T \mid^4\}$	Oddy et al. [20]
$\tau^{2/3} \mid T^{-1} \mid^2$	Nondimensional Winslow
$\tau^{-2/3} \mid T \mid^2$	Mean ratio^{-1} [18]
$\mid T \mid \mid T^{-1} \mid$	Condition number [16]

Definition. *f is* scale-invariant *if $f(\{\rho B_i\}) = f(\{B_i\})$ for $\rho > 0$.*

Example. For $n = 3$, $\tau^{-2/3} \mid T \mid^2$ is scale-invariant, while for $n = 2$ it is not.

Let $\Theta \in \mathcal{SO}(n)$. From the definition of the Frobenius norm it is easy to show that $\mid T\Theta \mid = \mid T \mid$ and $\mid \Theta T \mid = \mid T \mid$, i.e., the Frobenius norm is invariant to rotations of the element. Because of this property, many natural algebraic metrics are orientation-free.

Definition. *Let f be an algebraic metric. Then f is* orientation-invariant *if $f(\{\Theta_1 B_i \Theta_2\}) = f(\{B_i\})$ for $\Theta_1, \Theta_2 \in \mathcal{SO}(n)$.*[5]

Examples.

$$f(T) = \mid T \mid,$$
$$f(T) = \det(T).$$

Definition. *f is* scale and orientation-invariant *if $f(\{H_1 B_i H_2\}) = f(\{B_i\})$, where $H_1, H_2 \in \mathcal{SR}(n)$.*

Example. $\kappa(T) = \mid T \mid \mid T^{-1} \mid$.

Definition. *f is* positive *if $f(\{B_i\}) > 0$ for all $B_i \neq \mathcal{O}$.*

Example. $f(T) = \mid T \mid$.

Definition. *f is* even *if $f(\{-B_i\}) = f(\{B_i\})$. f is* odd *if $f(\{-B_i\}) = -f(\{B_i\})$.*

Example. $trace(T)$ is odd, $\mid T \mid$ is even, and $\det(T)$ is odd when n is odd and even when n is even.

Definition. *f is* transpose-invariant *if $f(\{B_i^t\}) = f(\{B_i\})$.*

Example. $f(T) = \mid T \mid$.

Since norm, determinant, and trace are all invariant to matrix transpose, the majority of mesh quality metrics are transpose-invariant. An example of a metric that is not transpose-invariant is $f(T) = \mid T - C \mid$, where C is an arbitrary constant matrix.

Definition. *The* conjugate *metric of $f(\{B_i\})$ is $f^*(\{B_i\}) = f(\{B_i^{-t}\})$. Note that $f^{**} = f$.*

Example. For $n = 3$, $\mu(T) = \tau^{-2/3} \mid T \mid^2$ has conjugate $\mu^*(T) = \tau^{2/3} \mid T^{-1} \mid^2$. Thus the mean ratio metric is conjugate to the modified Winslow metric.

Definition. *f is* self-conjugate *if $f^* = f$.*

[5] f is *left orientation-invariant* if $f(\{\Theta_1 B_i\} = f(\{B_i\})$. Example: $f(T) = \mid T^t T - I \mid$.

Examples.

$$f(T) = \kappa(T) = \mid T \mid \mid T^{-1} \mid,$$
$$f(T) = \tau^{-2/3} \mid T \mid^2 + \tau^{2/3} \mid T^{-1} \mid^2 .$$

7. Singular values. In this section we show that algebraic mesh quality metrics may be expressed in terms of singular values and that this provides a useful tool in analyzing properties of such metrics.

The singular value decomposition of a matrix T says there exists $\Theta, \Phi \in \mathcal{SO}(n)$ such that

$$T = \Theta^t \, D \, \Phi,$$

where $D = \text{diag}(\sigma_1, \sigma_2, \sigma_3) \in \mathcal{D}(n)$. The singular values $\sigma_k(T)$, $k = 1, 2, 3$, are real and positive. They are related to the eigenvalues of T^tT by $\sigma_k(T) = \sqrt{\lambda_k(T^tT)}$. Hence $\mid T \mid_2 = \sqrt{\lambda_{max}} = \sigma_{max}$.

Let $\sigma(T) = (\sigma_1, \sigma_2, \sigma_3)^t \in R^3$ be the vector of singular values of T. Let $\lambda(T^tT) = (\lambda_1, \lambda_2, \lambda_3)^t \in R^3$ be the vector of eigenvalues of T^tT. Then $\sigma(T)$ and $\lambda(T)$ map $T \in \mathcal{M}_n$ to vectors in R^3.

PROPOSITION 7.1. *For $\tau > 0$, $\Psi \in SO(n)$, and $\rho > 0$,*

$$\begin{aligned} \sigma(T^t) &= \sigma(T), \\ \sigma(\Psi T) &= \sigma(T), \\ \sigma(\rho T) &= \rho \, \sigma(T), \\ \sigma_k(T^{-1}) &= 1/\sigma_k(T), \\ \sigma_k(T^tT) &= \sigma_k^2(T). \end{aligned}$$

PROPOSITION 7.2.

$$\begin{aligned} \mid T \mid^2 &= \sum_k \sigma_k^2(T) = \sum_k \lambda_k(T^tT), \\ \tau &= \prod_k \sigma_k(T) = \sqrt{\prod_k \lambda_k(T^tT)}. \end{aligned}$$

Thus $\tau = 0$ if and only if $\sigma_{min} = 0$.

One can express algebraic metrics $f(T)$ as functions of the singular values $\tilde{f}(\sigma(T))$. Note that $\tilde{f}(\sigma)$ maps a vector in R^3 to a scalar. Thus $f = \tilde{f} \circ \sigma$ maps T to a scalar, i.e., it is the composition of the two maps. For example, if $\tilde{f}(\sigma) = \mid \sigma \mid^2$, then $f(T) = \mid T \mid^2$. Some other examples are

$$\begin{aligned} \mid T^tT \mid^2 &= \sum_k \sigma_k^4 = \sum_k \lambda_k^2, \\ \mid T \mid^4 = \mid \sigma \mid^4 &= \left(\sum_k \lambda_k \right)^2, \\ \mid adjT \mid^2 &= \sum_l \prod_{k \neq \ell} \sigma_k^2 = \sum_\ell \prod_{k \neq \ell} \lambda_k, \\ \mid T^{-1} \mid^2 &= \sum_k (1/\sigma_k)^2 = \sum_k 1/\lambda_k. \end{aligned}$$

Given $f(T)$, the corresponding function $\tilde{f}(\sigma)$ always exists because the SVD of T always exists. Thus, algebraic mesh quality metrics may always be expressed in terms of singular values. On the other hand, given some arbitrary function $\tilde{f}$ of σ, there may not correspond an algebraic mesh quality metric $f(T)$. For example, $\tilde{f}(\sigma) = \sum_k \sin \sigma_k$ cannot be derived from a mesh quality metric.

PROPOSITION 7.3. *$f(T) = trace(T)$ gives rise to a linear function $\tilde{f}$ of its singular values of the form $\tilde{f}(\sigma) = t \cdot \sigma$. The vector t has components $t_\ell = \sum_k \Theta_{\ell,k} \Phi_{\ell,k}$.*

DEFINITION. *f is* homogeneous of degree m *if for $\rho > 0$, $f(\{\rho\, B_i\}) = \rho^m\, f(\{B_i\})$. Metrics with no such property are inhomogeneous.*

Examples. $f(T) = \tau$ is homogeneous of degree n. $f(T) = trace(T)$ and $\mid T \mid$ are homogeneous of degree 1. $\mid T \mid^2 + \tau^2$ is inhomogeneous for both $n = 2, 3$.

PROPOSITION 7.4. *Let $f(T)$ be homogeneous of degree m. Then the product $f^* f$ is homogeneous of degree 0, i.e., scale-invariant.*

Example. $f(T) = \mid T \mid^2$ gives $(f^* f)(T) = \kappa^2(T)$.

DEFINITION. *Let $\mathcal{A}_H^m \subset \mathcal{A}$ be the set of all homogeneous algebraic metrics of degree m.*

PROPOSITION 7.5. *Let $f_1 \in \mathcal{A}_H^m$ and $f_2 \in \mathcal{A}_H^\ell$. Then $f_1 f_2 \in \mathcal{A}_H^{m+\ell}$. From this we observe that we can generate metrics having any degree of homogeneity.*

PROPOSITION 7.6. *Let $f_1 \in \mathcal{A}_H^m$ and $f_2 \in \mathcal{A}_H^m$. Then $f_1 + f_2 \in \mathcal{A}_H^m$.*

DEFINITION. *Let $\tilde{\mathcal{A}}$ be the set of functions $\tilde{f}(\sigma)$ derived from the set $\mathcal{A}$ of algebraic mesh quality metrics. Let $\tilde{\mathcal{A}}_H^m \subset \tilde{\mathcal{A}}$ be the set of functions in $\tilde{\mathcal{A}}$ that are homogeneous of degree m.*

PROPOSITION 7.7. *If $f \in \mathcal{A}_H^m$, then $\tilde{f} \in \tilde{\mathcal{A}}_H^m$.*

Proof. If $f(\{\rho B_i\}) = \rho^m\, f(\{B_i\})$, then by definition

$$\tilde{f}(\sigma(\{\rho B_i\})) = \rho^m\, \tilde{f}(\sigma(\{B_i\})).$$

However, Proposition 7.1 then implies

$$\tilde{f}(\rho\sigma) = \rho^m\, \tilde{f}(\sigma). \qquad \square$$

PROPOSITION 7.8. *If f is positive, so is $\tilde{f}$. If f is even/odd, so is $\tilde{f}$. If f is self-conjugate, so is $\tilde{f}$.*

Singular values can be used to prove two important identities which hold for Frobenius norms of 3×3 matrices:[6]

PROPOSITION 7.9.

$$\begin{aligned} \mid T^t T \mid^2 + 2\,\tau^2 \mid T^{-1} \mid^2 \;&\equiv\; \mid T \mid^4, \\ 3 \mid T \mid^2 \mid T^t T \mid^2 - \mid T \mid^6 + 6\,\tau^2 \;&\equiv\; 2 \mid T T^t T \mid^2 . \end{aligned}$$

These identities give the following bounds for $T_{3\times 3}$:

$$\begin{aligned} \mid adj T \mid \;\le\; \mid T^t T \mid \;\le\; \mid T \mid^2, \\ \mid T T^t T \mid^2 - \frac{1}{2} \mid T \mid^6 \;\le\; 3\tau^2 \;\le\; \mid T T^t T \mid^2 + \frac{1}{2} \mid T \mid^6 . \end{aligned}$$

[6] The corresponding identities for $T_{2\times 2}$ are

$$\begin{aligned} &\mid T \mid^2 \equiv \frac{1}{2} \mid T - T^t \mid^2 + trace(T^2), \\ &\mid T \mid^4 \equiv \mid T^t T \mid^2 + 2\tau^2. \end{aligned}$$

One can also relate $\kappa(T^tT)$ to $\kappa(T)$ using singular values:

$$\kappa^2(T^tT) \equiv \kappa^4(T) + 4\kappa^2(T) - 2(\tau^{-2} \mid T \mid^6 + \tau^2 \mid T^{-1} \mid^6).$$

Singular values have an important application in analyzing the equivalence of certain quality metrics.

8. Shape measures and equivalences. Tetrahedral *shape measures* for detecting distorted elements abound in the literature [6]. The list of measures includes such well-known quantities as the radius ratio [9], mean ratio [18], solid angle, and several aspect ratios [21].

A tetrahedral *shape measure* is formally defined in [6] as

> "... a continuous function that evaluates the quality of a tetrahedron. It must be invariant under translation, rotation, reflection, and uniform scaling of the tetrahedron. It must be maximum for the regular tetrahedron and it must be minimum for a degenerate tetrahedron. There is no local maximum other than the global maximum for a regular tetrahedron and there is no local minimum other than the global minimum for a degenerate tetrahedron. For the ease of comparison, it should be scaled to the interval [0,1], and be 1 for the regular tetrahedron and 0 for a degenerate tetrahedron."

This definition was used to show that mean ratio and radius ratio are shape measures while minimum dihedral angle and edge ratio are not [6].

Shape measures are clearly mesh quality metrics but, in general, they are not *algebraic* mesh quality metrics. One exception is the mean ratio shape measure η, whose definition is given in [18]:

$$\eta(T) = \frac{3\tau^{2/3}}{\mid T \mid^2}.$$

DEFINITION. *Let a, b, c be elements in a set. Recall that an equivalence relation $\sim$ on this set holds if*

- *$a \sim a$ for any a,*
- *$a \sim b$ if $b \sim a$,*
- *$a \sim b$ and $b \sim c$ implies $a \sim c$.*

DEFINITION (see Liu and Joe [18]). *Let M_1 and M_2 be tetrahedral shape measures. Then $M_1 \sim M_2$ if there exist constants $0 < c_1 \leq c_2$ and $0 < p \leq q$ such that*

$$c_1 M_1^p \leq M_2 \leq c_2 M_1^q.$$

The equivalence is strong *if $p = q$. We use the notation $M_1 \simeq M_2$ for strong equivalence.*

Informally, equivalent shape metrics sense the same shape distortions, grow large together, and grow small together. The original motivation for introducing the idea of equivalences was to reduce the list of shape measures to some manageable number. For example, the shape measures radius ratio, mean ratio, and sine of solid angle are equivalent [18].

DEFINITION. *The definition of shape measure equivalence can be generalized to include all positive algebraic mesh quality measures. The definition for the latter is the same as the former, except the phrase "tetrahedral shape measures" is replaced by "positive algebraic mesh quality measures."*

Proposition 8.1. *$\mid T\mid^2 \simeq \mid T^tT \mid$ and neither is a shape measure.*

Proof. From a well-known equivalence that can be found in [10],

$$\mid T \mid_2 \quad \leq \quad \mid T \mid \leq \sqrt{n} \ \mid T \mid_2,$$

we have

$$\mid T^tT \mid_2 \quad \leq \quad \mid T^tT \mid \leq \sqrt{n} \ \mid T^tT \mid_2,$$

that is,

$$\sigma^2_{max}(T) \leq \mid T^tT \mid \leq \sqrt{n}\, \sigma^2_{max}(T).$$

From Proposition 7.2 one can show

$$\sigma^2_{max}(T) \leq \mid T \mid^2 \leq n\, \sigma^2_{max}(T);$$

thus,

$$\frac{1}{n} \mid T \mid^2 \leq \mid T^tT \mid \leq \sqrt{n} \ \mid T \mid^2 . \qquad \square$$

Proposition 8.2. *Let $\nu > 0$ be given and M be an algebraic mesh quality measure. Then $M^\nu \simeq M$. Strong equivalence thus does not force homogeneous metrics to have the same degree of homogeneity.*

Proposition 8.3. *Let $\nu > 0$. Then $M_1 \sim M_2$ if and only if $M_1^\mu \sim M_2$.*

The statement that *if two metrics are equivalent, then it does not matter which one is used* is an exaggeration. For example,

$$\kappa(A)/\kappa(W) \leq \kappa(T) \leq \kappa(A)\,\kappa(W)$$

shows the strong equivalence of $\kappa(A)$ and $\kappa(T)$, yet the weight matrix W is a critical factor in assessing the quality of an element.

Metrics with the same degree of homogeneity need not be equivalent. For example, for $n = 3$, τ^2 and $\mid T \mid^6$ are homogeneous of degree 6 but are not equivalent.

Proposition 8.4. *$f_1 \sim f_2$ if and only if $\tilde{f_1} \sim \tilde{f_2}$.*

Proposition 8.5. *Using singular values, $\kappa^2(T) \simeq \kappa(T^tT)$, since*

$$\frac{1}{3}\,\kappa^2(T) \leq \kappa(T^tT) \leq \kappa^2(T).$$

Proposition 8.6. *Let $\kappa_2(T) = \mid T \mid_2 \mid T^{-1} \mid_2$. Then $\kappa \sim \kappa_2$.*

Proof. Using the first line of the proof of Proposition 8.1, one can readily show that

$$\kappa_2 \leq \kappa \leq n\kappa_2. \qquad \square$$

Proposition 8.7. *For $n = 3$, let $\mu(T) = \tau^{-2/3} \ \mid T \mid^2$ with conjugate $\mu^*(T) = \tau^{2/3} \mid T^{-1} \mid^2$. Then $\mu \sim \mu^* \sim \kappa$.*

Proof. Let $0 < \lambda_1 \leq \lambda_2 \leq \lambda_3$ be the eigenvalues of T^tT.

Part A. *$\mu \sim \kappa$.*

$$\mu = \frac{1 + \lambda_2/\lambda_1 + \lambda_3/\lambda_1}{[(\lambda_2/\lambda_1)(\lambda_3/\lambda_1)]^{1/3}}.$$

Therefore,

$$\begin{gathered}(\lambda_3/\lambda_1)^{1/3} \le \mu \le 3\,(\lambda_3/\lambda_1)^{2/3},\\ \kappa_2^{1/3} \le \mu \le 3\,\kappa_2^{2/3},\\ (\kappa/3)^{1/3} \le \mu \le 3^{5/3}\,(\kappa/3)^{2/3}.\end{gathered}$$

Part B. $\mu^* \sim \kappa$.

$$\mu^* = [(\lambda_2/\lambda_3)(\lambda_1/\lambda_3)]^{1/3}(1 + \lambda_3/\lambda_2 + \lambda_3/\lambda_1).$$

Therefore,

$$\begin{gathered}(\lambda_3/\lambda_1)^{1/3} \le \mu^* \le 3\,\lambda_3/\lambda_1,\\ \kappa_2^{2/3} \le \mu^* \le 3\,\kappa_2^2,\\ (\kappa/3)^{2/3} \le \mu^* \le 27\,(\kappa/3)^2.\end{gathered}$$

Then by the definition of equivalences, $\mu \sim \mu^*$. □

For $n = 2$, the corresponding scale-invariant metric is $\mu = \mid T \mid^2 / \tau$. In this case it is easy to show that $\mu = \mu^* = \kappa$.

We began this section by giving the definition of a tetrahedral shape measure. The definition is vague on the definition of a degenerate element. In the next section we fix this and define *algebraic shape metrics.*

9. Algebraic shape metrics and the condition number. We formalize the definition of a degenerate element by first defining a degenerate matrix.

DEFINITION. *Let $B \in \mathcal{M}_n^+ \cup \partial\mathcal{M}_n^+$. Then B is* degenerate *if B is singular but nonzero (i.e., $\mid B \mid > 0$ with* $\det(B) = 0$*). B is nondegenerate if* $\det B > 0$*, i.e., $B \in \mathcal{M}_n^+$. Let $\mathcal{DG}(n)$ be the set of degenerate $n \times n$ matrices. The set of singular matrices ∂M_n^+ then consists of $\mathcal{DG}(n)$ plus the zero matrix.*

DEFINITION. *A simplicial element ε_n is degenerate if and only if the matrices A_k, $k = 0, 1, \ldots, K-1$, are degenerate. Sliver elements are "near-degenerate" elements.*

PROPOSITION 9.1. *ε_n is degenerate if and only if the matrix T is degenerate.*

Proof. If ε_n is degenerate, then A_k is degenerate for all k. Since $T = A_k W_k^{-1}$, $\tau = \det(T) = \alpha_k/w_k = 0$. Hence T is singular. Suppose $T = \mathcal{O}$. Then $\mathcal{O} = A_k W_k^{-1}$, which gives $A_k = \mathcal{O}$ and $\mid A_k \mid = 0$. But since A_k is degenerate, its norm must be strictly positive. To avoid this contradiction we must have $\mid T \mid > 0$, i.e., T is degenerate. The proof in the other direction is similar. □

As a reminder, we assume here and in subsequent sections that $\alpha \ge 0$, $0 < \mid A \mid < \infty$ and that W is nondegenerate.

We return to the factorization of the Jacobian matrix discussed in section 3. As observed, the Jacobian matrix contains the following information: skew (Q), length ratio (D), shape (S), volume (U), and orientation (R). It should therefore be possible to define algebraic mesh quality metrics for each of these geometric quantities. In this section we will consider algebraic *shape* metrics. Let

$$\begin{aligned} A &= \mu R S,\\ W &= \mu_W R_W S_W. \end{aligned}$$

The shape of A will equal the shape of W if $S = S_W$. The shape of an element is a measure of element skew and aspect ratio, relative to the reference shape. We adapt the Dompierre definition of shape measures to the algebraic setting.[7]

Definition. *Let f be an algebraic mesh quality metric. Then f is an* algebraic shape metric *if*

- *the domain of f is restricted to the matrix T,*
- *f is scale and orientation invariant,*
- *$0 \leq f(T) \leq 1$ for all T,*
- *$f(T) = 1$ if and only if $T \in \mathcal{SR}(n)$,*[8]
- *$f(T) = 0$ if and only if T is degenerate.*

Proposition 9.2. *Algebraic shape metrics are invariant to uniform scalings and rotations of the physical element.*

Proof. Uniform scalings and rotations of an element mean that $A_k \to BA_k$, where $B \in \mathcal{SR}(n)$. Then $T = A_k W_k^{-1} \to BT$. But by definition, $f(BT) = f(T)$. □

Proposition 9.3. *$f(T) = n/\kappa(T)$ is an algebraic shape metric.*

Proof. The first criterion is immediate. Second, because T is nodally invariant, f is invariant to the node at which it is computed. Observe that

$$\kappa^2(T) = \sum_i \sum_j \frac{\lambda_i}{\lambda_j}$$

with λ_i the eigenvalues of T^tT. Setting $\partial\kappa/\partial\lambda_i = 0$ to find the extremum, one finds that $n \leq \kappa < \infty$; hence $0 \leq f \leq 1$. If $f = 1$, then $\kappa = n$, i.e., $\lambda_i = \lambda_j$ for all i, j. Therefore, by the singular value decomposition, $T = \lambda_i \Theta$, i.e., $T \in \mathcal{SR}(n)$. If $T \in \mathcal{SR}(n)$, then $\kappa = n$, so $f = 1$. This proves f meets the third and fourth requirements. Fifth, if T is degenerate, then $\lambda_1 = 0$ and so $\kappa \to \infty$ and $f = 0$. Finally, if $f = 0$, then $\kappa \to \infty$, and so $\lambda_1 = 0$ and $\lambda_3 > 0$, so T is degenerate. □

Similarly, one can prove $3/\mu(T)$ and $3/\mu^*(T)$ are algebraic shape metrics.[9]

The distinguishing property of the condition number is given in the following well-known result stated in Proposition 9.4 (see [5, pp. 33–34] for proof).

Proposition 9.4. *Let X and Y be 3×3 matrices with X nonsingular and $X + Y$ singular. Let*

$$d \equiv \min \{ \mid Y \mid_2 / \mid X \mid_2 \colon X + Y \mathit{singular} \}$$

be the distance between X and the set of singular matrices. The distance between X and the set of singular matrices is $1/\kappa_2(X)$.

Proposition 9.5. *$f = 3/\kappa$ is an equivalent measure of the minimum distance to the set of singular matrices.*

Proof. From Proposition 8.6, $\kappa_2 \sim \kappa$, i.e.,

$$\kappa_2 \leq \kappa \leq 3\,\kappa_2,$$

we have

$$\frac{f}{3} \leq d \leq f,$$

[7] In our definition we do not say anything about the metric lacking local minimae or maximae. The property is related to the convexity of f with respect to T. This condition, while highly desirable, is probably too restrictive in most cases, i.e., if added to the definitions, there will be no function that can satisfy all of the requirements. Numerical results in section 13 show that the metrics we suggest do not possess local extremae with respect to some parameters, but perhaps not all.

[8] This requirement forces $S = S_W$ when $f = 1$.

[9] Note that for $n = 3$ the Winslow metric $\tau \mid T^{-1} \mid^2$ is not a shape metric because it is not scale-invariant. From the definition in section 12, it is not a shape-volume metric either. This may explain why three-dimensional Winslow smoothing of structured grids has had only limited success.

i.e., f goes to zero if and only if d goes to zero. □

COROLLARY. *Since $\mathcal{DG}$ is a subset of the singular matrices, f also measures the distance to degenerate matrices and thus the distance to degenerate elements.*

10. Algebraic metrics for skew and length ratio. The algebraic shape metrics, as defined in the previous section, are invariant to the node at which they are computed. Unfortunately, the elegant way in which this is achieved by using the matrix T cannot be done for properties such as skew and length ratio. To create nodally invariant skew metrics, we can define functions that use matrices at all of the nodes.[10]

DEFINITION. *Let f be an algebraic mesh quality metric. Then f is an* algebraic skew metric *if*

- *the domain of f consists of the matrices $X_k = Q_k Q_{W_k}^{-1}$, $k = 0, 1, \ldots, K-1$, in the decompositions of A_k and W_k,*
- *$0 \le f(\{X_k\}) \le 1$ for all matrices X_k,*
- *$f(\{X_k\}) = 1$ if and only if $X_k = \mathcal{I}_n$ for all k,*
- *$f(\{X_k\}) = 0$ if and only if X_k is degenerate for at least one k.*

PROPOSITION 10.1. *Algebraic skew metrics are invariant to uniform scalings and rotations of the physical element.*

Proof. Under such an element transformation, $A_k \to BA_k$, where $B \in \mathcal{SR}(n)$. Then $Q_k = skew A_k \to skew BA_k = Q_k$. Thus X_k is unchanged under element scaling and rotation. □

PROPOSITION 10.2. *If $X_k = Q_k Q_{W_k}^{-1}$, then $f = \prod_k \frac{n}{\kappa(X_k)}$ is an algebraic skew metric.*

Proof. By construction, f is nodally invariant because it uses all nodes, so f is an algebraic metric. The remainder of the proof relies on the facts about n/κ noted in Proposition 9.3. If $f = 1$, then for all k, $X_k \in \mathcal{SR}(n)$, i.e., Q_k can differ from Q_{W_k} only by a rotation and scaling. But since these two matrics are both skew matrices, we must have $Q_k = Q_{W_k}$, hence $X_k = \mathcal{I}_n$. If $f = 0$, then $\det(X_k) = 0$ for some k. Furthermore, $\sqrt{n} \mid X_k \mid = \mid X_k \mid\mid Q_{W_k} \mid \ge \mid X_k Q_{W_k} \mid = \mid Q_k \mid = \sqrt{n} > 0$. Hence $\mid X_k \mid = 1$ and X_k is degenerate. □

PROPOSITION 10.3. $f = \min_k\{\frac{n}{\kappa(X_k)}\}$ *is an algebraic skew metric.*

DEFINITION. *Let f be an algebraic mesh quality metric. Then f is an* algebraic length ratio metric *if*

- *the domain of f consists of the matrices $X_k = D_k D_{W_k}^{-1}$, $k = 0, 1, \ldots, K-1$, in the decompositions of A_k and W_k,*
- *$0 \le f(\{X_k\}) \le 1$ for all X_k,*
- *$f(\{X_k\}) = 1$ if and only if $X_k = \mathcal{I}_n$ for all k,*
- *$f(\{X_k\}) = 0$ if and only if X_k is degenerate for at least one node.*

PROPOSITION 10.4. *Algebraic length ratio metrics are invariant to scalings and rotations of the element.*

PROPOSITION 10.5. *By this definition, $f = \prod_k n/\kappa(X_k)$ and $f = \min_k n/\kappa(X_k)$ are algebraic length ratio metrics.*

Proof. The proof is similar to that of Proposition 10.2. □

11. Algebraic metrics for volume and orientation. Roughly speaking, the orientation of an element is a measure of its orientation in space relative to the orientation of a reference element. Orientation is defined in terms of the matrix R in Proposition 3.1.

[10] Shape metrics can also be defined in this way, using $X_k = S_k S_{W_k}^{-1}$. Then if f is orientation-invariant, $f(\{X_k\}) = f(\{\mu R X_k R_W^{-1} \mu_W^{-1}\}) = f(\{A_k W_k^{-1}\}) = f(T)$.

For the definition of an orientation metric, let three diagonal matrices $\Psi^\ell \in \mathcal{SO}(3)$, $\ell = 1, 2, 3$, be defined as follows:

$$\Psi^\ell_{ij} = \begin{cases} 1, & i = j = \ell, \\ -1, & i = j \neq \ell, \\ 0, & i \neq j. \end{cases}$$

Definition. *Let f be an algebraic mesh quality metric. Then f is an* algebraic orientation metric *if*

- *the domain of f is restricted to matrix $X_0 = R_0 R_{W_0}^{-1}$,*
- *$0 \leq f(X_0) \leq 1$ for all X_0,*
- *$f(X_0) = 1$ if and only if $X_0 = \mathcal{I}_n$,*
- *$f(X_0) = 0$ if and only if $X_0 = -\mathcal{I}_n$ when $n = 2$, and $X_0 = \Psi^\ell$ for some ℓ when $n = 3$.*

The matrix R_0 is the orientation matrix in the factorization of A_0. It is assumed that $\det(R_0) = 1$ so that inverted elements are not considered. Algebraic orientation metrics are nodally invariant because the nodes on which they depend are specified (i.e., $k = 0$). However, they critically depend on the node-numbering scheme of the element (i.e., which node is numbered zero).

Proposition 11.1. *Algebraic orientation metrics are invariant to uniform scalings of the physical element.*

Proof. The proof is immediate since uniform scaling does not affect R_0 and thus X_0. □

Proposition 11.2. *$f(X_0) = 1 + (trace(X_0) - n)/4$ is an algebraic orientation metric. So is $f(X_0) = 1 - \frac{1}{8} \mid X_0 - \mathcal{I}_n \mid^2$.*

Proof. Consider the first statement. Since $X_0 \in \mathcal{SO}(n)$, $\mid X_{ii} \mid \leq 1$. Then we must have $n - 4 \leq trace(X_0) \leq n$, which gives $0 \leq f(X_0) \leq 1$. Suppose $f(X_0) = 1$. Then $traceX_0 = n$, which forces $X_0 = \mathcal{I}_n$. Suppose $traceX_0 = 0$. Then $traceX_0 = n - 4$, which, for $n = 2$, forces $X_0 = -\mathcal{I}_2$ and, for $n = 3$, forces $X_0 = \Psi^\ell$. The proof of the second statement is similar. □

The volume of an element depends both on edge lengths and element skew. A referenced volume metric is defined below.

Definition. *Let f be an algebraic mesh quality metric. Then f is an* algebraic volume metric *if*

- *the domain of f is restricted to the matrix T,*
- *f is orientation-invariant,*
- *f is homogeneous of degree n,*
- *$0 \leq f(T) < \infty$ for all T,*
- *$f(T) = 1$ if and only if $T \in Z(n)$,*
- *$f(T) = 0$ if and only if T is degenerate.*

A value of f greater (less) than one means the physical element has volume greater (less) than the volume of the reference element. Since element volume is unbounded, the upper limit of f is unbounded.

Proposition 11.3. *$f(T) = \det(T)$ is an algebraic mesh volume metric.*

If $f(T) = 1$, then $A_k = HW_k$, where $H \in \mathcal{Z}(n)$. Therefore the volume of the element is the same as the reference element, but the shape may differ.

12. Combination metrics. Combinations of the various metrics are often more useful than single metrics. Below we define algebraic volume-shape metrics.

Definition. *Let f be an algebraic mesh quality metric. Then f is an* algebraic volume-shape metric *if*

- *the domain of f is restricted to the matrix T,*
- *f is orientation-invariant,*
- $0 \leq f(T) \leq 1$ *for all T,*
- $f(T) = 1$ *if and only if $T \in \mathcal{SO}(n)$,*
- $f(T) = 0$ *if and only if T is degenerate or* $\det(T) \to \infty$.

Proposition 12.1. *Algebraic volume-shape metrics are invariant to rotations of the physical element.*

If the requirement that f be homogeneous of degree n is included in the definition of a volume-shape metric, we cannot find specific examples. For example, adding the homogeneity requirement in one attempt resulted in a discontinuous function, which is not allowed under the definition of an algebraic metric.

Proposition 12.2. *Define*

$$f(T) = \min(\tau, 1/\tau)\, n/\kappa(T).$$

Then $f(T)$ is an algebraic volume-shape metric.

Proof. f is continuous because $\lim_{\tau \to 1} f$ is the same whether one approaches from above or below. Suppose $\tau \leq 1$. Since $\det(T) \leq 1$ and $\det(T) n/\kappa(T) \leq 1$ for any T, $f(T) = 1$ forces $\det(T) = 1$ and $n/\kappa(T) = 1$. Thus $T \in \mathcal{Z}_n \cap \mathcal{SR}(n) = \mathcal{SO}(n)$. Similarly, if $\det(T) > 1$, $T \in \mathcal{SO}(n)$. If $f = 0$, then either $\tau = 0$, $\tau \to \infty$, or $n/\kappa(T) = 0$. Since $\mid T \mid > 0$, T is degenerate or $\tau \to \infty$. $\square$

f in the previous proposition is homogeneous of degree n when $\tau < 1$ and homogeneous of degree $-n$ when $\tau > 1$.

It is possible in a similar manner to define and give examples of combined shape-orientation and volume-orientation metrics. However, we skip forward to the following volume-shape orientation metric which is potentially useful in adaptive meshing schemes because it simultaneously measures element size, skew, aspect ratio, and degree of alignment (say, with a flow-field).

Definition. *Let f be an algebraic mesh quality metric. Then f is an* algebraic volume-shape orientation metric *if*

- *the domain of f is restricted to the matrices T and $X_0 = R_0 R_{W_0}^{-1}$,*
- $0 \leq f(T, X_0) \leq 1$ *for all T and X_0,*
- $f(T, X_0) = 1$ *if and only if $T = \mathcal{I}_n = X_0$,*
- $f(T) = 0$ *if and only if T is degenerate or $X_0 = -\mathcal{I}_n$ for $n = 2$ and $X_0 = \Psi^\ell$ for $\ell = 1, 2,$ or 3.*

Proposition 12.3. *Let $g(T) = n/\kappa(T)$, $h(X_0) = 1 + \frac{1}{4}(trace(X_0) - n)$, and*

$$f(T, X_0) = \min(\tau, 1/\tau)\, g(T)\, h(X_0).$$

Then $f(T, X_0)$ is an algebraic volume-shape orientation metric.

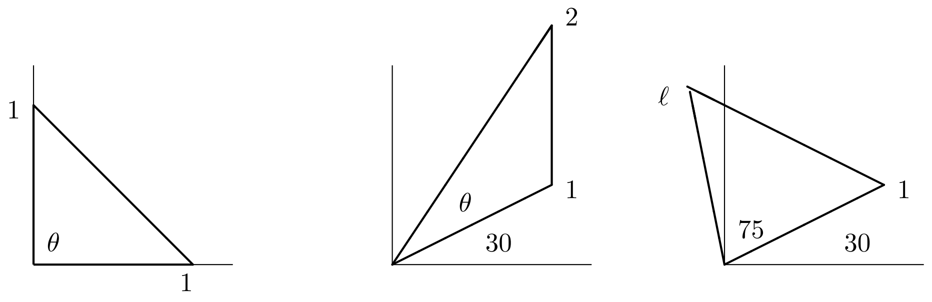
Proof. Suppose $\tau \leq 1$. Since $\det(T) \leq 1$, $n/\kappa(T) \leq 1$, and $h(X_0) \leq 1$ for any T, $f(T) = 1$ forces $\det(T) = 1$, $n/\kappa(T) = 1$, and $h(X_0) = 1$. Thus $T \in \mathcal{Z}_n \cap \mathcal{SR}(n) \cap \mathcal{I}_n = \mathcal{I}_n$. Similarly, if $\det(T) > 1$, $T = \mathcal{I}_n$. If $f = 0$, then either $\det(T) = 0$, $n/\kappa(T) = 0$, or $h(X_0) = 0$. Since $\mid T \mid > 0$, T is degenerate and $X_0 = \mathcal{I}_n$ for $n = 2$ and $X_0 = \Psi^\ell$ for $\ell = 1, 2,$ or 3. $\square$

Table 3 summarizes the metrics described so far (SC stands for scale, O for orientation, SK for skew, and AR for aspect ratio).

13. Numerical examples. We have given general definitions of algebraic metrics for simplicial elements including shape, skew, length ratio, volume, orientation, and combinations thereof. Using the specific examples given in sections 9, 10, 11,

Table 3
Summary of algebraic metrics for simplicial elements.

Metric	Invariants	Noninvariants	Example
Shape	SC, O	SK, AR	$n/\kappa(T)$
Skew	SC, O, AR	SK	$\prod_k n/\kappa(Q_k Q_{W_k}^{-1})$
Length-ratio	SC, O, SK	AR	$\prod_k n/\kappa(D_k D_{W_k}^{-1})$
Orientation	SC, SK, AR	O	$1 + (trace(X_0) - n)/4$
Volume	O, SK, AR	SC	$\det(T)$
Relative size	O, SK, AR	relative SC	$\min(\tau, 1/\tau)$
Volume-shape	O	SK, AR, relative SC	$\min(\tau, 1/\tau) n/\kappa(T)$
Volume-shape orientation	none	all	$\min(\tau, 1/\tau) n/\kappa(T)[1 + (trace(X_0) - n)/4]$



Test one Test two Test three

Fig. 2. *Triangle geometry for three test cases.*

and 12, we illustrate the behavior of these metrics with several test cases shown in Figure 2.

In the first test (see Figure 3) the metrics are plotted vs. the included angle of a triangular physical element with sides of unit length emanating from the origin. The first side lies on the x-axis, while the second side is oriented by a variable included angle. The reference triangle is the unit equilateral triangle with base on the x-axis. Figure 3 shows that all the metrics except volume vary between zero and unity, as desired. Shape and length ratio peak when the included angle matches the 60 degree angle of the reference triangle. The skew curve is not plotted, because it is nearly identical to the shape plot (because the relative lengths of sides of the physical triangle are the same as the reference triangle). The volume metric (p1-size) peaks at 1.15 when the included angle is 90 degrees, i.e., the area of the physical triangle is 1.15 times the area of the reference triangle. The orientation of the physical triangle was varied by an angle from the x-axis. The results for the orientation metric in Figure 3 show a cosine curve, which agrees with theory. The combined shape and volume metric (p1-ss) is similar to the shape metric, but less smooth and with lower values.

In the second test (Figure 4), the same physical triangle was used except that the length of the second side was increased to 2, the base of the triangle made an angle of 30 degrees with the x-axis, and the reference triangle was an isosceles triangle

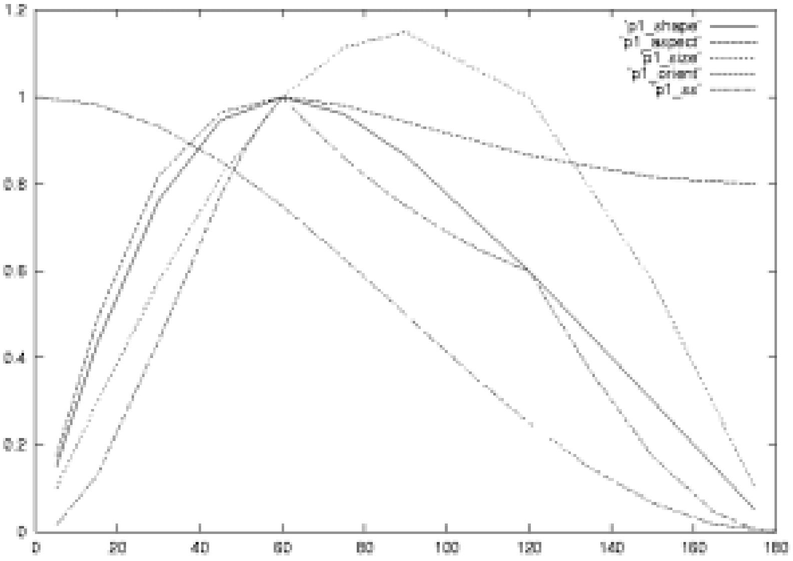


FIG. 3. *Unit equilateral reference triangle, included angle varied.*

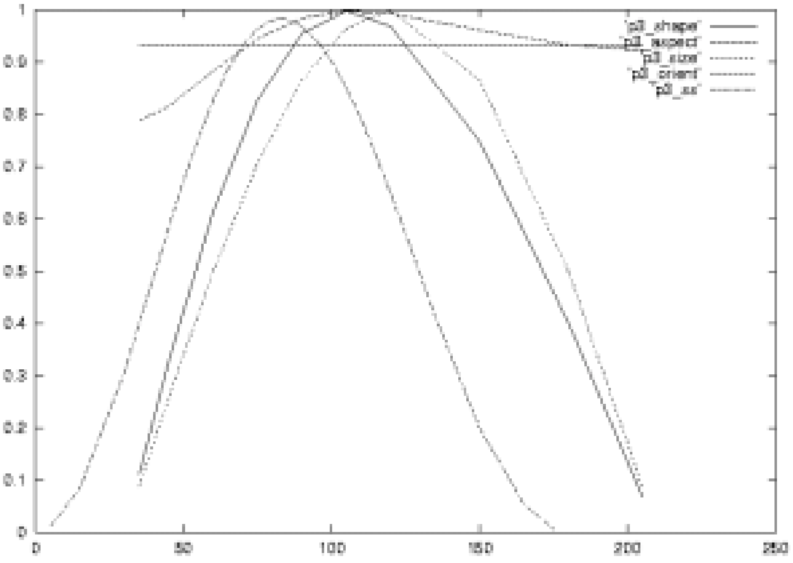


FIG. 4. *Isosceles reference triangle, included angle varied*

(perhaps describing some desired anisotropy in the mesh), with base 1 and height 2. As the included angle was varied from 0 to 180 degrees, the angle shown in the plots varied from 30 to 210 degrees. The metrics ranged between zero and unity, peaking around 105 degrees for shape and 120 degrees for volume. The skew curve again

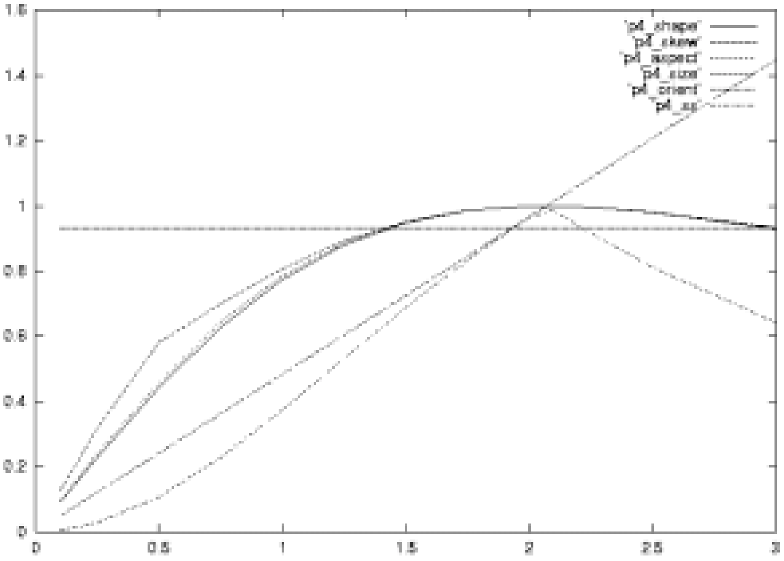


FIG. 5. *Isosceles reference triangle, second side length varied.*

overlaid the shape curve.

In the third test (Figure 5), the physical triangle had a unit length base which made an angle of 30 degrees with the x-axis. The included angle between the first and second sides was 75 degrees. The length of the second side was varied from 0 to 3. The reference triangle was the same as in the second test. The shape and skew curves differed from each other somewhat because of the differences in lengths between the physical and reference triangles. In general, however, shape, skew, and length ratio followed the same trend as one another, peaking when the second length matched the reference triangle. Volume varied linearly with the variation in the length of the second side, as expected.

In our opinion, shape, volume, and combined shape-volume are the most valuable of the metrics. Skew varies nearly the same as shape while length ratio is misleading because it is not the ratio of element width to breadth but rather the ratio of the lengths of consecutive sides. Orientation may be of use provided element nodes can be numbered in a consistent manner.

14. Nonsimplicial element metrics. Nonsimplicial elements such as quadrilaterals, hexahedra, and wedges fail to obey Propositions 4.4, 4.5, and 5.1.[11] There is no single nodally invariant matrix T which can represent all the geometric properties of nonsimplicial elements. To build algebraic quality metrics for such elements we can resort to the technique used in section 10, in which multiple matrices are used in the definition of the metric. Nonsimplicial elements for which Jacobian matrices A_k can be defined may be treated as follows. Choose a reference element and compute the reference weight matrices W_k. Let $T_k = A_k W_k^{-1}$, $k = 0, 1, \ldots, K-1$, be the

[11] Pyramids and other three-dimensional elements having more than three edges meeting in a node are still more problematic since the Jacobian matrix fails to exist.

weighted matrix, where K is the number of nodes in the element. The matrices are factored as $A_k = R_k U_k = R_k Q_k \Delta_k$ and similarly for W_k. Basic assumptions are that $\alpha_k = \det(A_k) \geq 0$, $0 < \mid A_k \mid < \infty$, and that W_k is nondegenerate for all k. The W_k should be self-consistent, i.e., computed from an element that exists.

Shape and volume metrics are defined for nonsimplicial elements, with the others left to the reader.

Definition. *Let f be an algebraic mesh quality metric. Then f is a* nonsimplicial algebraic shape metric *if*

- *the domain of f is the complete set of matrices $T_k = A_k W_k^{-1}$, $k = 0, 1, \ldots, K-1$,*
- *f is scale- and orientation-invariant,*
- *$0 \leq f(\{T_k\}) \leq 1$ for all T_k,*
- *$f(\{T_k\}) = 1$ if and only if $T_k \in \mathcal{SR}(n)$ for all k,*
- *$f(\{T_k\}) = 0$ if and only if T_k is degenerate for some k.*

Proposition 14.1. *$f(\{T_k\}) = \min_k\{n/\kappa(T_k)\}$ is an algebraic shape metric for nonsimplicial elements. So is $K/\sum_k(\kappa(T_k)/3)^2$.*

If the definition of a volume metric given for simplicial elements in section 11 is directly extended to nonsimplicial elements, the metric

$$f(\{T_k\}) = \min_k\{\det(T_k)\}$$

fails to satisfy the requirements because $f = 1$ does not force $T_k \in \mathcal{Z}(n)$ for all k. Other attempts to fix this also fail. We thus redefine algebraic volume metrics as follows.

Definition. *Let f be an algebraic mesh quality metric. Then f is an* algebraic volume metric *if*

- *the domain of f is restricted to the matrices T_k, $k = 0, 1, 2, \ldots, K-1$,*
- *f is orientation-invariant,*
- *$0 \leq f(\{T_k\}) \leq 1$ for all T_k,*
- *$f(\{T_k\}) = 1$ if and only if $T_k \in Z(n)$ for all k,*
- *$f(\{T_k\}) = 0$ if and only if T_k is degenerate for all k (or if $\det(T_k) \to \infty$ for all k).*

Proposition 14.2. *$f(\{T_k\}) = (1/K)\sum_k \min(\tau_k, 1/\tau_k)$ is an algebraic volume metric for nonsimplicial elements.*

For simplicial elements, if the value of the volume metric is say, 1/2, then either the physical element has half or twice the volume of the reference element.

The definition of volume-shape metrics given in section 12 readily extends to the nonsimplicial case.

Proposition 14.3. *$f(\{T_k\}) = (1/K)\sum_k \min(\tau_k, 1/\tau_k)\,\min_k\{n/\kappa(T_k)\}$ is an algebraic volume-shape metric for nonsimplicial elements.*

Figure 6 shows how such metrics vary for a quadrilateral element referenced to a unit square. The quadrilateral is a symmetric trapezoid, with a unit length base oriented in agreement with the reference element. The angle of the two vertical sides with respect to the base side was varied from 60 to 165 degrees.

15. Summary and conclusions. A theory of algebraic mesh quality metrics was proposed based on element Jacobian matrices. Jacobian matrices can be decomposed into geometrically meaningful factors representing element volume, orientation, and shape. The factor matrices are node-dependent and thus cannot be used to construct algebraic mesh quality measures unless all are used in a symmetric way.

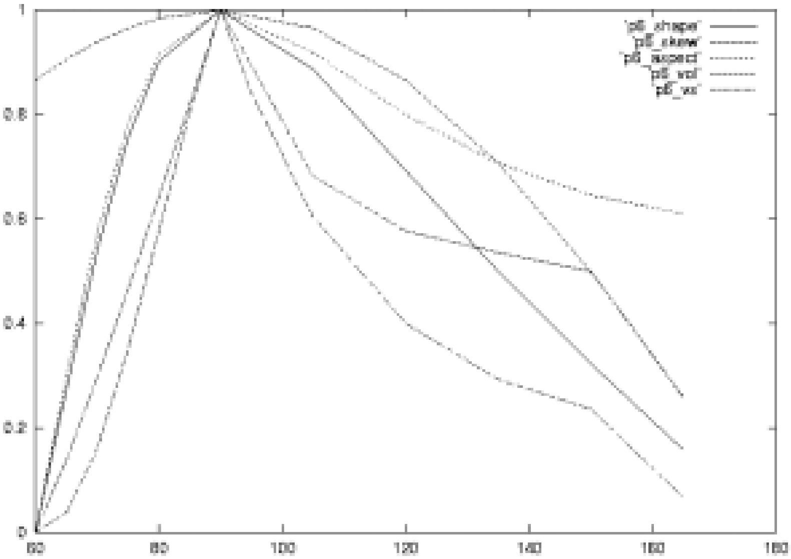


FIG. 6. *Square reference quadrilateral, trapezoid physical element.*

However, for simplicial elements one can define a single nodally invariant matrix T using the Jacobian matrices W_k of a reference element. We emphasize the point that mesh quality metrics should be explicitly referenced to a logical element. Thus, for example, shape metrics may be referenced to an isosceles, equilateral, or right-angled simplicial element, depending on the application. We list the properties which must be satisfied by an algebraic mesh quality metric. An algebraic definition of mesh quality metrics permits relatively easy analysis of the properties of a metric, for example, in terms of its singular values. Abstract definitions of metrics are given in terms of precise requirements for algebraic shape, length ratio, skew, volume, orientation, volume-shape, and volume-shape orientation metrics. The abstract definitions are slightly subjective, especially in the range and domain of the metrics, but are largely noncontroversial. The requirements in the abstract definitions clearly must be satisfied by any algebraic metric purporting to be of a particular type. Specific examples for each type of metric are given. The examples, for the most part, are conspicuous in that they are new. Few traditional metrics (even were they referenced) will qualify under the definitions given, with the notable exceptions of mean ratio and determinant. Shape, volume, and volume-shape metrics for simplicial elements can be posed in terms of the nodally invariant matrix T while the other metrics must use a set of nodally dependent matrices. Examples of volume-shape metrics are difficult to construct due to the large number of requirements they must satisfy. Volume-shape metrics are critical to adaptive meshing, and it is significant that a rigorous definition and example has been provided. Except for volume, the metrics are scaled between zero and unity for ease of comparison. Multiple Jacobian matrices are needed in the definitions of metrics for nonsimplicial elements due to the lack of an analogy to the matrix T. The rigorous definitions given for the various types of metrics have made it clear that it is not, in general, easy to devise metrics having all the right properties;

this is especially true for nonsimplicial element metrics. For example, to obtain the proper behavior for a volume metric for nonsimplicial elements, we sacrificed the homogeneity requirement. The difficulties encountered suggest that one reason why so many mesh quality metrics have been defined in the past is that few metrics satisfy all of the requirements. Although the metric definitions given require metrics to satisfy rigorous criteria to qualify being a metric of a particular type, there remains some freedom to define alternative metrics. Redundant metrics can be eliminated by investigating possible equivalences via singular values. It was shown that the algebraic shape metric, condition number, measures the distance to the set of degenerate elements. Not all geometric properties of potential interest can be given in terms of an algebraic metric. For example, nonalgebraic metrics based on solid angle or length-to-width ratios cannot be expressed as algebraic metrics. However, there seems to be little need for these additional metrics since, for example, solid-angle-based shape metrics are equivalent to the algebraically based mean ratio shape metric.

Future work may include extending the theory to higher-order finite elements having midside nodes as a means to measuring element curvature. Development of connections between algebraic element quality metrics and effects upon analysis error, efficiency, and robustness should be pursued. Finally, the metrics given are likely candidates for objective functions in mesh smoothing and optimization techniques.

Acknowledgments. The author wishes to thank David Womble, Robert Leland, Tim Tautges, Stanly Steinberg, and the late Fred Howes for their support.

REFERENCES

[1] J. G. Tinico-Ruiz and P. Barrera-Sanchez, *Area functionals in plane grid generation*, in Proceedings of the 6th International Conference on Numerical Grid Generation in Computational Field Simulations, M. Cross et al., eds., Mississippi State University, Starkville, MS, 1998, pp. 293–302.

[2] J. Brackbill and J. Saltzman, *Adaptive zoning for singular problems in two dimensions*, J. Comput. Phys., 46 (1982), pp. 342–368.

[3] S. Canann, J. Tristano, and M. Staten, *An approach to combined Laplacian and optimization-based smoothing for triangular, quadrilateral, and quad-dominant meshes*, in Proceedings of the 7th International Meshing Roundtable, Dearborn, MI, 1998, Sandia Report SAND 98-2250, Sandia National Laboratories, Albuquerque, NM, 1998, pp. 479–494.

[4] C. L. Chen, K. Y. Szema, and S. R. Chakravarthy, *Optimization of unstructured grid*, AIAA 95-0217, in Proceedings of the 33rd Aerospace Sciences Meeting, Reno, NV, American Institute of Aeronautics and Astronautics, 1995.

[5] J. W. Demmel, *Applied Numerical Linear Algebra*, SIAM, Philadelphia, 1997.

[6] J. Dompierre, P. Labbe, F. Guibault, and R. Camerero, *Proposal of benchmarks for* 3*D unstructured tetrahedral mesh optimization*, in Proceedings of the 7th International Meshing Roundtable, Dearborn, MI, 1998, Sandia Report SAND 98-2250, Sandia National Laboratories, Albuquerque, NM, 1998, pp. 459–478.

[7] D. Field, *Qualitative measures for initial meshes*, Internat. J. Numer. Methods Engrg., 47 (2000), pp. 887–906.

[8] L. Freitag and P. Knupp, *Tetrahedral element shape optimization via the Jacobian determinant and condition number*, in Proceedings of the 8th International Meshing Roundtable, Lake Tahoe, CA, 1999, Sandia Report SAND 99-2288, Sandia National Laboratories, Albuquerque, NM, 1999, pp. 247–258.

[9] P. L. George and H. Borouchaki, *Delaunay Triangulation and Meshing*, Hermes, Paris, 1998.

[10] G. Golub and C. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, 1983.

[11] D. Ives, *Geometric grid generation*, in Surface Modeling, Grid Generation, and Related Issues in Computational Fluid Dynamic (CFD) Solutions, NASA CP-3291, NASA Lewis Research

Center, Cleveland, OH, 1995.

[12] O. P. Jacquotte and J. Cabello, *A variational method for the optimization and adaption of grids in computational fluid dynamics*, in Numerical Grid Generation in Computational Fluid Mechanics '88, S. Sengupta et. al., eds., PineRidge Press, Swansea, UK, 1988, pp. 405–413.

[13] P. M. Knupp and N. Robidoux, *A framework for variational grid generation: Conditioning the Jacobian matrix with matrix norms*, SIAM J. Sci. Comput., 21 (2000), pp. 2029–2047.

[14] P. Knupp, *Matrix norms and the condition number*, in Proceedings of the 8th International Meshing Roundtable, Lake Tahoe, CA, 1999, Sandia Report SAND 99-2288, Sandia National Laboratories, Albuquerque, NM, 1999, pp. 13–22.

[15] P. Knupp, *Achieving finite element mesh quality via optimization of the Jacobian matrix norm and associated quantities. Part* I: *A framework for surface mesh optimization*, Internat. J. Numer. Methods Engrg., 48 (2000), pp. 401–420.

[16] P. Knupp, *Achieving finite element mesh quality via optimization of the Jacobian matrix norm and associated quantities. Part* II: *A framework for volume mesh optimization*, Internat. J. Numer. Methods Engrg., 48 (2000), pp. 1165–1185.

[17] G. Liao, *Variational approach to grid generation*, Numer. Partial Differential Equations, 8 (1992), pp. 143–147.

[18] A. Liu and B. Joe, *Relationship between tetrahedron quality measures*, BIT, 34 (1994), pp. 268–287.

[19] A. Liu and B. Joe, *On the shape of tetrahedra from bisection*, Math. Comp., 63 (1994), pp. 141–154.

[20] A. Oddy, J. Goldak, M. McDill, and M. Bibby, *A distortion metric for isoparametric elements*, Trans. Canad. Soc. Mech. Engrg., 12 (1988), pp. 213–217.

[21] V. N. Parthasarathy, C. M. Graichen, and A. F. Hathaway, *A comparison of tetrahedron quality measures*, Finite Elem. Anal. Des., 15 (1993), pp. 255–261.

[22] J. Robinson and G. Haggenmacher, *Element warning diagnostics*, Finite Element News, June-August, 1982.

[23] J. Robinson, *Some new distortion measures for quadrilaterals*, Finite Element Anal., 3 (1987), pp. 183–197.

[24] J. Robinson, *CRE method of element testing and the Jacobian shape parameters*, Engrg. Comput., 4 (1987), pp. 113–118.

SIAM J. SCI. COMPUT.
Vol. 23, No. 1, pp. 219–229

SAMPLING AND EIGENVALUES OF NON-SELF-ADJOINT STURM–LIOUVILLE PROBLEMS[*]

AMIN BOUMENIR[†]

Abstract. In this work we extend the sampling method to the case of non-self-adjoint Sturm–Liouville problems generated by complex-valued potentials and boundary conditions. It is shown that the function which is recovered by sampling on the real line extends analytically into the complex plane. A direct connection with polynomial approximation is established.

Key words. eigenvalues, non-self-adjoint Sturm–Liouville, interpolation, sampling theory

AMS subject classifications. 34L15, 42A15

PII. S1064827500374078

1. Introduction.

We would like to extend the sampling method to the computation of eigenvalues of regular non-self-adjoint Sturm–Liouville problems defined by

$$\begin{cases} -y''(x,\mu)+q(x)y(x,\mu)=\mu^2 y(x,\mu), & 0\le x\le \pi,\\ a_1 y(0,\mu)+a_2 y'(0,\mu)=0, & \\ b_1 y(\pi,\mu)+b_2 y'(\pi,\mu)=0, & \end{cases}$$

where $q\in L^1(0,\pi),\ a_i,b_i,\ q\in\mathbb{C},\ |a_1|+|a_2|\neq 0$, and $|b_1|+|b_2|\neq 0$.

It is well known that eigenvalues, if they exist, are in general nonreal and scattered in the complex plane, which makes the task of computing them very arduous; see [3]. Classical methods relying on tracking the number of zeros of the eigenfunctions, such as the Prufer method, cannot deal with a complex potential. Thus new methods evaluating the miss-distance function by Cauchy-type integrals were developed for computing eigenvalues in the complex plane; see [4].

We recall that the sampling method (see [2]) is basically an interpolation of the characteristic function whose zeros are the eigenvalues. We show that although in the non-self-adjoint case this function is complex valued, it is still in a Paley–Wiener space; see [1] and [7]. Thus we need to interpolate a complex-valued function defined on the complex plane. This immediately raises a new question: Should we need a two-dimensional sampling strategy, or should we sample on the real line only and then use an analytic extension to reach those roots situated far from the real line? Surprisingly, we shall see that the latter alternative is sufficient. The fact that only a few values are needed to interpolate the characteristic function with computable error bounds (see [2]) makes the sampling method very attractive. The actual code or algorithm is very simple and consists of only a few lines. A discussion on the computational aspects of the spectral theory of Sturm–Liouville problems and a comparison between existing codes can be found in [5] and [6].

[*] Received by the editors June 20, 2000; accepted for publication (in revised form) March 2, 2001; published electronically June 19, 2001.
http://www.siam.org/journals/sisc/23-1/37407.html

[†] Department of Mathematics, State University of West Georgia, Carrollton, GA 30118 (boumenir@westga.edu).

2. The characteristic function. We recall that the sampling method is basically a shooting data processing. If $y(x,\mu)$ denotes the solution of the initial value problem defined by

$$\begin{cases} -y''(x,\mu)+q(x)y(x,\mu)=\mu^2 y(x,\mu), \\ y(0,\mu)=a_2 \quad \text{and} \quad y'(0,\mu)=-a_1, \end{cases} \qquad 0\le x\le \pi,$$

then the characteristic function is simply defined by the second boundary condition

$$\Delta(\mu):=b_1 y(\pi,\mu)+b_2 y'(\pi,\mu), \tag{2.1}$$

and thus all zeros of Δ are the eigenvalues. We now study the analytic decomposition of Δ. The variation of parameters yields

$$y(x,\mu)=a_2\cos(x\mu)-a_1\frac{\sin(x\mu)}{\mu}+\int_0^x \frac{\sin((x-t)\,\mu)}{\mu}q(t)y(t,\mu)dt, \tag{2.2}$$

which implies that

$$y'(x,\mu)=-a_2\mu\sin(x\mu)-a_1\cos(x\mu)+\int_0^x \cos((x-t)\,\mu)q(t)y(t,\mu)dt.$$

Thus we deduce from (2.1) that

$$\begin{aligned}\Delta(\mu)&=b_1\left[a_2\cos(\pi\mu)-a_1\frac{\sin(\pi\mu)}{\mu}\right]+b_2\left[-a_2\mu\sin(\pi\mu)-a_1\cos(\pi\mu)\right]\\ &\quad+b_1\int_0^\pi \frac{\sin((\pi-t)\,\mu)}{\mu}q(t)y(t,\mu)dt+b_2\int_0^\pi \cos((\pi-t)\,\mu)q(t)y(t,\mu)dt. \end{aligned}\tag{2.3}$$

In order to find the roots of Δ, we need to approximate the last two integrals. To this end we use standard iterative methods to approximate the solution of the Volterra equation (2.2) as

$$y(x,\mu):=\sum_{n=0}^{\infty} y_n(x,\mu),$$

where

$$y_0(x,\mu)=a_2\cos(x\mu)-a_1\frac{\sin(x\mu)}{\mu}$$

and

$$y_n(x,\mu)=\int_0^x \frac{\sin((x-t)\,\mu)}{\mu}q(t)y_{n-1}(t,\mu)dt.$$

With the help of

$$\frac{|\sin(z)|}{|z|}\le\frac{c}{1+|z|}\exp\left(|\operatorname{Im} z|\right) \quad \text{and} \quad |\cos(z)|\le\exp\left(|\operatorname{Im} z|\right) \tag{2.4}$$

we successively obtain the following estimates:

$$|y_0(x,\mu)| \leq \left(|a_2| + |a_1| \frac{cx}{1+x|\mu|}\right) \exp(x|\operatorname{Im}\mu|),$$

$$\begin{aligned}|y_1(x,\mu)| &\leq \int_0^x \left|\frac{\sin((x-t)\mu)}{(x-t)\mu}\right| (x-t)|q(t)|\,|y_0(t,\mu)|\,dt \\ &\leq \left(|a_2| + |a_1| \frac{cx}{1+x|\mu|}\right) \int_0^x \frac{c(x-t)|q(t)|}{1+(x-t)|\mu|} dt \exp(x|\operatorname{Im}\mu|) \\ &\leq \left(|a_2| + |a_1| \frac{cx}{1+x|\mu|}\right) \frac{cx}{1+x|\mu|} \int_0^x |q(t)|\,dt \exp(x|\operatorname{Im}\mu|),\end{aligned}$$

which leads to

$$\begin{aligned}|y_2(x,\mu)| &\leq \frac{cx}{1+x|\mu|} \int_0^x |q(t)|\,|y_1(t,\mu)|\,dt \exp(x|\operatorname{Im}\mu|) \\ &\leq \left(|a_2| + |a_1| \frac{cx}{1+x|\mu|}\right) \frac{cx}{1+x|\mu|} \int_0^x \frac{ct}{1+t|\mu|} |q(t)| \int_0^t |q(\eta)|\,d\eta dt \exp(x|\operatorname{Im}\mu|) \\ &\leq \left(|a_2| + |a_1| \frac{cx}{1+x|\mu|}\right) \left(\frac{cx}{1+x|\mu|}\right)^2 \int_0^x |q(t)| \int_0^t |q(\eta)|\,d\eta dt \exp(x|\operatorname{Im}\mu|) \\ &\leq \left(|a_2| + |a_1| \frac{cx}{1+x|\mu|}\right) \left(\frac{cx}{1+x|\mu|}\right)^2 \left(\frac{1}{2}\int_0^x |q(t)|\,dt\right)^2 \exp(x|\operatorname{Im}\mu|).\end{aligned}$$

The successive integrations yield

$$\begin{aligned}|y_n(x,\mu)| &\leq \frac{cx}{1+x|\mu|} \int_0^x |q(t)|\,|y_{n-1}(t,\mu)| \exp((x-t)|\operatorname{Im}\mu|)\,dt \\ &\leq \left(|a_2| + \frac{|a_1|\,cx}{1+x|\mu|}\right) \frac{1}{n!} \left(\frac{cx}{1+x|\mu|} \int_0^x |q(t)|\,dt\right)^n \exp(x|\operatorname{Im}\mu|).\end{aligned}$$

Thus we recover the well-known exponential growth in μ,

$$|y(x,\mu)| \leq \left(|a_2| + \frac{|a_1|\,cx}{1+x|\mu|}\right) \exp\left(cx\int_0^x |q(t)|\,dt\right) \exp(x|\operatorname{Im}\mu|), \tag{2.5}$$

and the integrals appearing in (2.3) can be estimated:

$$\left|\int_0^\pi \frac{\sin((\pi-t)\mu)}{\mu} q(t)y(t,\mu)dt\right| \leq \frac{c\pi M(q)}{1+\pi|\mu|} \exp(\pi|\operatorname{Im}\mu|) \tag{2.6}$$

and similarly

$$\left|\int_0^\pi \cos((\pi-t)\mu) q(t)y(t,\mu)dt\right| \leq M(q) \exp(\pi|\operatorname{Im}\mu|), \tag{2.7}$$

where

$$M(q) := \left(|a_2| + \frac{|a_1|\,c\pi}{1+\pi|\mu|}\right) \int_0^\pi |q(t)| \exp\left(ct\int_0^t |q(\eta)|\,d\eta\right) dt.$$

Observe that the upper bound in (2.7) is not square integrable on the real line if $a_2 \neq 0$. In order to have a faster decay in μ, we use the difference $y(x,\mu) - y_0(x,\mu)$ instead of $y(x,\mu)$,

$$|y(x,\mu) - y_0(x,\mu)| \leq \frac{c\pi M(q)}{1+\pi|\mu|} \exp\left(x\left|\operatorname{Im}\mu\right|\right), \tag{2.8}$$

and so we can recast (2.7) as

$$\left|\int_0^\pi \cos((\pi-t)\,\mu)q(t)\left(y(t,\mu)-y_0(t,\mu)\right)dt\right| \leq \frac{c\pi M(q)}{1+\pi|\mu|}\int_0^\pi |q(t)|\,dt \exp\left(\pi\left|\operatorname{Im}\mu\right|\right). \tag{2.9}$$

This leads to a first practical decomposition of Δ,

$$\begin{aligned}
\Delta(\mu) &= b_1\left[a_2\cos(\pi\mu) - a_1\frac{\sin(\pi\mu)}{\mu}\right] - b_2\left[a_2\mu\sin(\pi\mu) + a_1\cos(\pi\mu)\right] \\
&\quad + a_2b_2\int_0^\pi \cos((\pi-t)\,\mu)\cos(t\mu)q(t)dt - a_1b_2\int_0^\pi \cos((\pi-t)\,\mu)\frac{\sin(t\mu)}{\mu}q(t)dt \\
&\quad + b_1\int_0^\pi \frac{\sin((\pi-t)\,\mu)}{\mu}q(t)y(t,\mu)dt + b_2\int_0^\pi \cos((\pi-t)\,\mu)q(t)\left(y(t,\mu)-y_0(t,\mu)\right)dt \\
&= G(\mu) + S(\mu),
\end{aligned}$$

where after a few algebraic simplifications

$$\begin{aligned}
G(\mu) &= \cos(\mu\pi)\left[a_2b_1 + b_2\left(-a_1 + a_2\int_0^\pi \cos^2(t\mu)\,q(t)dt - \frac{a_1}{2}\int_0^\pi \frac{\sin(2t\mu)}{\mu}q(t)dt\right)\right] \\
&\quad + \frac{\sin(\pi\mu)}{\mu}\left[-a_1b_1 + b_2\left(-a_2\mu^2 + \frac{1}{2}a_2\mu\int_0^\pi \sin(2t\mu)\,q(t)dt - a_1\int_0^\pi \sin^2(t\mu)\ \ q(t)dt\right)\right]
\end{aligned} \tag{2.10}$$

and

$$S(\mu) := b_1\int_0^\pi \frac{\sin((\pi-t)\,\mu)}{\mu}q(t)y(t,\mu)dt + b_2\int_0^\pi \cos((\pi-t)\,\mu)q(t)\left(y(t,\mu)-y_0(t,\mu)\right)dt.$$

Thus using (2.6) and (2.9) we obtain the following.

Proposition 1. *Assume that* $q \in L^1(0,\pi)$, $a_i, b_i,\ q \in \mathbb{C}$, $|a_1| + |a_2| \neq 0$, *and* $|b_1| + |b_2| \neq 0$; *then* $S(\mu) \in PW_\pi$.

The above proposition allows us to recover the function S by the Whittaker–Shannon–Kotelnikov (W.S.K.) theorem (see [7]):

$$S(\mu) = \sum_{n\in Z} S(n)\frac{\sin(\pi(\mu-n))}{\pi(\mu-n)}, \tag{2.11}$$

where the sampling values $S(n)$,

$$S(n) := \Delta(n) - G(n),$$

are computed by evaluating (2.1) and (2.10) at the integers only.

We now analyze the truncation error due to replacing S by a finite sum:

$$S_N(\mu) := \sum_{n=-N}^{N} S(n)\frac{\sin(\pi(\mu-n))}{\pi(\mu-n)}. \tag{2.12}$$

$$\begin{aligned}\Delta(\mu) &= G(\mu) + S(\mu)\\ &= G(\mu) + \sum_{n=-N}^{N} S(n)\frac{\sin(\pi(\mu-n))}{\pi(\mu-n)} + \sum_{|n|>N} S(n)\frac{\sin(\pi(\mu-n))}{\pi(\mu-n)}\\ &= G(\mu) + \sin(\pi\mu)\sum_{n=-N}^{N} \frac{(-1)^n S(n)}{\pi(\mu-n)} + \sin(\pi\mu)\sum_{|n|>N} \frac{(-1)^n S(n)}{\pi(\mu-n)}\\ &= G(\mu) + \sin(\pi\mu)\sum_{n=-N}^{N} \frac{(-1)^n S(n)}{\pi(\mu-n)} + \sin(\pi\mu)\sum_{|n|>N} \frac{(-1)^n S(n)}{\pi(\mu-n)}.\end{aligned}$$

Observe that $y(\pi,\mu)$ is an even function of μ, and thus $S(-n) = S(n)$, which further simplifies the above sums:

$$\begin{aligned}\Delta(\mu) &= G(\mu) + \mu\sin(\pi\mu)\frac{2}{\pi}\left(\frac{S(0)}{2\mu^2} + \sum_{n=1}^{N}\frac{(-1)^n S(n)}{(\mu^2-n^2)}\right) + \mu\sin(\pi\mu)\frac{2}{\pi}\sum_{n>N}\frac{(-1)^n S(n)}{(\mu^2-n^2)}\\ &= G(\mu) + \frac{\sin(\pi\mu)}{Q_N(\mu)}P_N(\mu) + T_N(\mu),\end{aligned}$$

where P_N and

$$Q_N(\mu) = \pi\mu\prod_{n=1}^{N}\left(1-\frac{\mu^2}{n^2}\right)$$

are polynomials of degree $2N$ and $2N+1$, respectively, and observe that the only zeros of the entire function $\frac{\sin(\pi\mu)}{Q_N(\mu)}$ are $\{n \in Z : |n| \geq N+1\}$, and furthermore

$$\frac{\sin(\pi\mu)}{Q_N(\mu)} \to 1 \text{ as } N \to \infty.$$

It is readily seen that if K is a compact set of the complex plane not containing large integers, i.e., $\{n \in Z : |n| > N\} \cap K = \emptyset$, then the truncation error T_N is bounded by

$$\sup_{\mu\in K}|T_N(\mu)| \leq \frac{2}{\pi}\sqrt{\sum_{n>N}|S(n)|^2}\,\sup_{\mu\in K}\left|\mu\sin(\pi\mu)\sqrt{\sum_{n>N}\frac{1}{(\mu^2-n^2)^2}}\right|.$$

Since $|S(\mu)| = o(\mu^{-1})$, the convergence of $\sum_{n\geq 0}|S(n)|^2$ implies that $\sup_{\mu\in K}|T_N(\mu)| \to 0$ as $N \to \infty$, and thus we can use the approximation provided by

$$\Delta_N(\mu) := G(\mu) + \frac{2}{\pi}\frac{\sin(\pi\mu)}{Q_N(\mu)}P_N(\mu) = 0 \tag{2.13}$$

to compute the roots of $\Delta(\mu)$ in the complex plane. Uniform convergence implies that $\Delta_N(\mu) \to 0$ as $N \to \infty$, and so the roots of Δ_N approximate the roots of $\Delta(\mu)$.

PROPOSITION 2. *Let $q \in L^1(0,\pi)$ and K be a compact set in the complex plane; then* $\sup_{\mu\in K}|\Delta_N(\mu)-\Delta(\mu)| \to 0$ *as* $N\to\infty$.

It is true that the integrals in the function G (2.10) may not be easy to compute. Can we avoid computing the Fourier transforms appearing in G? Observe that if $q\in L^2(0,\pi)$, then $\sin\mu\pi\int_0^\pi \sin(2t\mu)\,q(t)dt$ is square integrable for $\mu\in R$ since it is the Fourier transform of q. It is also an entire function, and its exponential growth is given by (2.4)

$$\left|\sin\mu\pi\int_0^\pi \sin(2t\mu)\,q(t)dt\right| = O\left(\exp\left(3\pi\left|\operatorname{Im}(\mu)\right|\right)\right),$$

and thus $\sin\mu\pi\int_0^\pi \sin(2t\mu)\,q(t)dt \in PW_{3\pi}$, and similarly we obtain

$$\cos\mu\pi\int_0^\pi \cos t\mu\frac{\sin(t\mu)}{\mu}q(t)dt,\ \sin\mu\pi\int_0^\pi \frac{\sin^2 t\mu}{\mu}q(t)dt \in PW_{3\pi}.$$

The last remaining integral needs a different treatment, namely,

$$\int_0^\pi \cos^2(t\mu)\,q(t)dt = \frac{1}{2}\int_0^\pi \cos(2t\mu)\,q(t)dt + \frac{1}{2}\int_0^\pi q(t)dt,$$

and we obviously have $\cos\mu\pi\int_0^\pi\cos(2t\mu)\,q(t)dt = O\left(\exp\left(3\pi\left|\operatorname{Im}(\mu)\right|\right)\right)$ and

$$\cos\mu\pi\int_0^\pi \cos(2t\mu)\,q(t)dt \in L^2(-\infty,\infty).$$

Thus we deduce again that

$$\cos\mu\pi\int_0^\pi \cos(2t\mu)\,q(t)dt \in PW_{3\pi}.$$

The remaining term $\frac{1}{2}\int_0^\pi q(t)dt\cos\mu\pi$ is easy to compute and can be included in G. Hence a simpler decomposition follows if we assume that $q\in L^2(0,\pi)$ instead of $q\in L^1(0,\pi)$.

PROPOSITION 3. *Assume that* $q\in L^2(0,\pi)$, $a_i, b_i, q\in\mathbb{C}$, $|a_1|+|a_2|\neq 0$, *and* $|b_1|+|b_2|\neq 0$; *then*

$$\Delta(\mu) := G_1(\mu)+S_1(\mu),$$

where

$$G_1(\mu) = \left(b_1a_2 - a_1b_2 + a_2b_2\frac{1}{2}\int_0^\pi q(t)dt\right)\cos(\pi\mu) - a_2b_2\mu\sin(\pi\mu)$$

and $S_1\in PW_{3\pi}$.

Here the cost of the simplification is that we need three times more sampling points since the step size is $\frac{1}{3}$ instead of 1. There is a further interesting simplification that will be crucial for computing purposes. In the Dirichlet case, e.g., a_2 and b_2 are nil; then Δ simplifies since $G_1(\mu)=0$.

We now summarize the main points of the above results. In the case where we are working with Proposition 1 then we use the space PW_π, step size = 1, and an approximation of Δ is given by

$$\Delta_N(\mu) = \cos(\pi\mu)F(\mu)+\sin(\pi\mu)H(\mu)+P_N(\mu)\sin(\pi\mu)/Q_N(\mu),$$

where F, H are Fourier transforms, P_N is a polynomial, $\sin(\pi\mu)/Q_N(\mu)$ is an entire function whose zeros are $\{n \in Z : |n| \geq N+1\}$ and converges to 1. If the functions F and H are too difficult to evaluate, then one could use Proposition 2 where the approximation takes place in $PW_{3\pi}$

$$\Delta_N(\mu) := \alpha\cos(\pi\mu) + \beta\mu\sin(\pi\mu) + P_N(\mu)\sin(3\pi\mu)/Q_N(\mu) = 0,$$

where α and β are constants and $\sin(3\pi\mu)/Q_N(\mu)$ is an entire function which converges to 1.

3. Examples. We examine the simplest case corresponding to the Dirichlet case. Indeed if

$$a_1 = b_1 = 1 \quad \text{and} \quad a_2 = b_2 = 0,$$

then the characteristic function given by (2.10) and Proposition 1 reduces to

$$\Delta(\mu) = -\frac{\sin(\pi\mu)}{\mu} + S(\mu)$$

and the approximation of S takes place in PW_π. The approximation is then given by (2.13):

$$\begin{aligned}\Delta_N(\mu) &= -\frac{\sin(\pi\mu)}{\mu} + \frac{\sin(\pi\mu)}{\pi\mu}S(0) + \frac{2}{\pi}\sum_{n=1}^{N}\frac{(-1)^n\,\mu S(n)}{\left(\frac{\mu^2}{n^2}-1\right)n^2}\\ &= \frac{\sin(\pi\mu)}{Q_N(\mu)}P_N(\mu),\end{aligned}$$

where $Q_N(\mu) = \pi\mu\prod_{n=1}^{N}(1-\frac{\mu^2}{n^2})$. Recall that by Weiestrass factorization theorem $\sin(\pi\mu) = \pi\mu\prod_{n=1}^{\infty}(1-\frac{\mu^2}{n^2})$ and so

$$\frac{\sin(\pi\mu)}{Q_N(\mu)} = \prod_{n=N+1}^{\infty}\left(1-\frac{\mu^2}{n^2}\right)$$

is an entire function whose zeros are on the real line. As $N \to \infty$, $\frac{\sin(\pi\mu)}{Q_N(\mu)} \to 1$, and hence the roots of Δ are given by the zeros of P_∞, which is easily approximated by P_N.

Example 1. We first use a simple example where the eigenvalues are known explicitly:

$$\begin{cases} -y''(x,\mu) + (3-2*I)y(x,\mu) = \mu^2 y(x,\mu),\\ y(0,\mu) = y(\pi,\mu) = 0.\end{cases}$$

In this case the eigenvalues are $\mu_n^2 = n^2 + 3 - 2*I$ for $n = 1, 2, \ldots$. With $y(0,\mu) = 0$, $y'(0,\mu) = 1$ as the initial condition, the differentials equation is integrated numerically with a precision set to 18 digits to obtain the sampling values $y(\pi,k)$ for $k = 0, 1, 2, \ldots, N$. We take $N = 40$ points and the first values are

$$\begin{aligned} y(\pi,0) &= 10.7815379571786450 - 78.7057658098574994I,\\ y(\pi,1) &= -2.28608041879586698 - 39.1205731809301484I.\end{aligned}$$

The eigenvalues recovered by sampling are
4.0000311171782160105199196630 8 – 1.9999563057552963981972457278 1 I,
7.0002041538324838527701078139 0 – 2.0000556601345079145822561257 4 I,
12.000390342967500517056588989 1 – 2.0003451987009086998878652117 5 I,
19.000597046429035307830509552 2 – 2.0007706599021278195036136178 0 I,
28.000842352581089759639060374 4 – 2.0013225756357561780364942035 2 I,
39.001134662493089360324769575 3 – 2.0020018764908230296513558823 9 I,
52.001478344765443878125034253 2 – 2.0028116961809404980215556865 1 I,
67.001876365826759664074200520 7 – 2.0037560982796744907881186829 4 I,
84.002331278299643129331987134 5 – 2.0048399214279273555057382519 2 I,
103.00284562728609774049050949 6 – 2.0060688237627928651288931507 8 I,
124.00342214572743137417555280 2 – 2.0074493746317404999523116749 8 I,
147.00406386957507468770564345 0 – 2.0089891690201940341999819138 6 I,
172.00477422314591148744531578 7 – 2.0106969640173962412499716114 5 I,
199.00555709649031113021078736 4 – 2.0125828420346421396131082195 4 I,
228.00641692584410660945969643 0 – 2.0146584075809120628016870572 0 I,
259.00735878415616346227673150 6 – 2.0169370262841335181000480092 5 I,
292.00838848739345037557091778 4 – 2.0194341172163921495885526852 4 I,
327.00951272248183777699520525 6 – 2.0221675133047959908408128637 2 I,
364.01073920355017051389062823 0 – 2.0251579072680384492887521209 5 I,
403.01207686575065811647983738 4 – 2.0284294165562888167715848494 6 I,
444.01353610576668851364461668 3 – 2.0320102704138144935470438477 5 I,
487.01512909237949520393322068 7 – 2.0359337795858903927297957248 9 I,
532.01687014274427786690076036 9 – 2.0402393087327391677867929374 8 I,
579.01877627095912566714035769 3 – 2.0449742848185298970895469849 1 I,
628.02086774340163330529097095 4 – 2.0501951691457745516059570722 2 I,
679.02316921652842989962510527 7 – 2.0559721627780062341349776246 7 I,
732.02571041421037280685514618 1 – 2.0623897267841536007868094988 0 I,
787.02852875009615882110251438 5 – 2.0695563486329058198192558206 7 I,
844.03167008451948717133588500 1 – 2.0776066635715632519701711464 4 I,
903.03519442728300193123156125 8 – 2.0867179463458670076492914936 4 I,
964.03917801583662977249122223 6 – 2.0971226789360428188083331800 5 I,
1027.0437245202254184242882594 1 – 2.1091376304665728017530905623 5 I,
1092.0489742371666033471636546 1 – 2.1232075528428609796258626612 1 I,
1159.0551283224759296669904414 1 – 2.1399809425363863111049376092 3 I,
1228.0624849190055402700593030 4 – 2.1604559471335069627574106124 8 I,
1299.0715154524844623686775442 3 – 2.1862736842674060080899925653 8 I,
1372.0830262152627467942839815 1 – 2.2204029731717988537688109878 1 I,
1447.0985667408315500147689169 5 – 2.2690302318731190889933998464 3 I,
1524.1216991431764869342294399 0 – 2.3485304385436856926755202546 3 I,
1603.1630623747688137832609560 6 – 2.5315296352398540224535584060 8 I.

Observe that the precision on the last 3 eigenvalues deteriorates. Although we evaluated Δ on the small set $\mu \in \{0, 1, \ldots, 40\}$, we have recovered all the eigenvalues up to $1603 - 2I$; see Figure 1.

Example 2. Consider by the following example:

$$\begin{cases} -y''(x,\mu) + (1+I)x^2 y(x,\mu) = \mu^2 y(x,\mu), \\ y(0,\mu) = y(\pi,\mu) = 0. \end{cases}$$

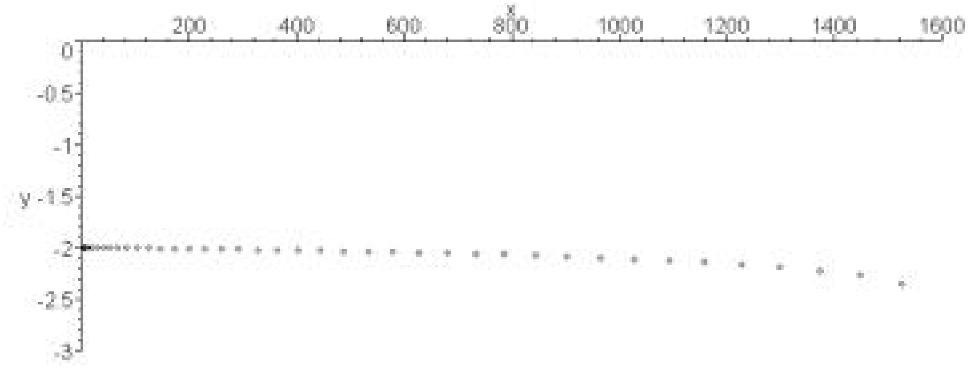


FIG. 1. *The eigenvalues when $q(x) = 3 - 2I$.*

The first values of $y(\pi, n)$ are computed with the same precision as previously:

$$y(\pi, 0) = -30.0194706498185406 + 76.7008318167070961I,$$
$$y(\pi, 1) = -22.3863633586727145 + 31.8682226573447697I,$$
$$y(\pi, 2) = -2.64524868593933849 - 4.08802765222044812I.$$

The polynomial P_{40} is easily factored and the eigenvalues are given by μ_n^2:

```
3.29253037825943601596867687809+1.36632084628037852259220558655*I,
7.55934410143874839356684029135+3.05050755362608645207883714464*I,
12.3408421150254241854985713087+3.59194317093833780714293144087*I,
19.2651371680320693733196765612+3.50329090722762858791667705990*I,
28.2688247053416851502952013831+3.43560053370828196477751781983*I,
39.2750946834316762649570963510+3.39572923480482837241129814250*I,
52.2793839856433889856205908903+3.37121167635693801412242664588*I,
67.2822026002188346591998175950+3.35561341445895745136095390597*I,
84.2841102728951751078269094230+3.34552449836996791448784753725*I,
103.285450312128415699438155443+3.33904366406027612120531704486*I,
124.286423757722412736910588494+3.33505837193087094508063602670*I,
147.287150889041087373630006356+3.33288996603419795032708046613*I,
172.287706118871959236503471586+3.33211050348722059823590983364*I,
199.288136988681506832586572371+3.33244371842929364250935479801*I,
228.288474685593763430633784664+3.33370916386452896480688558308*I,
259.288740043679755686191437388+3.33578954127871107401292199048*I,
292.288947074511241288307635854+3.33861103233429450553327338156*I,
327.289105096996969683611664530+3.34213123137761314807034400174*I,
364.289220040454918278616868688+3.34633171420510740002752388090*I,
403.289295240261827964727565710+3.35121357171289675695360109194*I,
444.289331888580153121356639146+3.35679496567974679685691770136*I,
487.289329293256090054635381932+3.36311014438980555314836391379*I,
532.289284808827018900856035124+3.37020975220170691041324735695*I,
579.289193989785253647074016795+3.37816211061365866944365896078*I,
628.289049605915489613261968037+3.38705606335841537336424927432*I,
679.288842465198419391422191127+3.39700479351993937533821580180*I,
732.288558118692501189449830682+3.40815216955141510314990510260*I,
```

787.288179391664247960717905984+3.42068158514385139540949304562*I,
844.287679123652095024874502221+3.43482821186760605745143092907*I,
903.287023268197191205711806008+3.45089984621597911838037436493*I,
964.286159451355571564987371247+3.46930340656725306330552137688*I,
1027.28501392766321993385486814+3.49059496673576521878753578240*I,
1092.28347283002434782727284039+3.51555190297238020372008460264*I,
1159.28135621021468994770225772+3.54530966130896483842231064423*I,
1228.27836136536422690944574066+3.58160594382394663019250161305*I,
1299.27393766588942806766218238+3.62728628033340094539060165642*I,
1372.26696926830263340704927922+3.68745740327327333471564892036*I,
1447.25478880843818305788161984+3.77263663648478767674925779622*I,
1524.22894351618947865743660277+3.91007105114086717378638998354*I,
1603.13915280328851611418157527+4.21360415942128687293484359992*I.

The characteristic function can be reduced to a hypergeometric series $\Delta(\mu) = c\mathrm{WhittakerM}(\frac{\mu^2}{4\sqrt{1+I}},\ \frac{1}{4},\ \pi^2\sqrt{1+I})$. Since its zeros are not known explicitly we can, for example, check the validity of the numerical results by evaluating $\Delta\left(\mu_n\right)$ numerically; for example,

μ^2	$\Delta(\mu)$
$3.292530378 + 1.366320846280 * I$	$-0.000129911013565507 - 0.0000337654876381 * I$
$103.2854503 + 3.339043664060 * I$	$-0.000080009701487999 + 0.0003554960921802 * I$
$679.28884246 + 3.39700479352 * I$	$-0.000140099029306098 + 0.0004554926139912 * I.$

This shows that N sampling values on the real line are enough to approximate N eigenvalues in the complex plane; see Figure 2. It is also clear that the precision improves as we take more sampling points.

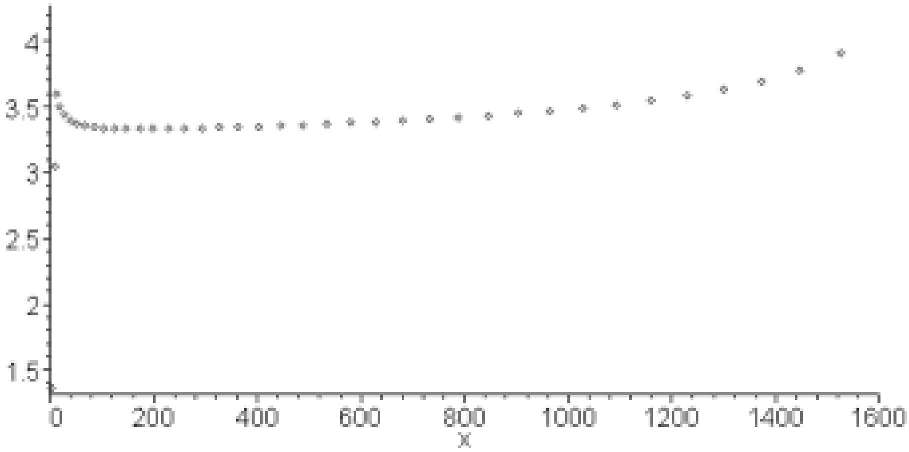


FIG. 2. *Eigenvalues when $q(x) = (1 + I)x^2$.*

REFERENCES

[1] N. I. ACHIESER, *Theory of Approximation*, Dover, New York, 1992.

[2] A. BOUMENIR, *Computing eigenvalues of a periodic Sturm Liouville problem by the Shannon Whittaker sampling theorem*, Math. Comp., 68 (1999), pp. 1057–1066.

[3] E. B. Davies, *Pseudo-spectra, the harmonic oscillator and complex resonances*, Proc. Roy. Soc. London Ser. A, 455 (1999), pp. 585–599.

[4] L. Greenberg and M. Marletta, *Numerical solution of non–self-adjoint Sturm–Liouville problems and related systems*, SIAM J. Numer. Anal., 38 (2001), pp. 1800–1845.

[5] D. Hinton and P. Schaefer, *Spectral Theory and Computation Methods of Sturm-Liouville Problems*, Lectures Notes in Pure and Appl. Math. 191, Dekker, New York, 1997.

[6] J. D. Pryce, *Numerical Solution of Sturm-Liouville Problems*, Monographs on Numerical Analysis, Oxford University Press, London, 1993.

[7] A. I. Zayed, *Advances in Shannon's Sampling Theory*, CRC Press, Boca Raton, FL, 1993.

SIAM J. Sci. Comput.
Vol. 23, No. 1, pp. 230–244

SOME IMPROVEMENTS OF THE FAST MARCHING METHOD*

DAVID L. CHOPP†

Abstract. The fast marching method published by Sethian [*Proc. Natl. Acad. Sci. USA*, 93 (1996), pp. 1591–1595] is an optimally efficient algorithm for solving problems of front evolution where the front speed is monotonic. It has been used in a wide variety of applications such as robotic path planning [R. Kimmel and J. Sethian, *Fast Marching Methods for Computing Distance Maps and Shortest Paths*, Tech. Report 669, CPAM, University of California, Berkeley, 1996], crack propagation [M. Stolarska et al., *Internat. J. Numer. Methods Engrg.*, 51 (2001), pp. 943–960; N. Sukumar, D. L. Chopp, and B. Moran, *Extended finite element method and fast marching method for three-dimensional fatigue crack propagation*, J. Comput. Phys., submitted], seismology [J. Sethian and A. Popovici, *Geophysics*, 64 (1999), pp. 516–523], photolithography [J. Sethian, *Fast marching level set methods for three-dimensional photolithography development*, in Proceedings of the SPIE 1996 International Symposium on Microlithography, Santa Clara, CA, 1996], and medical imaging [R. Malladi and J. Sethian, *Proc. Natl. Acad. Sci. USA*, 93 (1996), pp. 9389–9392]. It has also been a valuable tool for the implementation of modern level set methods where it is used to efficiently compute the distance to the front and/or an extended velocity function.

In this paper, we improve upon the second order fast marching method of Sethian [*SIAM Rev.*, 41 (1999), pp. 199–235] by constructing a second order approximation of the interface generated from local data on the mesh. The data is interpolated on a single box of the mesh using a bicubic approximation. The distance to the front is then calculated by using a variant of Newton's method to solve both the level curve equation and the orthogonality condition for the nearest point to a given node. The result is a second order approximation of the distance to the interface which can then be used to produce second order accurate initial conditions for the fast marching method and a third order fast marching method.

Key words. fast marching method, level set method, bicubic interpolation, reinitialization

AMS subject classifications. 65M06, 65D05, 65D10

PII. S106482750037617X

1. Introduction.

The fast marching method published by Sethian [12] is an optimally efficient algorithm for solving general problems of front evolution where the front speed is monotonic. It has been used in a wide variety of applications such as robotic path planning [7], crack propagation [15, 16], seismology [14], photolithography [10], and medical imaging [8]. It has also been a valuable tool for the implementation of modern level set methods where it is used to efficiently compute the distance to the front and/or an extended velocity function. The fast marching method can trace its roots back to graph theoretical results first published by Dijkstra [4], and a review of the fast marching method can be found in Sethian's review article [13].

The fast marching method, presented in [12], was first order accurate, using a first order approximation of the derivatives. In [13], a second order fast marching method is constructed by using corresponding higher order approximations of the derivatives. For one to achieve higher order methods, the initial data must be more accurate. This is possible if the initial distance to the front is given explicitly, but it is more complicated if the initial front is given as a level curve of a function on the mesh, as

[*] Received by the editors August 4, 2000; accepted for publication (in revised form) February 20, 2001; published electronically June 19, 2001. This work was supported by the NSF/DARPA VIP program under award 96-15877.

http://www.siam.org/journals/sisc/23-1/37617.html

[†] Engineering Sciences and Applied Mathematics Dept., Northwestern University, Evanston, IL 60208-3125 (chopp@northwestern.edu).

commonly arises in applications of the level set method. The method for initializing the fast marching data in [13] uses first order accurate linear interpolation limiting the global error to second order at best.

The level set method is a numerical method which has gained significant popularity in recent years for solving complex problems of evolving interfaces, particularly where the topology of the interface may change. The key advantages of the method are that it can change topology without special cases, can propagate sharp corners and cusps in the interface, and is easily extendible to any number of spatial dimensions. This compares with marker particle-type methods (see, e.g., [5]), where topology changes require collision detection and interface surgery, and the mesh itself requires node point redistribution for stability (and also smooths sharp edges formed by the flow). On the other hand, the interface is known only implicitly in a level set formulation, while it is known explicitly for a marker particle method. The results of this paper offer a way to close this latter gap between the two types of interface representations. Both methods have their uses, and a complete comparison between these two methods is beyond the scope of this paper. For a complete discussion of the level set method and reinitialization, see [9, 11].

One common application of the fast marching method is for implementing reinitialization within the framework of the level set method. Reinitialization, introduced in [3], is where the signed distance map to the zero level set of a function is computed. One common criticism of the level set method is its purported problem with mass conservation. Reinitialization techniques, such as those given in [3] and [18], are well known to have difficulty preserving mass. This is primarily due to the poor accuracy around the zero level set.

In this paper, we improve upon the second order fast marching method of Sethian [13] by constructing a second order approximation of the interface generated from local data on the mesh. The data is interpolated on a single box of the mesh using a bicubic approximation. The distance to the front is then calculated by using a variant of Newton's method to solve both the level curve equation and the orthogonality condition for the nearest point to a given node. The result is a second order approximation of the distance to the interface which can then be used to produce second order accurate initial conditions for the fast marching method and a third order fast marching method. We show with computed examples that the method appears to be second order accurate locally around the zero level set, can be used to build a globally third order accurate fast marching method, preserves mass markedly better during reinitialization than existing methods, and easily generalizes to any number of spatial dimensions—all without increasing the computational complexity of the fast marching method.

In section 2, we describe the fast marching method as presented in [12, 13]. In section 3 we describe the bicubic interpolation and how it is used to construct a second order approximation to the distance map. We also give the formulation for implementing a third order accurate fast marching method. In section 4 we compare the new bicubic approximation method with the second order method in [13] with a series of examples. Finally, in section 5 we summarize our results, discuss how this local approximation can be used for velocity extensions, and briefly describe a current application in crack propagation where this method is already successfully in use.

2. The fast marching method.

The fast marching method is an optimal method for solving an equation of the form

$$\|\nabla\phi\| = 1/F(\mathbf{x}), \tag{2.1}$$

where $F(\mathbf{x})$ is a monotonic speed function for an advancing interface. If $\phi^{-1}(0)$ represents the initial interface, and ϕ solves (2.1), then $\phi^{-1}(t)$ gives the location of the interface evolving with normal velocity $F(\mathbf{x}) > 0$ at time t. The advantage of this method over the original level set method is that the entire evolution of the front is computed in one pass over the mesh with an operation count of $O(N \log N)$ for N mesh points. It is also advantageous over other front tracking algorithms in that it uses techniques borrowed from hyperbolic conservation laws to properly advance fronts with sharp corners and cusps. We present here a basic description of the method; the interested reader is referred to [11, 12, 13].

In the fast marching method, (2.1) is solved numerically by using upwind finite differences to approximate $\nabla\phi$. The use of upwind finite differences indicates a causality, or a direction for the flow of information propagating from the initial contour $\phi^{-1}(0)$ outward to larger values of ϕ. This causality means that the value of $\phi(\mathbf{x})$ depends only on values of $\phi(\mathbf{y})$ for which $\phi(\mathbf{y}) \le \phi(\mathbf{x})$. Thus, if we solve for the values of ϕ in a monotonically increasing fashion, then the upwind differences are always valid and all the mesh points are eventually computed. This sequential procession through the mesh points is maintained by a heap sort which controls the order in which the mesh points are computed.

To begin, the mesh points are separated into three disjoint sets: the set of *accepted* points A, the set of *tentative* points T, and the set of *distant* points D. The mesh points in the set A are considered computed and are always closer to the initial interface than any of the remaining mesh points. The mesh points in T are all potential candidates to be the next mesh point to be added to the set A. The mesh points in T are always kept sorted in a heap sort so that the best candidate is always easily found. The mesh points in D are considered too far from the initial interface to be possible candidates for inclusion in A. Thus, if $\mathbf{x} \in A$, $\mathbf{y} \in T$, and $\mathbf{z} \in D$, then $\phi(\mathbf{x}) < \phi(\mathbf{y}) < \phi(\mathbf{z})$. Figure 2.1 shows the relationship between the different sets of mesh points.

To describe the main algorithm for the fast marching method, we will use the notation for discrete derivatives given by

$$\begin{aligned}
\phi_{i,j} &= \phi(\mathbf{x}_{i,j}),\\
D_x^+\phi_{i,j} &= \frac{1}{\Delta x}(\phi_{i+1,j} - \phi_{i,j}),\\
D_x^-\phi_{i,j} &= \frac{1}{\Delta x}(\phi_{i,j} - \phi_{i-1,j}),\\
D_y^+\phi_{i,j} &= \frac{1}{\Delta y}(\phi_{i,j+1} - \phi_{i,j}),\\
D_y^-\phi_{i,j} &= \frac{1}{\Delta y}(\phi_{i,j} - \phi_{i,j-1}),
\end{aligned}$$

where Δx, Δy are the space step sizes in the x and y directions, respectively. One of the key components in the fast marching method is the computation of the estimate of ϕ for points in T. Suppose, for example, mesh points $\mathbf{x}_{i-1,j}$, $\mathbf{x}_{i,j+1} \in A$ and $\mathbf{x}_{i,j} \in T$. Given the values of $\phi_{i-1,j}$, $\phi_{i,j+1}$, we must estimate the value of $\phi_{i,j}$. This is accomplished by looking at the discretization of (2.1) given by

$$(D_x^-\phi_{i,j})^2 + (D_y^+\phi_{i,j})^2 = \frac{1}{F_{i,j}^2}. \tag{2.2}$$

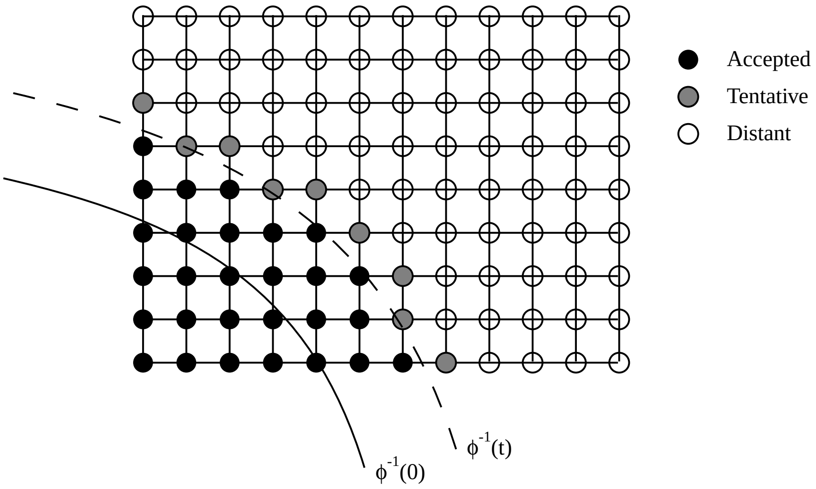


FIG. 2.1. *Illustration of the sets A, T, and D.*

Equation (2.2) reduces to a quadratic equation in $\phi_{i,j}$ given by

$$\left(\frac{1}{\Delta x^2}+\frac{1}{\Delta y^2}\right)\phi_{i,j}^2-2\left(\frac{\phi_{i-1,j}}{\Delta x^2}+\frac{\phi_{i,j+1}}{\Delta y^2}\right)\phi_{i,j}+\frac{\phi_{i-1,j}^2}{\Delta x^2}+\frac{\phi_{i,j+1}^2}{\Delta y^2}-\frac{1}{F^2}=0. \tag{2.3}$$

The new estimate for $\phi_{i,j}$ is given by the largest of the two roots of (2.3). The remaining configurations and the resulting quadratic equations can be derived in a similar fashion.

Note that (2.2) is a first order approximation of (2.1). The general second and third order approximations of (2.1) requires the use of switches determined by which points are in A:

$$\begin{aligned}&\max\Bigg(D_x^-\phi_{i,j}+s_{x,-1}\frac{\Delta x}{2}D_x^-D_x^-\phi_{i,j}+s_{x,-1}s_{x,-2}\frac{\Delta x^2}{6}D_x^-D_x^-D_x^-\phi_{i,j},\\&\qquad -D_x^+\phi_{i,j}+s_{x,1}\frac{\Delta x}{2}D_x^+D_x^+\phi_{i,j}+s_{x,1}s_{x,2}\frac{\Delta x^2}{6}D_x^+D_x^+D_x^+\phi_{i,j},0\Bigg)^2\\&+\max\Bigg(D_y^-\phi_{i,j}+s_{y,-1}\frac{\Delta y}{2}D_y^-D_y^-\phi_{i,j}+s_{y,-1}s_{y,-2}\frac{\Delta y^2}{6}D_y^-D_y^-D_y^-\phi_{i,j},\\&\qquad -D_y^+\phi_{i,j}+s_{y,1}\frac{\Delta y}{2}D_y^+D_y^+\phi_{i,j}+s_{y,1}s_{y,2}\frac{\Delta y^2}{6}D_y^+D_y^+D_y^+\phi_{i,j},0\Bigg)^2=\frac{1}{F_{i,j}^2},\end{aligned} \tag{2.4}$$

where the switches are

$$s_{x,k}=\begin{cases}1, & \mathbf{x}_{i+k,j}\in A,\\ 0 & \text{otherwise,}\end{cases}$$

$$s_{y,k}=\begin{cases}1, & \mathbf{x}_{i,j+k}\in A,\\ 0 & \text{otherwise.}\end{cases}$$

Now the fast marching method can be assembled as an algorithm:

1. Initialize all the points adjacent to the initial interface with an initial value; put those points in A. A discussion about initialization follows in section 3. All points $\mathbf{x}_{i,j} \notin A$ but are adjacent to a point in A are given initial estimates for $\phi_{i,j}$ by solving (2.4). These points are tentative points and put in the set T. All remaining points unaccounted for are placed in D and given initial value of $\phi_{i,j} = +\infty$.
2. Choose the point $\mathbf{x}_{i,j} \in T$ which has the smallest value of $\phi_{i,j}$ and move it into A. Any point which is adjacent to $\mathbf{x}_{i,j}$ (i.e., the points $\mathbf{x}_{i-1,j}$, $\mathbf{x}_{i,j-1}$, $\mathbf{x}_{i+1,j}$, $\mathbf{x}_{i,j+1}$) which is in T has its value $\phi_{i,j}$ recalculated using (2.4). Any point adjacent to $\mathbf{x}_{i,j}$ and in D has its value $\phi_{i,j}$ computed using (2.4) and is moved into the set T.
3. If $T \neq \emptyset$, go to step 2.

The algorithm presented in [12] is first order accurate and can be recovered from (2.4) by taking all the switches $s_{\bullet,\bullet} = 0$. The second order accurate method presented in [13] can also be recovered from (2.4) by taking all the switches $s_{\bullet,\pm 2} = 0$.

3. Locally second order approximation of the level set function.

In [13], it is shown that the fast marching method behaves as a globally second order accurate method. However, it is also noted in [13] that the method is only first order accurate in the set of nodes immediately adjacent to the initial front. This does not destroy the global accuracy because as the mesh shrinks, so too does the size of the region of first order accuracy. To achieve a fully second order accurate fast marching method, we must ensure that the initial data is at least second order accurate. Furthermore, we can also get a globally third order accurate fast marching method by virtue of having at least second order initial data.

The underpinning of this higher degree of accuracy around the initial front is the use of a bicubic interpolation function p which is a second order accurate local representation of a level set function ϕ (i.e., $p(\mathbf{x}) \approx \phi(\mathbf{x})$). The interpolation function $p(\mathbf{x})$ can serve many purposes including second order accuracy for the distance to the zero level set, subgrid resolution of the shape of the interface, as well as subgrid resolution of the level set function $\phi(\mathbf{x})$ itself.

3.1. The bicubic interpolation.

We begin with a description of the bicubic interpolation for a level set function given on a rectangular mesh. The approximation is done locally in a box of the mesh bounded by grid points; call them $\mathbf{x}_{i,j}$, $\mathbf{x}_{i+1,j}$, $\mathbf{x}_{i,j+1}$, and $\mathbf{x}_{i+1,j+1}$, as in Figure 3.1.

A bicubic interpolation $p(\mathbf{x})$ of a function $\phi(\mathbf{x})$ is a function

$$p(\mathbf{x}) = p(x,y) = \sum_{m=0}^{3}\sum_{n=0}^{3} a_{m,n} x^m y^n \tag{3.1}$$

which solves the following set of equations:

$$\begin{aligned} p(\mathbf{x}_{k,\ell}) &= \phi(\mathbf{x}_{k,\ell}), \\ \frac{\partial p}{\partial x}(\mathbf{x}_{k,\ell}) &= \frac{\partial \phi}{\partial x}(\mathbf{x}_{k,\ell}), \\ \frac{\partial p}{\partial y}(\mathbf{x}_{k,\ell}) &= \frac{\partial \phi}{\partial y}(\mathbf{x}_{k,\ell}), \\ \frac{\partial^2 p}{\partial x \partial y}(\mathbf{x}_{k,\ell}) &= \frac{\partial^2 \phi}{\partial x \partial y}(\mathbf{x}_{k,\ell}) \end{aligned}$$

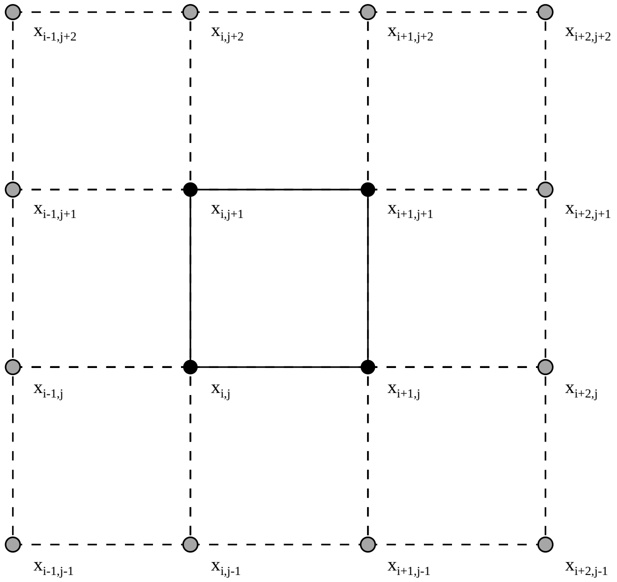


FIG. 3.1. *Sample portion of the mesh where a bicubic interpolation is used.*

for $k = i,\ i+1;\ \ell = j,\ j+1$. This gives sixteen equations for the sixteen unknown coefficients $a_{m,n}$. Solving for the $a_{m,n}$ makes $p(x, y)$ a bicubic interpolating function of $\phi(x, y)$ on the rectangle bounded by the corners $\mathbf{x}_{i,j}$, $\mathbf{x}_{i+1,j}$, $\mathbf{x}_{i,j+1}$, and $\mathbf{x}_{i+1,j+1}$.

Since ϕ is known only on the mesh points, the values for the derivatives of ϕ must be approximated. We use second order finite difference approximations for the derivatives of ϕ:

$$\begin{aligned}
\frac{\partial\phi}{\partial x}(\mathbf{x}_{m,n}) &\approx \frac{1}{2\Delta x}(\phi(\mathbf{x}_{m+1,n}) - \phi(\mathbf{x}_{m-1,n})),\\
\frac{\partial\phi}{\partial y}(\mathbf{x}_{m,n}) &\approx \frac{1}{2\Delta y}(\phi(\mathbf{x}_{m,n+1}) - \phi(x_{m,n-1})),\\
\frac{\partial^2\phi}{\partial x\partial y}(\mathbf{x}_{m,n}) &\approx \frac{1}{4\Delta x\Delta y}(\phi(\mathbf{x}_{m+1,n+1}) - \phi(\mathbf{x}_{m-1,n+1})\\
&\qquad\qquad - \phi(\mathbf{x}_{m+1,n-1}) + \phi(\mathbf{x}_{m-1,n-1}))
\end{aligned}$$

for $m = i, i+1$ and $n = j, j+1$. Thus, construction of the interpolant p requires all the points shown in Figure 3.1. The observant reader will note that the accuracy of this method is restricted to second order by our use of second order approximations for the derivatives of ϕ. In principle, higher order local approximations can be made using higher order finite difference approximations and using a larger set of grid points around the box where the interpolant is used.

One useful property of this local interpolation method is that interpolations in neighboring boxes match smoothly along their common edge. To see this, consider two interpolations $p(x, y)$ and $q(x, y)$ interpolating in two adjacent boxes as depicted in Figure 3.2. To show continuity, we must show $p(x_i, y) = q(x_i, y)$ for $y_j \leq y \leq y_{j+1}$. Now, $p(x_i, y)$ and $q(x_i, y)$ are both cubic polynomials in y. Furthermore, by construction, $p(x_i, y_j) = q(x_i, y_j)$, $p(x_i, y_{j+1}) = q(x_i, y_{j+1})$, $\frac{\partial p}{\partial y}(x_i, y_j) = \frac{\partial q}{\partial y}(x_i, y_j)$,

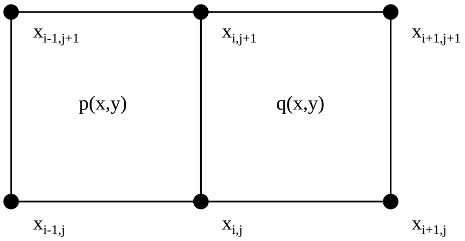


FIG. 3.2. *Two neighboring bicubic interpolants.*

and $\frac{\partial p}{\partial y}(x_i, y_{j+1}) = \frac{\partial q}{\partial y}(x_i, y_{j+1})$. These four conditions uniquely determine the cubic polynomial along the common edge; hence $p(x_i, y) \equiv q(x_i, y)$.

We can go one step further by noting that $\frac{\partial p}{\partial x}(x_i, y)$ and $\frac{\partial q}{\partial x}(x_i, y)$ are also cubic polynomials in y and by construction, $\frac{\partial p}{\partial x}(x_i, y_j) = \frac{\partial q}{\partial x}(x_i, y_j)$, $\frac{\partial p}{\partial x}(x_i, y_{j+1}) = \frac{\partial q}{\partial x}(x_i, y_{j+1})$, $\frac{\partial^2 p}{\partial x \partial y}(x_i, y_j) = \frac{\partial^2 q}{\partial x \partial y}(x_i, y_j)$, and $\frac{\partial^2 p}{\partial x \partial y}(x_i, y_{j+1}) = \frac{\partial^2 q}{\partial x \partial y}(x_i, y_{j+1})$. As above, these four equations uniquely determine the cubic polynomial $\frac{\partial p}{\partial x}(x_i, y) \equiv \frac{\partial q}{\partial x}(x_i, y)$.

If we differentiate the two results $p(x_i, y) \equiv q(x_i, y)$, $\frac{\partial p}{\partial x}(x_i, y) \equiv \frac{\partial q}{\partial x}(x_i, y)$ with respect to y, we get the additional equations $\frac{\partial p}{\partial y}(x_i, y) \equiv \frac{\partial q}{\partial y}(x_i, y)$, $\frac{\partial^2 p}{\partial x \partial y}(x_i, y) \equiv \frac{\partial^2 q}{\partial x \partial y}(x_i, y)$. Therefore, the combined bicubic interpolation over two adjacent boxes is at least $C^{(1)}$.

The approximation $p(x, y)$ is now a local subgrid representation of the level set function ϕ inside the box and can be used as an initializer for a fast marching or velocity extension method.

3.2. Reinitialization. Reinitialization is the construction of the distance map to the zero level set of a given level set function ψ. Thus, given $\psi(\mathbf{x})$, we want to construct $\phi(\mathbf{x})$ such that $\phi^{-1}(0) = \psi^{-1}(0)$ and $\|\nabla\phi\| = 1$. Reinitialization was first introduced in [3], where it was shown to improve the stability of the level set method. It continues to play a crucial role in many level set applications, for example, narrow-band level set methods [1], incompressible fluid flow [18], and computer-aided design [6], to name a few.

One of the main drawbacks of using reinitialization in level set methods is the difficulty in maintaining the original position of the interface, often leading to breakdown in conservation of area/volume. This problem surfaced in the original method proposed in [3] and persists today as observed in [17]. To overcome this problem with the level set method, different solutions have been proposed. Adalsteinsson and Sethian [2] used velocity extensions constructed with the fast marching method to drastically reduce or even eliminate the need for reinitialization in level set method applications. However, their velocity extension method is only first order accurate in a neighborhood of the interface. Sussman and Fatemi [17] proposed a mass correction procedure which artificially enforces mass conservation by modifying the front velocity. While this method forces mass conservation by construction, it comes at the price of modifying the original front velocity with a nonphysical correction term.

In this paper, we focus on reinitialization because it is a stepping stone to initializing the more general fast marching method, and also because it is important to the level set method on its own. The method for reinitialization we present here using bicubic interpolation has two very important properties that previous reinitialization techniques lack: very good front location preservation and, consequently, very good mass conservation. This will be demonstrated by showing that repeated reinitializations do not significantly degrade the location of the interface compared with existing first order methods. In the next section we will describe how this method of reinitialization can be used to solve more general fast marching method applications.

Given a function $\psi(\mathbf{x})$, we want to construct $\phi(\mathbf{x})$ such that $\phi^{-1}(0) = \psi^{-1}(0)$ and $\|\phi\| = 1$. We begin by passing through the mesh to locate all rectangles bounded by nodes which are not all of the same sign, indicating that the zero level set passes through that rectangle. Suppose the rectangle is bounded by the nodes $\mathbf{x}_{i,j}$, $\mathbf{x}_{i+1,j}$, $\mathbf{x}_{i,j+1}$, $\mathbf{x}_{i+1,j+1}$ as in Figure 3.1. Using the values of ψ at these nodes, we construct the bicubic interpolation function $p(\mathbf{x})$ as described above.

Now, let $\mathbf{x}^0$ be a point in the rectangle where we wish to compute the distance to the set $\psi^{-1}(0) \approx p^{-1}(0)$. Let $\mathbf{y}$ be a point in $p^{-1}(0)$ nearest to $\mathbf{x}^0$; then it follows that

$$p(\mathbf{y}) = 0, \tag{3.2}$$

$$\nabla p(\mathbf{y}) \times (\mathbf{x}^0 - \mathbf{y}) = 0. \tag{3.3}$$

Equation (3.2) is a requirement that $\mathbf{y}$ must be on the zero contour of p and (3.3) is a requirement that the normal to the zero contour, given by $\nabla p(\mathbf{y})$, must be aligned with the line through the points $\mathbf{x}^0$, $\mathbf{y}$.

To solve (3.2)–(3.3), we employ a variant of Newton's method to simultaneously solve the pair of conditions. We compute a sequence of iterates $\mathbf{x}^k$ with initial point $\mathbf{x}^0$. The update for the iterates is given by

$$\boldsymbol{\delta}_1 = -p(\mathbf{x}^k)\frac{\nabla p(\mathbf{x}^k)}{\nabla p(\mathbf{x}^k) \cdot \nabla p(\mathbf{x}^k)}, \tag{3.4}$$

$$\mathbf{x}_{k+1/2} = \mathbf{x}^k + \boldsymbol{\delta}_1, \tag{3.5}$$

$$\boldsymbol{\delta}_2 = (\mathbf{x}^0 - \mathbf{x}^k) - \frac{(\mathbf{x}^0 - \mathbf{x}^k) \cdot \nabla p(\mathbf{x}^k)}{\nabla p(\mathbf{x}^k) \cdot \nabla p(\mathbf{x}^k)}\nabla p(\mathbf{x}_k), \tag{3.6}$$

$$\mathbf{x}_{k+1} = \mathbf{x}_{k+1/2} + \boldsymbol{\delta}_2. \tag{3.7}$$

This method takes advantage of the orthogonality of the solution sets for (3.2) and (3.3). Equation (3.4) is the Newton step for the function $g_1(t) = p(\mathbf{x}^k + t\nabla p(\mathbf{x}^k))$. This step moves in the direction of ∇p, and hence holds the value of the left side of (3.3) approximately fixed. Equation (3.6) is the Newton step for the function $g_2(t) = \mathbf{k} \cdot (\nabla p(\mathbf{x}^k) \times (\mathbf{x}_0 - (\mathbf{x}^k + t(\mathbf{k} \times \nabla p(\mathbf{x}^k)))))$. This step moves in the direction normal to ∇p, and hence holds the value of p approximately fixed. Figure 3.3 illustrates this algorithm.

We found the number of iterations required to obtain convergence using this two-step Newton's method was half the standard Newton step or less, but we do not believe the convergence rate differs from the standard Newton iteration. Should this method prove more generally useful, a more complete study of this method will be undertaken by the author.

For the results presented in section 4, we used the convergence criterion

$$\sqrt{\|\boldsymbol{\delta}_1\|^2 + \|\boldsymbol{\delta}_2\|^2} < 10^{-3}\Delta x \Delta y,$$

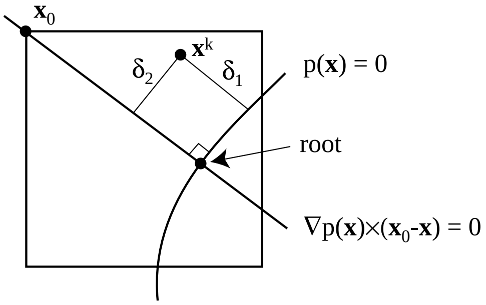


FIG. 3.3. *Example of solution sets for* (3.2) *and* (3.3).

where Δx, Δy are the space step sizes in the x and y directions, respectively. For nearly all evaluations, two or three iterations are necessary to meet the criterion. The distance to the front from the point $\mathbf{x}^0$ is now given by $\phi(\mathbf{x}^0) = \pm\|\mathbf{x}^0 - \mathbf{x}^\infty\|$, where the sign of $\phi(\mathbf{x}^0)$ is taken to be the same as that of $p(\mathbf{x}^0)$.

The above process gives more than just the distance to the zero level set of p; it also gives the location of the nearest point, $\mathbf{x}^\infty$. This will have important consequences later when we discuss the application of this method to velocity extensions and the general fast marching method.

Reinitialization now proceeds by using the above process to calculate the distance to the zero contour from each of the nodes $\mathbf{x}_{i,j}$, $\mathbf{x}_{i+1,j}$, $\mathbf{x}_{i,j+1}$, $\mathbf{x}_{i+1,j+1}$, i.e., each of these nodes plays the role of $\mathbf{x}^0$ above. Now, some of these nodes will be involved in multiple interpolation procedures, so the final value taken for ϕ at those nodes is the minimum of all computed distances to the front. To initialize the fast marching method to complete reinitialization, all nodes whose distance values are constructed using the bicubic interpolation are placed in the set A of accepted points.

The modified Newton iteration will not always converge. Divergence generally can happen when there is no solution for the pair of equations inside the box formed by the nodes $\mathbf{x}_{i,j}$, $\mathbf{x}_{i+1,j}$, $\mathbf{x}_{i,j+1}$, $\mathbf{x}_{i+1,j+1}$. In this event, the calculated distance is taken to be infinite. The iteration rarely requires more than five iterations, and hence twenty iterations is taken to be the maximum before divergence is determined.

3.3. Application to general fast marching method. The above construction was designed to solve the problem of reinitialization. To apply this method for more general fast marching methods, note that the initialization of the general fast marching method requires two components: the local front speed F and the initial distance to the front ϕ. The front speed is determined by the application, and the above process produces the initial distance to the front. Thus, the initial value for a point adjacent to the initial front for the general fast marching method solving (2.1) is ϕ/F. The same set of initially accepted points A are used for general fast marching as for reinitialization.

The fast marching method is also used to construct extension velocity fields for the level set method [2]. Extended velocity fields are used in the level set method when the front velocity F is defined only on the zero level set. In order to advance the level set method, a value of F must exist for the entire domain of the level set function ϕ, not just at the zero contour. The extended velocity field F_{ext} is a speed function defined on the domain of the level set function which satisfies the conditions

$$F_{\text{ext}}\big|_{\phi^{-1}(0)} = F\big|_{\phi^{-1}(0)}, \tag{3.8}$$

$$\nabla F_{\text{ext}} \cdot \nabla \phi = 0, \tag{3.9}$$

where ϕ is again the signed distance to the front. The solution of (3.9) is accomplished using a technique based on the fast marching method; for details, see [2]. In this application, the speed function F_{ext} must be initialized in the same way as for the fast marching method. In this case, the actual location on the front nearest a given node in the mesh is useful, because we can now set $F_{\text{ext}}(\mathbf{x}^0) = F(\mathbf{x}^\infty)$. This solves (3.9) because F_{ext} is now constant on a line between $\mathbf{x}^\infty$ and $\mathbf{x}^0$, which means that

$$\nabla F_{\text{ext}} \perp (\mathbf{x}^\infty - \mathbf{x}^0).$$

But by construction,

$$(\mathbf{x}^\infty - \mathbf{x}^0) \parallel \nabla \phi(\mathbf{x}^\infty).$$

Therefore,

$$F_{\text{ext}}(\mathbf{x}^0) \cdot \nabla \phi(\mathbf{x}^\infty) = 0.$$

Solving (3.9) at the remainder of the nodes is done using fast marching where the heap sort sorts points according to their ϕ value.

4. Results. In this section, we demonstrate the advantages of this new method for initializing the fast marching method.

4.1. Accuracy of the local distance map. The first test of this method is to show that the initialization method is second order accurate in a neighborhood of the initial zero level set. To test this, we show that the error in the computed distance function to a given zero level set is second order accurate. We measure the error in only those points which are computed via the bicubic interpolation method and exclude all points which are subsequently computed by the fast marching method.

The tests were conducted on three initial contours, which are shown in Figure 4.1. We see in Table 4.1 that the initialization method presented in [13] behaves as a first order approximation, and the bicubic interpolation method shows second order convergence as shown in Table 4.2 for smooth functions. Note that for test cases with corners or cusps (the second two examples), the bicubic interpolation also reduces to first order due to the local discontinuity in the derivatives of the level set function.

The computational cost of the new initialization method compared with the old method increased by 20% for an 80 × 80 mesh, down to less than a 4% increase for the 320 × 320 mesh. This is expected because the more refined the mesh, the quicker the iterative step converges on average, and the overall computational cost of the initialization procedure is $O(N)$ with the number of nodes N compared with the fast marching method which increases in cost by $O(N \log N)$.

4.2. Order of accuracy of the fast marching method. Since we now have an initialization procedure which is second order accurate, we should in theory be able to build a third order fast marching method as given in (2.4). For comparison purposes, Table 4.3 shows the results of applying the second order method from [13] with first order initialization. Table 4.4 shows the error measurements for the third

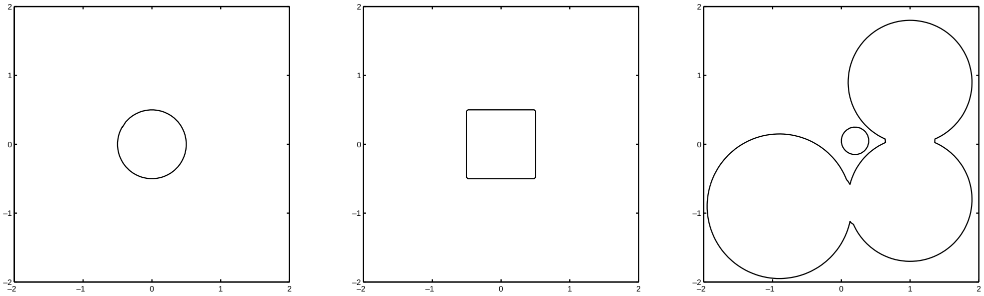


Fig. 4.1. *Three test cases for the new method.*

Table 4.1
Error measurements for linear initialization method.

Test case	Mesh size	L_1 error	L_2 error	L_∞ error
Circle	20×20	8.93×10^{-3}	1.05×10^{-2}	1.43×10^{-2}
	40×40	1.41×10^{-3}	1.81×10^{-3}	3.67×10^{-3}
	80×80	3.71×10^{-4}	4.87×10^{-4}	1.19×10^{-3}
	160×160	1.02×10^{-4}	1.36×10^{-4}	2.93×10^{-4}
	320×320	2.47×10^{-5}	3.14×10^{-5}	6.97×10^{-5}
Square	20×20	7.71×10^{-3}	2.04×10^{-3}	5.40×10^{-2}
	40×40	5.93×10^{-4}	2.58×10^{-3}	1.12×10^{-2}
	80×80	1.43×10^{-4}	8.91×10^{-4}	5.56×10^{-3}
	160×160	3.50×10^{-5}	3.11×10^{-4}	2.76×10^{-3}
	320×320	9.58×10^{-7}	2.10×10^{-5}	4.59×10^{-4}
Disks	20×20	7.60×10^{-3}	1.73×10^{-2}	9.14×10^{-2}
	40×40	1.70×10^{-3}	4.82×10^{-3}	4.57×10^{-2}
	80×80	5.71×10^{-4}	2.73×10^{-3}	3.92×10^{-2}
	160×160	1.31×10^{-4}	8.10×10^{-4}	1.43×10^{-2}
	320×320	3.01×10^{-5}	2.39×10^{-4}	6.23×10^{-3}

Table 4.2
Error measurements for bicubic initialization.

Test case	Mesh size	L_1 error	L_2 error	L_∞ error
Circle	20×20	5.73×10^{-4}	9.29×10^{-4}	1.81×10^{-3}
	40×40	5.68×10^{-5}	7.17×10^{-5}	1.41×10^{-4}
	80×80	6.07×10^{-6}	7.25×10^{-6}	1.36×10^{-5}
	160×160	6.34×10^{-7}	8.03×10^{-7}	1.96×10^{-6}
	320×320	9.38×10^{-8}	1.20×10^{-7}	2.74×10^{-7}
Square	20×20	1.00×10^{-2}	2.01×10^{-2}	4.94×10^{-2}
	40×40	1.55×10^{-3}	3.98×10^{-3}	1.33×10^{-2}
	80×80	3.82×10^{-4}	1.39×10^{-3}	6.57×10^{-3}
	160×160	9.49×10^{-5}	4.88×10^{-4}	3.26×10^{-3}
	320×320	1.30×10^{-5}	1.81×10^{-4}	2.79×10^{-3}
Disks	20×20	5.74×10^{-3}	1.80×10^{-2}	1.41×10^{-1}
	40×40	2.00×10^{-3}	7.29×10^{-3}	7.63×10^{-2}
	80×80	3.46×10^{-4}	1.43×10^{-3}	1.70×10^{-2}
	160×160	1.11×10^{-4}	1.04×10^{-3}	2.36×10^{-2}
	320×320	2.04×10^{-5}	2.68×10^{-4}	8.47×10^{-3}

order method. This method appears to be third order from the error measurements for the circle example. Note that the higher rate of convergence does not apply to the second two test cases because the exact solution is not smooth. We show them only to illustrate how the method behaves for more difficult problems.

Table 4.3
Error measurements for linear initialization method.

Test case	Mesh size	L_1 error	L_2 error	L_∞ error
Circle	20×20	2.65×10^{-2}	4.33×10^{-2}	1.68×10^{-1}
	40×40	4.17×10^{-3}	5.02×10^{-3}	1.78×10^{-2}
	80×80	1.03×10^{-3}	1.30×10^{-3}	9.79×10^{-3}
	160×160	1.60×10^{-4}	2.29×10^{-4}	5.07×10^{-3}
	320×320	3.27×10^{-5}	5.47×10^{-5}	2.57×10^{-3}
Square	20×20	5.40×10^{-2}	6.58×10^{-2}	2.05×10^{-1}
	40×40	9.76×10^{-3}	1.32×10^{-2}	2.72×10^{-2}
	80×80	4.53×10^{-3}	6.58×10^{-3}	1.50×10^{-2}
	160×160	2.17×10^{-3}	3.30×10^{-3}	7.93×10^{-3}
	320×320	2.06×10^{-3}	4.04×10^{-3}	1.60×10^{-2}
Disks	20×20	3.27×10^{-2}	7.28×10^{-2}	3.53×10^{-1}
	40×40	1.30×10^{-2}	3.26×10^{-2}	2.09×10^{-1}
	80×80	4.12×10^{-3}	2.00×10^{-2}	2.71×10^{-1}
	160×160	2.66×10^{-3}	1.59×10^{-2}	2.60×10^{-1}
	320×320	2.21×10^{-3}	1.43×10^{-2}	2.52×10^{-1}

Table 4.4
Error measurements for bicubic initialization.

Test case	Mesh size	L_1 error	L_2 error	L_∞ error
Circle	20×20	8.23×10^{-3}	1.13×10^{-2}	3.27×10^{-2}
	40×40	1.00×10^{-3}	1.68×10^{-3}	8.56×10^{-3}
	80×80	1.76×10^{-4}	3.20×10^{-4}	4.71×10^{-3}
	160×160	2.73×10^{-5}	6.88×10^{-5}	2.38×10^{-3}
	320×320	4.38×10^{-6}	1.48×10^{-5}	1.19×10^{-3}
Square	20×20	4.28×10^{-2}	6.15×10^{-2}	1.35×10^{-1}
	40×40	8.48×10^{-3}	1.17×10^{-2}	2.55×10^{-2}
	80×80	3.80×10^{-3}	5.52×10^{-3}	1.29×10^{-2}
	160×160	1.79×10^{-3}	2.67×10^{-3}	6.62×10^{-3}
	320×320	2.32×10^{-4}	9.13×10^{-4}	4.30×10^{-3}
Disks	20×20	9.85×10^{-3}	2.62×10^{-2}	1.61×10^{-1}
	40×40	5.16×10^{-3}	1.79×10^{-2}	1.94×10^{-1}
	80×80	3.50×10^{-3}	1.74×10^{-2}	2.60×10^{-1}
	160×160	2.43×10^{-3}	1.45×10^{-2}	2.54×10^{-1}
	320×320	2.20×10^{-3}	1.40×10^{-2}	2.44×10^{-1}

4.3. Conservation of mass. One common criticism of the level set method concerns the conservation properties of the method. The primary source of variation in mass is due to the process of reinitialization. In some applications of the level set method, such as narrow band methods, for example, frequent reinitializations are used. Repeated reinitializations eventually lead to problems in mass conservation. In Figure 4.2, we show that after twenty reinitializations, the location of the zero level set, and hence the measured value of the area bounded by the zero level set, is very well preserved. Table 4.5 shows the calculated area of the center circle after repeated reinitializations for both the old linear method and the new bicubic method.

4.4. Extension to higher dimensions. Another advantage of the initialization method presented in this paper concerns the extension to higher dimensions. The linear initialization method used does not easily extend to higher dimensions. The method presented in this paper is easily extended to any number of dimensions. For

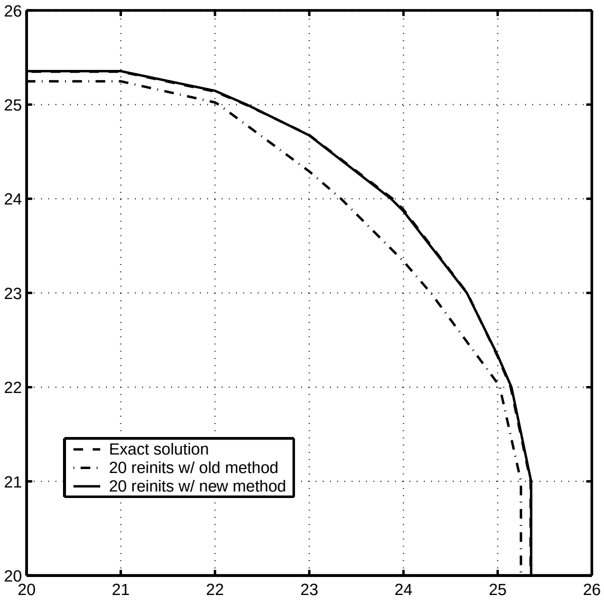


FIG. 4.2. *Deviation after twenty reinitializations.*

TABLE 4.5
Conservation of mass during reinitialization.

Method	Area after 10 reinits	Area after 20 reinits	% error after 20
Exact	0.778683	0.778683	
Linear	0.742338	0.713886	−8%
Bicubic	0.778822	0.779032	0.04%

TABLE 4.6
Error measurements for first order fast marching.

Mesh size	L_1 error	L_2 error	L_∞ error
$20 \times 20 \times 20$	4.49×10^{-2}	5.39×10^{-2}	1.22×10^{-1}
$40 \times 40 \times 40$	2.47×10^{-2}	2.90×10^{-2}	8.04×10^{-2}
$80 \times 80 \times 80$	1.31×10^{-2}	1.53×10^{-2}	5.42×10^{-2}

N dimensions, we use an N-cubic interpolant

$$p(x_1, \dots, x_N) = \sum_{k_1=0}^{3} \cdots \sum_{k_N=0}^{3} a_{k_1,\dots,k_N} \prod_{\ell=1}^{N} x_\ell^{k_\ell}.$$

The iterative procedure given by (3.4)–(3.7) is still valid in this higher dimensional framework. We employed this method to reinitialize a sphere in $\mathbb{R}^3$ and used the second order fast marching method to get a second order accurate method including near the zero level set. Table 4.6 shows the errors from the original first order method from [12]. Table 4.7 shows the bicubic initialization method using second order fast marching. Table 4.8 shows the third order fast marching method results.

Table 4.7

Error measurements for bicubic initialization method, second order fast marching.

Mesh size	L_1 error	L_2 error	L_∞ error
$20 \times 20 \times 20$	5.00×10^{-3}	6.36×10^{-3}	3.91×10^{-2}
$40 \times 40 \times 40$	1.66×10^{-3}	2.05×10^{-3}	2.54×10^{-2}
$80 \times 80 \times 80$	4.84×10^{-4}	5.87×10^{-4}	1.39×10^{-2}

Table 4.8

Error measurements for bicubic initialization method, third order fast marching.

Mesh size	L_1 error	L_2 error	L_∞ error
$20 \times 20 \times 20$	2.15×10^{-3}	2.91×10^{-3}	2.16×10^{-2}
$40 \times 40 \times 40$	3.74×10^{-4}	6.25×10^{-4}	1.22×10^{-2}
$80 \times 80 \times 80$	6.62×10^{-5}	1.37×10^{-4}	6.12×10^{-3}

5. Conclusion. In this paper, we have presented a new interpolation method for initializing the fast marching method. This new method makes higher order accurate fast marching methods possible. Furthermore, the new method extends to higher dimensions much more readily than the old first order method. The new method is more accurate than existing methods and the additional cost is minimal. The additional accuracy also significantly improves the mass conservation of the level set method.

The additional accuracy has already proven useful in practice. We have used this new technique to do both reinitialization and front evolution using fast marching for fatigue crack growth modeling in three dimensions [16]. In this work, this higher order fast marching method is used in three different ways. First, reinitialization is used to construct the signed distance from the crack front. This is required to be as accurate as possible to obtain accurate crack front velocity values. Second, after the crack front velocities are obtained through a finite element calculation, the velocities are extended using the velocity extension method described above. Third, the crack front is advanced using the fast marching method and the extended velocity values. All three steps make use of the bicubic interpolation scheme. This approach is different than most uses of the level set method and is taken because the velocity calculation is by far the most expensive part of the process. By using the fast marching method instead of the level set method, arbitrarily large time steps can be taken to advance the front.

The application presented in [16] limits the crack to a single plane in a three-dimensional solid, and hence all the fast marching applications are done on a two-dimensional domain. When the crack front is allowed to move in arbitrary nonplanar directions, the fast marching method will be done in a three-dimensional domain. In that case, this higher order interpolation method will be critical because the refinement of the level set mesh will by necessity be much coarser than for two dimensions.

The bicubic interpolation presented here will produce at best a third order accurate fast marching method. Higher order fast marching methods are possible by using biquintic interpolation. The construction of the interpolant will require a larger stencil of nodes than shown in Figure 3.1 to approximate the additional derivative conditions, and the finite difference derivative approximations will require additional points to obtain sufficient accuracy. For initialization of the fast marching method, the iterative scheme (3.4)–(3.7) for computing the distance to the zero level set remains the same. The fast marching method would also have to be modified to be higher order by adding more terms and more switches.

There is a limit to the order of accuracy of the fast marching method. It is interesting to note that in one dimension, the fast marching method reduces to a backward difference formula method. These methods are known to be stable only for order less than seven. Convergence of the second and higher order fast marching methods is still not proven, and there is some question about their behavior in very dense meshes. The convergence of the higher order methods and their relationship to backward difference formulas is the subject of future research.

REFERENCES

[1] D. Adalsteinsson and J. Sethian, *A fast level set method for propagating interfaces*, J. Comput. Phys., 118 (1995), pp. 269–277.

[2] D. Adalsteinsson and J. Sethian, *The fast construction of extension velocities in level set methods*, J. Comput. Phys., 48 (1999), pp. 2–22.

[3] D. L. Chopp, *Computing minimal surfaces via level set curvature flow*, J. Comput. Phys., 106 (1993), pp. 77–91.

[4] E. Dijkstra, *A note on two problems in connection with graphs*, Numer. Math., 1 (1959), pp. 269–271.

[5] J. Glimm, J. W. Grove, X. L. Li, and D. C. Tan, *Robust computational algorithms for dynamic interface tracking in three dimensions*, SIAM J. Sci. Comput., 21 (2000), pp. 2240–2256.

[6] R. Kimmel and A. M. Bruckstein, *Shape offset via level sets*, Comput.-Aided Des., 25 (1993), pp. 154–162.

[7] R. Kimmel and J. Sethian, *Fast Marching Methods for Computing Distance Maps and Shortest Paths*, Tech. Report 669, CPAM, University of California, Berkeley, 1996.

[8] R. Malladi and J. Sethian, *An $O(N \log N)$ algorithm for shape modeling*, Proc. Natl. Acad. Sci. USA, 93 (1996), pp. 9389–9392.

[9] S. Osher and J. A. Sethian, *Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations*, J. Comput. Phys., 79 (1988), pp. 12–49.

[10] J. Sethian, *Fast marching level set methods for three-dimensional photolithography development*, in Proceedings of the SPIE 1996 International Symposium on Microlithography, Santa Clara, CA, 1996.

[11] J. Sethian, *Level Set Methods: Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision and Material Science*, Cambridge University Press, Cambridge, UK, 1996.

[12] J. Sethian, *A marching level set method for monotonically advancing fronts*, Proc. Natl. Acad. Sci. USA, 93 (1996), pp. 1591–1595.

[13] J. Sethian, *Fast marching methods*, SIAM Rev., 41 (1999), pp. 199–235.

[14] J. Sethian and A. Popovici, *Three dimensional traveltimes computation using the fast marching method*, Geophysics, 64 (1999), pp. 516–523.

[15] M. Stolarska, D. L. Chopp, N. Möes, and T. Belytschko, *Modelling crack growth by level sets in the extended finite element method*, Internat. J. Numer. Methods Engrg., 51 (2001), pp. 943–960.

[16] N. Sukumar, D. L. Chopp, and B. Moran, *Extended finite element method and fast marching method for three-dimensional fatigue crack propagation*, J. Comput. Phys., submitted.

[17] M. Sussman and E. Fatemi, *An efficient interface-preserving level set redistancing algorithm and its application to interfacial incompressible fluid flow*, SIAM J. Sci. Comput., 20 (1999), pp. 1165–1191.

[18] M. Sussman, P. Smereka, and S. Osher, *A level set approach for computing solutions to incompressible two-phase flow*, J. Comput. Phys., 94 (1994), pp. 146–159.

SIAM J. SCI. COMPUT.
Vol. 23, No. 1, pp. 245–270

ON SOME MIXED FINITE ELEMENT METHODS WITH NUMERICAL INTEGRATION*

STEFANO MICHELETTI†, RICCARDO SACCO†, AND FAUSTO SALERI‡

Abstract. In this paper a new family of mixed finite volume methods is analyzed for the approximation of a reaction-diffusion problem. All the methods are obtained starting from the dual mixed formulation of the problem and then employing the lowest-order Raviart–Thomas finite element spaces plus a suitable quadrature formula for the mass matrix. This allows for the use of different averages of the inverse diffusion coefficient to enforce the constitutive law for the fluxes at the interelement boundary in a finite volume fashion. The soundness of the methods is supported by an error analysis which shows optimal $\mathcal{O}(h)$ convergence rate with respect to the standard mixed finite element norm in the case of both smooth and piecewise smooth coefficients. Numerical results on test problems with both smooth and nonsmooth coefficients support the theoretical estimates.

Key words. mixed finite volume methods, quadrature formulae, error estimates

AMS subject classifications. 65N12, 65N15, 35J25, 35R05

PII. S1064827500348788

1. Introduction.

In this article we propose a unified framework for the analysis of a family of cell-centered finite volume schemes for the approximation of the Dirichlet problem for the reaction-diffusion operator $Lu = -\text{div}\ (\mu\underline{\nabla} u) + \sigma u$ on a polygonal domain $\Omega \subset \mathbb{R}^2$. The novel approach is based on the dual mixed discretization of the differential problem by the lowest-order Raviart–Thomas (RT) finite element space [35] on a given triangulation $\mathcal{T}_h$ of $\overline{\Omega}$.

Introducing the auxiliary flux variable $\underline{J} = \mu\underline{\nabla} u$ and denoting by u_h and $\underline{J}_h$ the RT approximations of u and $\underline{J}$, the linear system associated with the dual mixed discretization is of the form

$$\begin{pmatrix} A & B^T \\ -B & C \end{pmatrix} \begin{pmatrix} \mathbf{J} \\ \mathbf{u} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{f} \end{pmatrix}, \tag{1.1}$$

where $\mathbf{J}$ and $\mathbf{u}$ are the vectors of the degrees of freedom for $\underline{J}_h$ and u_h. The dual mixed formulation provides the interelement continuity of the normal traces of $\underline{J}_h$ and yields the same order of accuracy for both u_h and $\underline{J}_h$ (see [35, 12, 36, 34]).

This is quite desirable in practical applications where there is the need of providing a good approximation of a vector field as it enters other equations as an input datum. See, for example, [15, 16] for the case of the semiconductor problem.

However, two kind of difficulties arise when dealing with system (1.1).

First, the solution procedure can be quite expensive since the matrix acting on u_h, obtained after eliminating the flux unknowns from the first equation of (1.1), is full. Moreover, such a matrix, although symmetric and positive-definite, is not in

*Received by the editors March 1, 2000; accepted for publication (in revised form) February 7, 2001; published electronically June 19, 2001. This research has been supported by MURST 40% and MURST 60% research funds and by the CNR grants in the Progetto Strategico "Applicazioni della Matematica per la Tecnologia e la Società."

http://www.siam.org/journals/sisc/23-1/34878.html

†Dipartimento di Matematica "F. Brioschi," Politecnico di Milano, Via Bonardi 9, 20133 Milano, Italy (mike@mate.polimi.it, ricsac@mate.polimi.it).

‡Dipartimento di Matematica "F. Enriques," Università degli Studi di Milano, Via Saldini 50, 20133 Milano, Italy (fausto.saleri@mat.unimi.it).

general an M-matrix. (See [32] for an example in the one-dimensional case, and see [27] and [12, Remark 5.4, p. 195] in the special case where problem (1.1) arises from semiconductor device simulation.) This can be a serious drawback, especially in those applications where a discrete maximum principle is required (see, e.g., [39], [12, pp. 193–195]).

The usual way to circumvent the first difficulty is to resort to the hybridization of the mixed formulation, introducing Lagrange multipliers λ_h for u_h at the interelement boundaries. The matrix of the resulting system on λ_h, in the case when $\sigma = 0$, is symmetric and positive-definite and turns out to be an M-matrix provided that $\mathcal{T}_h$ is a weakly acute triangulation (see [12, p. 194]). However, to preserve the M-matrix property when $\sigma \neq 0$, it is necessary to change the RT finite element space for $\underline{J}_h$ [27].

An alternative approach that has been the object of several researches in the recent literature is to perform a *lumping* of the matrix A through some suitable quadrature formula.

Concerning the use of rectangular grids, this strategy was first proposed in [32] and was further investigated in [33] in the case when $\mu = 1$. Theoretical and implementation issues that link "nodal" finite elements, mesh-centered finite differences, and mixed-hybrid finite elements have been addressed in [22, 23]. In [31] a theoretical analysis shows that under appropriate smoothness assumptions, the quadrature error (in the evaluation of A and $\mathbf{f}$) does not spoil the accuracy of the mixed method with exact integration.

Passing to triangular elements, similar conclusions have been drawn in [9, 10], where a quadrature formula to diagonalize the matrix A is proposed and examined in the case of the Laplace operator on triangular grids. An extension of this latter lumping procedure has been carried out in [2] and [39] in the case of a diffusion problem with nonconstant coefficients, and in [40, 29] in the case of the convection-diffusion operator with a zero-order term. In particular, in [2] the case of piecewise constant diffusion coefficients is considered with only one harmonic average of μ computed over each triangle. We instead analyze in the present paper a family of averages, all characterized by being piecewise constant over the dual tessellation of the domain. Moreover, we also include in the analysis the zero-order term.

In the previous works [39, 40, 29] the philosophy is the same as in this paper but with two main differences. In the previous works, on the one hand, only a stability analysis was performed on u in a suitable mesh-dependent H^1-seminorm, and on the other hand, only one particular choice of the average of μ was considered, namely, the one corresponding to (8.3) in section 8. In the present work we carry out a complete error analysis for the mixed finite volume formulation in the standard mixed finite element norm.

The case of an elliptic problem where μ is a symmetric, positive-definite tensor has been studied in [5, 14], where RT mixed finite elements of lowest degree with numerical integration are considered on both rectangular and logically rectangular grids. Another numerical scheme on this subject has been proposed in [4], where an expanded mixed finite element method capable of handling the case of a discontinuous full tensor μ and general shape elements and geometry is derived. It can be checked that in the special case of a reference equilateral triangle and RT finite elements of lowest degree, the quadrature rule proposed in [4] to diagonalize the mass matrix yields the same result as the lumping formula of [9], which is at the basis of the discrete formulation proposed in the present article. However, the resulting method in [4] gives a ten-point finite difference stencil, while our method gives a four-point finite difference stencil. Moreover, the error analyses in [4] and in the present paper yield similar orders of convergence and superconvergence properties.

The common feature of the approaches based on a lumping procedure is that the mixed formulation using RT finite elements of lowest degree can be interpreted as a finite volume method acting on the primal unknown u_h. This connection has also been recently investigated in [20], where cell-centered finite difference schemes are constructed on triangular grids of regular shape (equilateral and isosceles right triangles). In this respect we also mention [45], where finite difference schemes on triangular cell-centered grids are derived under the assumption of weakly acute triangulation. In this case, and using the harmonic average of μ along the Voronoi edge, the methods of [45] and of the present paper coincide, although the error analysis in [45] is carried out only in the case when $\mu = 1$.

In the present paper, we provide a unified framework for a family of cell-centered finite volume schemes derived from the dual mixed formulation of the differential problem (see section 2). With this aim, the dual tessellation $\mathcal{L}_h$ associated with $\mathcal{T}_h$ is first introduced in section 3. Notice that $\mathcal{T}_h$ is required to be only a Delaunay triangulation, while previous schemes in the literature assume that $\mathcal{T}_h$ must be a weakly acute triangulation (see [13, 27, 8, 45]). The Delaunay property allows for the presence of obtuse triangles in the mesh, while in a weakly acute triangulation all the angles are required to be $\leq \pi/2$. Next, the RT discretization of the reaction-diffusion problem is considered in section 4, and the finite volume schemes are derived in section 5.

The methods are based on the combined use of a piecewise constant approximation $\overline{\alpha}$ of $\alpha \equiv \mu^{-1}$ over $\mathcal{L}_h$ and of the quadrature formula proposed in [9]. The resulting discretization is a cell-centered finite volume scheme where the degrees of freedom for u_h are picked up at the circumcenters of each element of $\mathcal{T}_h$, while the interelement fluxes are computed using the values of u_h at the circumcenters of two neighboring triangles of $\mathcal{T}_h$.

An abstract framework for the error analysis of the Galerkin discretization with numerical integration of saddle-point problems including zero-order terms is provided in section 6. The results of this section, based on the use of the Strang lemma, are a nontrivial extension to the case of an elliptic operator with zero-order term of the analysis carried out in [12, Chapter 2]. The abstract theory of section 6 is then used in section 7 to prove the (optimal) $\mathcal{O}(h)$ convergence of the new family of methods with respect to the $H(\text{div}; \Omega) \times L^2(\Omega)$-norm that is the standard norm for the analysis of dual mixed methods. We emphasize that the convergence analysis covers the case of both smooth and piecewise smooth coefficients.

The results of section 7 allow us to conclude that, provided $\overline{\alpha}$ is a suitable average of α, it is no longer necessary to resort to the modified RT finite element space introduced in [27]. We also emphasize that the derivation of the novel methods, as well as their convergence analysis, allows us in every respect to call them "mixed finite volume schemes" or, equivalently, "mixed finite element schemes with numerical integration."

Section 8 deals with the computational details of the novel mixed finite schemes. Three choices of $\overline{\alpha}$ are first considered; two averages are suitable approximations of the *harmonic average* of μ (see [7]), while the third one is the trapezoidal rule. The accuracy of the resulting discretization schemes is then examined and compared on two sets of numerical examples with both smooth and discontinuous diffusion coefficient

μ.

2. The continuous problem and its dual mixed formulation.

Let Ω be a bounded convex polygonal domain in $\mathbb{R}^2$ with boundary $\partial\Omega$, and denote by $\underline{n}$ the unit outward normal vector to $\partial\Omega$. We consider the second-order elliptic model problem with homogeneous Dirichlet boundary conditions

$$\left\{ \begin{array}{ll} -\text{div}\,(\mu\underline{\nabla} u) + \sigma u = f & \text{in } \Omega, \\ u = 0 & \text{on } \partial\Omega. \end{array} \right. \tag{2.1}$$

For any open set $\mathcal{S} \subset \mathbb{R}^d$ $(d = 1, 2)$, $|\mathcal{S}|$ and $\chi_{\mathcal{S}}$ will denote, respectively, the d-dimensional Lebesgue measure of $\mathcal{S}$ and the characteristic function of $\mathcal{S}$, while $(\cdot,\cdot)_{0,\mathcal{S}}$ will be the inner product in $L^2(\mathcal{S})$. Norms and seminorms for the spaces $W^{m,p}(\Omega)$ will be denoted by $\|\cdot\|_{m,p,\Omega}$ and $|\cdot|_{m,p,\Omega}$, respectively, with $p = 2$ omitted. Moreover, we shall denote henceforth by $H^m(\Omega)$ the spaces $W^{m,2}(\Omega)$. In particular, $H^0(\Omega) \equiv L^2(\Omega)$, and $H_0^1(\Omega)$ is the Sobolev space of functions of $H^1(\Omega)$ with vanishing traces on $\partial\Omega$ (see [1, 25]).

In (2.1), the diffusion coefficient $\mu \in L^\infty(\Omega)$ is such that $0 < \mu_0 \leq \mu \leq \mu_1$ almost everywhere (a.e.) in Ω, the absorption coefficient $\sigma \in L^\infty(\Omega)$ is nonnegative, and the source term f is a given function in $L^2(\Omega)$.

Using the Lax–Milgram lemma, it is immediate to check that the primal formulation associated with problem (2.1) admits a unique solution $u \in H_0^1(\Omega)$.

Let us introduce the Hilbert space

$$H(\text{div};\Omega) = \{\underline{\tau} \in \left[L^2(\Omega)\right]^2 \mid \text{div}\,\underline{\tau} \in L^2(\Omega)\} \tag{2.2}$$

and define the strictly positive function $\alpha = \mu^{-1} \in L^\infty(\Omega)$ such that $0 < \alpha_0 \leq \alpha \leq \alpha_1$ a.e. in Ω. Thanks to the regularity of u, μ, σ, and f, the vector field

$$\underline{J} = \mu\underline{\nabla} u \tag{2.3}$$

(henceforth denoted as the *flux*) turns out to belong to $H(\text{div};\Omega)$.

Let $V = H(\text{div};\Omega)$ and $Q = L^2(\Omega)$, and define the bilinear forms

$$\left\{ \begin{array}{ll} a(\underline{J},\underline{\tau}) = (\alpha\underline{J},\underline{\tau})_{0,\Omega} = \displaystyle\int_\Omega \alpha\underline{J}\cdot\underline{\tau}\,d\Omega & : V \times V \mapsto \mathbb{R}, \\ b(\underline{\tau},u) = (u,\text{div}\,\underline{\tau})_{0,\Omega} = \displaystyle\int_\Omega u\text{div}\,\underline{\tau}\,d\Omega & : Q \times V \mapsto \mathbb{R}, \\ c(u,v) = (\sigma u,v)_{0,\Omega} = \displaystyle\int_\Omega \sigma uv\,d\Omega & : Q \times Q \mapsto \mathbb{R}. \end{array} \right. \tag{2.4}$$

Then the dual mixed formulation of (2.1) reads

$$\left\{ \begin{array}{ll} \text{Find } \underline{J} \in V \text{ and } u \in Q \text{ such that} & \\ a(\underline{J},\underline{\tau}) + b(\underline{\tau},u) = 0 & \forall\underline{\tau} \in V, \\ -b(\underline{J},v) + c(u,v) = (f,v)_{0,\Omega} & \forall v \in Q. \end{array} \right. \tag{2.5}$$

The existence and uniqueness of $(\underline{J},u)$ will be addressed in section 6.

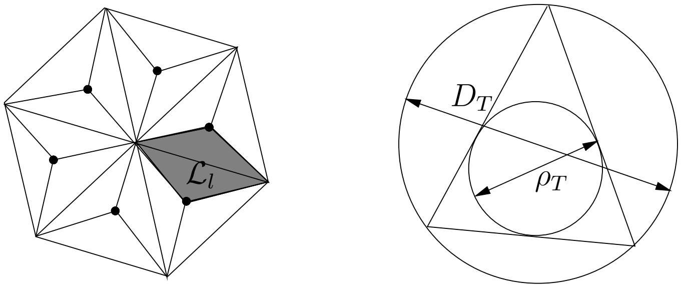


FIG. 1. *Primal triangulation $\mathcal{T}_h$ with the corresponding lumping regions $\mathcal{L}_l$ (left), mesh parameters (right).*

3. Geometry and notations. In view of the finite element discretization of problem (2.5), we introduce on the domain Ω a triangulation $\mathcal{T}_h$ and denote by T the generic triangle of $\mathcal{T}_h$, by h_T the diameter of T, and by h the maximum of h_T over $\mathcal{T}_h$.

Definition. $\mathcal{T}_h$ is a Delaunay triangulation if, for every $T \in \mathcal{T}_h$, the circumcircle of T contains no other vertices than those belonging to T (cf. [19]).

We assume henceforth that $\mathcal{T}_h$ is a Delaunay triangulation. Moreover, we shall indicate by N_h and M_h the total number of edges and triangles of $\mathcal{T}_h$, respectively. Throughout the article, any geometrical entity will always be understood as being an open bounded subset of $\mathbb{R}^d$. For any couple of neighboring triangles T_i and T_j of $\mathcal{T}_h$, let l_{ij} be the common edge, and let λ_{ij} be its corresponding numbering, with $1 \le \lambda_{ij} \le N_h$ for any $i, j = 1, \dots, M_h$. For brevity, we will write in this section l instead of l_{ij}.

We also consider the dual tessellation $\mathcal{L}_h$ of $\mathcal{T}_h$ and denote its elements by "lumping regions" (see Figure 1, left). The lumping region $\mathcal{L}_l$ corresponding to edge l is obtained by joining the common vertices and the two circumcenters C_i and C_j (see Figure 1, left). We also set $\mathcal{L}_{l,T_k} = \mathcal{L}_l \cap T_k$ for $k = i,\ j$. From now on we assume the following regularity constraint on $\mathcal{T}_h$:

$$\frac{D_T}{\rho_T} \le \mathcal{K} \qquad \forall T \in \mathcal{T}_h, \tag{3.1}$$

where D_T and ρ_T are the diameters of the circumscribed and inscribed circles of triangle T, respectively, and $\mathcal{K}$ is a suitable constant (see Figure 1, right). Inequality (3.1) immediately implies that

$$D_T \le \mathcal{K}\rho_T \le \mathcal{K}h_T \qquad \forall T \in \mathcal{T}_h, \tag{3.2}$$

from which it follows that, for an arbitrary triangulation $\mathcal{T}_h$,

$$h_{\mathcal{L}_l} \le D \le \mathcal{K}h \qquad \forall \mathcal{L}_l \in \mathcal{L}_h,$$

where $D = \max_T D_T$ (see Figure 2).

4. RT finite element discretization. Let us consider the Galerkin approximation of (2.5). For any integer $k \ge 0$ we denote by $\mathbb{P}_k$ the space of polynomials of degree $\le k$ in the space variables x, y and define for $k \ge 1$ the space of vector

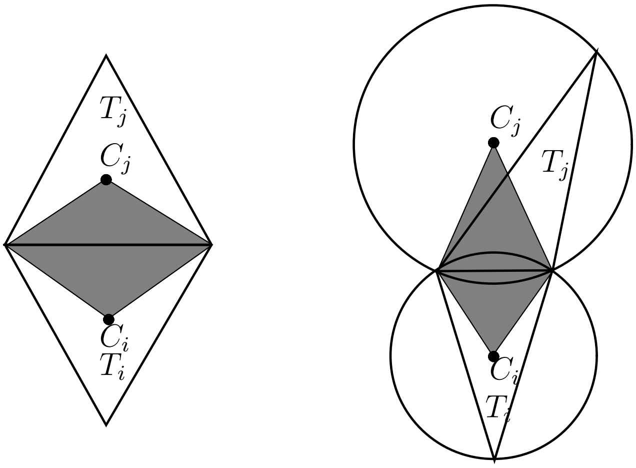


FIG. 2. *Geometrical dimensions (left) and a lumping region for an obtuse triangle.*

polynomials $\mathbb{D}_k = (\mathbb{P}_{k-1})^2 \oplus \underline{x}\mathbb{P}_{k-1}$, where $\underline{x} = (x, y)^T$. Then we introduce the RT finite element space of lowest order [35]

$$V_h = \{\underline{\tau}_h \in V \mid \underline{\tau}_h|_T \in \mathbb{D}_1, \forall T \in \mathcal{T}_h\},$$
$$Q_h = \{v_h \in Q \mid v_h|_T \in \mathbb{P}_0, \forall T \in \mathcal{T}_h\}. \tag{4.1}$$

The finite element discretization of (2.5) reads

$$\begin{cases} \text{Find } \underline{J}_h \in V_h \text{ and } u_h \in Q_h \text{ such that} \\ a(\underline{J}_h, \underline{\tau}_h) + b(\underline{\tau}_h, u_h) = 0 & \forall \underline{\tau}_h \in V_h, \\ -b(\underline{J}_h, v_h) + c(u_h, v_h) = (f, v_h)_{0,\Omega} & \forall v_h \in Q_h. \end{cases} \tag{4.2}$$

For any $\underline{\psi}_h \in V_h$ and $q_h \in Q_h$, we associate with $b(\underline{\psi}_h, q_h)$ an operator B_h from V_h into Q'_h (see [12, p. 51] for more details) and identify its kernel as the subspace $\mathcal{V}_h$ of V_h given by

$$\mathrm{Ker}(B_h) \equiv \mathcal{V}_h = \{\underline{\psi}_h \in V_h \mid b(\underline{\psi}_h, q_h) = 0 \quad \forall q_h \in Q_h\}. \tag{4.3}$$

Existence and uniqueness of the solution $(\underline{J}_h, u_h)$ of (4.2) are ensured by the fact that $a(\cdot,\cdot)$ is coercive on $\mathrm{Ker}(B_h)$ and positive-semidefinite on V_h (i.e., $a(\underline{\psi}_h, \underline{\psi}_h) \geq 0$ for all $\underline{\psi}_h \in V_h$), $c(\cdot,\cdot)$ is a continuous, symmetric, and positive-semidefinite bilinear form on $Q_h \times Q_h$ and that the spaces V_h and Q_h satisfy the inf-sup condition (see [12, Prop. 2.11] and [36, Thm. 10.4 and Remark 10.8, pp. 572-573]).

From previous notation it turns out that $N_h = \dim V_h$, $M_h = \dim Q_h$; let us introduce a basis on these spaces, namely, $\{\underline{\tau}_i\}_{i=1}^{N_h}$ for V_h and $\{v_i\}_{i=1}^{M_h}$ for Q_h. Next, we denote

$$a_{ij} = a(\underline{\tau}_j, \underline{\tau}_i), \qquad b_{ij} = b(\underline{\tau}_j, v_i), \qquad c_{ij} = c(v_j, v_i), \qquad f_i = (f, v_i)$$

and define the matrices $A = (a_{ij}) \in \mathbb{R}^{N_h \times N_h}$, $B = (b_{ij}) \in \mathbb{R}^{M_h \times N_h}$, $C = (c_{ij}) \in \mathbb{R}^{M_h \times M_h}$ and vectors $\mathbf{f} = (f_i) \in \mathbb{R}^{M_h}$, $\mathbf{J} = \{\Phi_i\} \in \mathbb{R}^{N_h}$, $\mathbf{u} = \{u_i\} \in \mathbb{R}^{M_h}$, where Φ_i and u_i are the degrees of freedom for $\underline{J}_h$ and u_h (the fluxes of $\underline{J}_h$ across each edge of

$\mathcal{T}_h$ and the values of u_h over each element $T \in \mathcal{T}_h$, respectively). Eliminating $\mathbf{J}$ from the first equation in (1.1), we end up with the following linear system on u_h:

$$(C + BA^{-1}B^T)\mathbf{u} = \mathbf{f}. \tag{4.4}$$

Solving (4.4) can be quite expensive since the matrix $W \equiv C + BA^{-1}B^T$ is full (due to the presence of A^{-1}); moreover, W is symmetric and positive-definite, but it is not in general an M-matrix (see [12, Remark 5.4, p. 195] and [32, 26]).

To circumvent these two difficulties, we construct in the next section, starting from (4.2), a family of cell-centered finite volume schemes characterized by a reduced computational cost and the same accuracy as the original mixed formulation.

5. A family of finite volume methods. For any $T_k \in \mathcal{T}_h$ and for any integrable function φ, define its mean value as

$$\varphi_k = \frac{(\varphi, 1)_{0,T_k}}{|T_k|} \qquad \forall T_k \in \mathcal{T}_h, \quad \varphi \in L^1(T_k). \tag{5.1}$$

Taking v_h equal to the characteristic function $\mathcal{X}_k$ of triangle T_k in the second equation of (4.2), we obtain

$$- \sum_{j \in \mathrm{adj}(T_k)} \Phi_{\lambda_{kj}} + u_k \sigma_k |T_k| = f_k |T_k| \qquad \forall T_k \in \mathcal{T}_h, \tag{5.2}$$

where $\mathrm{adj}(T_k)$ is the index set of the elements surrounding triangle T_k. Clearly, (5.2) is a genuine finite volume discretization for (2.1), provided that we are able to express each flux $\Phi_{\lambda_{kj}}$, $j \in \mathrm{adj}(T_k)$, in terms of the nodal values u_k and u_j only. With this aim, let us consider the first equation in (4.2) and denote by l_{ij} an edge of $\mathcal{T}_h$ and by T_i, T_j the couple of elements of $\mathcal{T}_h$ such that $\partial T_i \cap \partial T_j = l_{ij}$ (see Figure 2, left). Let also $\underline{n}_{l_{ij}}$ be the unit outward normal vector to ∂T_i (i.e., $\underline{n}_{l_{ij}}$ is positively oriented from T_i towards T_j).

Taking $\underline{\tau}_h = \underline{\tau}_{\lambda_{ij}}$ in the first equation in (4.2) and noting that $\mathrm{div}\, \underline{\tau}_{\lambda_{ij}}|_{T_i} = 1/|T_i|$ and $\mathrm{div}\, \underline{\tau}_{\lambda_{ij}}|_{T_j} = -1/|T_j|$, we get

$$\int_{T_i} \alpha \underline{J}_h \cdot \underline{\tau}_{\lambda_{ij}}\, dT + \int_{T_j} \alpha \underline{J}_h \cdot \underline{\tau}_{\lambda_{ij}}\, dT + u_i - u_j = 0. \tag{5.3}$$

For any $T_k \in \mathcal{T}_h$ let $i, j \in \{1, 2, 3\}$ be the local numbering of the edges of triangle T_k; we then introduce the following exact and approximate bilinear forms restricted over T_k:

$$\begin{aligned} a^{T_k}(\underline{\tau}_j, \underline{\tau}_i) &= \left(\alpha \underline{\tau}_j, \underline{\tau}_i\right)_{0,T_k} = \int_{T_k} \alpha \underline{\tau}_j \cdot \underline{\tau}_i\, dT, \\ a_h^{T_k}(\underline{\tau}_j, \underline{\tau}_i) &= \overline{\alpha}_{l_i} \left(\underline{\tau}_j, \underline{\tau}_i\right)_{h,0,T_k} = \delta_{ij} \frac{d_{k,i}}{|l_i|}, \end{aligned} \tag{5.4}$$

where δ_{ij} is the usual Kronecker symbol, $d_{k,i}$ is the distance between C_k and the edge l_i, and $\overline{\alpha}_{l_i}$ is a suitable average of α over $\mathcal{L}_{l_i}$.

Remark 5.1. *For each edge l_i the distance $d_{k,i}$ is computed using the formula*

$$d_{k,i} = \left(\underline{t}_i \times (\underline{x}_{C_k} - \underline{x}_{1,i})\right) \cdot \underline{n}_z, \qquad i = 1, 2, 3, \tag{5.5}$$

where the symbol "×" denotes the vector product, $\underline{n}_z$ is the unit vector orthogonal to the (x, y) plane, $\underline{x}_{C_k}$ is the coordinate vector of C_k, $\underline{x}_{1,i}$ is the coordinate vector of

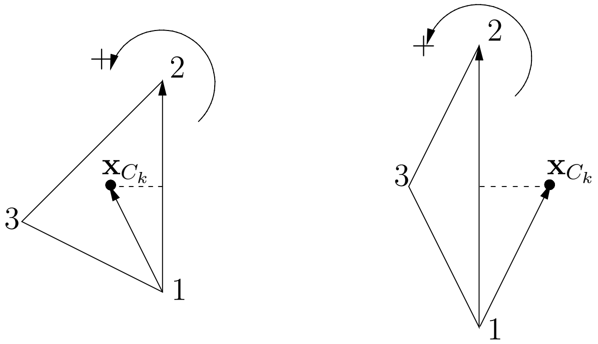


FIG. 3. *Distance of a triangle edge from the circumcenter. Acute triangle (left) and obtuse triangle (right).*

the first vertex of the edge, and $\underline{t}_i$ *is the unit tangent vector to edge* l_i *according to a counterclockwise orientation. From* (5.5) *it follows that the distance* $d_{k,i}$ *has actually a sign, being positive when the edge* l_i *is opposite to an acute angle (Figure* 3*, left) and negative when the angle is obtuse (Figure* 3*, right).*

The construction of the piecewise constant function $\overline{\alpha}$ addressed in the introduction will be fully discussed in sections 7.2 and 8; here we just emphasize the two basic properties of the average: (i) $\overline{\alpha}$ is constant on each lumping region; (ii) $\overline{\alpha}$ is some average of the restriction of α to a suitable subset of each lumping region.

We notice that in the case when $\alpha = 1$, the approximate bilinear form a_h^T coincides with the quadrature formula proposed in [9], where it is shown that the quadrature error is $\mathcal{O}(h_T)$. We let $(\underline{J}_h^*, u_h^*) \in (V_h \times Q_h)$ be henceforth the couple of discrete functions that are solutions of the dual mixed system (4.2) in presence of some quadrature error.

Using (5.4) in (5.3) and recalling that $\overline{\alpha}_{l_{ij}}$ is constant over *all* the lumping region $\mathcal{L}_{l_{ij}}$, we end up with the following equation for the approximate interelement flux $\Phi^*_{h,\lambda_{ij}}$:

$$\Phi^*_{h,\lambda_{ij}} = \overline{\alpha}_{l_{ij}}^{-1} \left(\frac{u_j^* - u_i^*}{d_{ij}} \right) |l_{ij}|, \qquad d_{ij} = d_{i,\lambda_{ij}} + d_{j,\lambda_{ij}}. \tag{5.6}$$

Equation (5.6) holds for any edge l_{ij} in the interior of Ω; if l_{ij} lies on the boundary $\partial\Omega$, it is understood that the value u_j^* is set equal to the average of the Dirichlet datum for u over l_{ij} and that d_{ij} is the distance between the circumcenter of T_i and l_{ij} (see [11]).

Substituting the exact fluxes $\Phi_{\lambda_{kj}}$ in (5.2) with the corresponding approximations (5.6), we finally obtain the family of cell-centered finite volume schemes in the new unknown u_h^*:

$$\begin{cases} \displaystyle\sum_{j \in \mathrm{adj}(T_k)} \overline{\alpha}_{l_{kj}}^{-1} \left(\frac{u_k^* - u_j^*}{d_{kj}} \right) |l_{kj}| + u_k^* \sigma_k |T_k| = f_k |T_k| & \forall T_k \in \mathcal{T}_h, \\ u_j^* = 0 & \forall l_{kj} \in \partial\Omega. \end{cases} \tag{5.7}$$

The set of linear algebraic equations (5.7) can be written in matrix form as

$$W^* \mathbf{u}^* = \mathbf{f}^*, \tag{5.8}$$

where the ith component of $\mathbf{f}^*$ is $f_i|T_i|$ and

$$W^*_{ij} = \begin{cases} \displaystyle\sum_{k\in \mathrm{adj}(T_i)} \overline{\alpha}^{-1}_{l_{ik}} \frac{|l_{ik}|}{d_{ik}} + \sigma_i|T_i| & \text{if } i = j, \\ \\ -\overline{\alpha}^{-1}_{l_{ij}} \dfrac{|l_{ij}|}{d_{ij}} & \text{if } i \neq j \end{cases} \tag{5.9}$$

for $i = 1, \dots, M_h$ and $j \in \mathrm{adj}(T_i)$.

Lemma 5.2. *Assume that $\mathcal{T}_h$ is a Delaunay triangulation. Then the matrix W^* in* (5.8) *is a symmetric, positive-definite, and irreducibly diagonally dominant M-matrix.*

Proof. We first notice that the Delaunay property for $\mathcal{T}_h$ implies that the quantity d_{ij} in (5.7) is positive. As a consequence, $W^*_{ii} > 0$ and $W^*_{ij} \leq 0$ in (5.9) for $i = 1, \dots, M_h$ and $j \in \mathrm{adj}(T_i)$. The expressions (5.9) show that W^* is a symmetric matrix with at most four nonzero entries on each row. Let s_i denote the sum of the entries of each row i of W^*; then

$$s_i = \begin{cases} \sigma_i|T_i| & \text{if } \partial T_i \cap \partial\Omega = \emptyset, \\ \\ \displaystyle\sum_{\substack{j\in \mathrm{adj}(T_i) \\ l_{ij}\notin \partial\Omega}} \overline{\alpha}^{-1}_{l_{ij}} \frac{|l_{ij}|}{d_{ij}} + \sigma_i|T_i| & \text{if } \partial T_i \cap \partial\Omega \neq \emptyset. \end{cases}$$

We see that the sum of the entries of the rows corresponding to boundary triangles is strictly positive, while it is nonnegative for the rows corresponding to internal triangles. Thus W^* is a symmetric, positive-definite, irreducibly diagonally dominant M-matrix (see [43, Corollary 1, p. 85]). □

Remark 5.3. *The M-matrix property ensures that system* (5.8) *is uniquely solvable and that the family of finite volume schemes* (5.7) *enjoys a discrete maximum principle (see* [17, 37]*). In particular, provided $f \geq 0$, the discrete solution u^*_h turns out to be nonnegative.*

The M-matrix property is very desirable in applications. As an example, consider the case where (2.1) is the linearized current continuity equation for semiconductor device simulation using the drift-diffusion or energy-balance transport models (see [13, 26, 39, 29]). In such a case, the unknown u (after a suitable scaling) has the physical meaning of a concentration, and therefore it must be a nonnegative function.

Notice that the M-matrix property guarantees that the solution is oscillation-free even if the ratio $\|\sigma\|_{\infty,\Omega}/\mu_0$ is arbitrarily large. This issue is also the case with a Galerkin finite element method, where it is well known that the performance of the method can be enhanced by lumping the mass matrix associated with the zero-order term.

Remark 5.4 (extension to the three-dimensional case). *Let us briefly comment about the extension of the finite volume formulation of the present paper to the case where Ω is a polygonal domain in $\mathbb{R}^3$.*

The extension is straightforward if Ω is of the form $\Omega = \Omega_{xy} \times (0, H)$ with $\Omega_{xy} \subset \mathbb{R}^2$ and $H > 0$, and $\mathcal{T}_h$ is a partition of Ω into prisms with three vertical edges parallel to the z-axis and two horizontal faces in the (x, y)-plane. We refer to [3] *for the definition and to* [30] *for the use of mixed approximations with RT finite elements on prismatic triangulations.*

The extension of the mixed finite volume method proposed in this paper to the case where $\mathcal{T}_h$ is a decomposition of Ω into simplices is less straightforward. In particular, *if the finite element mesh is made of tetrahedra, a condition for devising a consistent diagonalization of the mass matrix has been provided in* [10, sect. 4] *in the case of the Laplace operator. This condition is necessary and sufficient for a mesh element to be an orthocentered tetrahedron and, in particular, is satisfied if regular tetrahedra are employed in the discretization. Under such an assumption, the present mixed finite volume scheme can also be formulated in three dimensions.*

In the case of a general three-dimensional triangulation the method of the present paper, as well as the related approaches proposed in [9, 10, 2]*, cannot be extended, as pointed out in* [42].

In the following sections we are going to

(i) analyze the convergence in the $H(\mathrm{div};\Omega) \times L^2(\Omega)$-norm of the finite volume methods (5.7); and

(ii) consider a special choice of the average $\overline{\alpha}$ which extends to the two-dimensional case the generalized finite element methods proposed in [7].

6. Abstract theory of the saddle-point problem. In this section we set up an abstract framework for the error analysis of Galerkin methods with numerical quadratures to approximate saddle-point problems of the form

$$(6.1)\qquad \begin{cases} \text{Find } \underline{J} \in V \text{ and } u \in Q \text{ such that} \\ a(\underline{J}, \underline{\tau}) + b(\underline{\tau}, u) = \langle g, \underline{\tau} \rangle & \forall \underline{\tau} \in V, \\ -b(\underline{J}, q) + c(u, q) = \langle f, q \rangle & \forall q \in Q. \end{cases}$$

In (6.1), V and Q are Hilbert spaces with scalar products $(\cdot,\cdot)_V$ and $(\cdot,\cdot)_Q$, the norms $\|\cdot\|_V$ and $\|\cdot\|_Q$, $g \in V'$, $f \in Q'$ are given, and $\langle \cdot,\cdot \rangle$ denotes the duality pairing between V' and V or Q' and Q. The bilinear forms $a(\cdot,\cdot)$, $b(\cdot,\cdot)$, and $c(\cdot,\cdot)$ are assumed to fulfill all the requirements for (6.1) to admit a unique solution (see [12, Ch. 2, Thm. 1.2 and Prop. 2.11]).

In particular, defining the linear operator $B : V \to Q'$ by $\langle B\underline{\lambda}, q \rangle_{Q' \times Q} = b(\underline{\lambda}, q)$ for every $\underline{\lambda} \in V$, we assume that $a(\cdot,\cdot)$ is positive-semidefinite over V and coercive over $\mathrm{Ker}(B)$, $b(\cdot,\cdot)$ satisfies the inf-sup condition, and $c(\cdot,\cdot)$ is symmetric and positive-semidefinite over Q. Problem (2.5) is clearly a special instance of (6.1) provided that $g = 0$, $\langle f, q \rangle = (f, q)_{0,\Omega}$, and the forms $a(\cdot,\cdot)$, $b(\cdot,\cdot)$, and $c(\cdot,\cdot)$ are defined as in (2.4).

Concerning the approximation of (6.1) with the Galerkin method, error estimates and analysis can be found in [12, sect. II.2.4] and [36, Ch. 3, sect. 11]. Hereafter, we shall generalize these results to the case where the bilinear forms $a(\cdot,\cdot)$, $b(\cdot,\cdot)$, and $c(\cdot,\cdot)$ and the duality brackets $\langle \cdot,\cdot \rangle$ are replaced by suitable approximations $a_h(\cdot,\cdot)$, $b_h(\cdot,\cdot)$, $c_h(\cdot,\cdot)$, and $\langle \cdot,\cdot \rangle_h$ through the use of numerical quadratures. This leads us to considering the following Galerkin problem with numerical quadrature associated with (6.1):

$$(6.2)\qquad \begin{cases} \text{Find } \underline{J}_h \in V_h \text{ and } u_h \in Q_h \text{ such that} \\ a_h(\underline{J}_h, \underline{\tau}_h) + b_h(\underline{\tau}_h, u_h) = \langle g, \underline{\tau}_h \rangle_h & \forall \underline{\tau}_h \in V_h, \\ -b_h(\underline{J}_h, q_h) + c_h(u_h, q_h) = \langle f, q_h \rangle_h & \forall q_h \in Q_h. \end{cases}$$

The family of finite volume schemes (5.7) can be recovered setting into (6.2) $g = 0$, $b_h(\cdot,\cdot) = b(\cdot,\cdot)$, and

$$\begin{aligned} a_h(\underline{J}_h, \underline{\tau}_h) &= \sum_{T_k \in \mathcal{T}_h} a_h^{T_k}(\underline{J}_h, \underline{\tau}_h), \\ c_h(u_h, v_h) &= \sum_{T_k \in \mathcal{T}_h} \sigma_k \, (u_h, v_h)_{0,T_k}\,, \\ \langle f, q_h \rangle_h &= \sum_{T_k \in \mathcal{T}_h} f_k \, (q_h, 1)_{0,T_k}\,. \end{aligned} \tag{6.3}$$

(See section 5 for the definitions of all the discrete forms.)

As previously done in section 4, we associate with $b_h(\cdot,\cdot)$ an operator $\widehat{B}_h : V_h \to Q'_h$ whose kernel is the subspace $\widehat{\mathcal{V}}_h$ of V_h given by

$$\operatorname{Ker}(\widehat{B}_h) \equiv \widehat{\mathcal{V}}_h = \{ \underline{\tau}_h \in V_h \ \mid b_h(\underline{\tau}_h, q_h) = 0 \quad \forall q_h \in Q_h \} \,. \tag{6.4}$$

Moreover, following [12, pp. 46–47] and denoting by $\widehat{B}_h^t$ the transpose of $\widehat{B}_h$, we consider for every $\varepsilon > 0$ and for every $\bar{p}_h \in (\operatorname{Ker}(\widehat{B}_h^t))^\perp$ the equation in the unknown $p_{0h} \in \operatorname{Ker}(\widehat{B}_h^t)$,

$$\varepsilon \, (p_{0h}, q_h)_Q + c_h(p_{0h}, q_h) = -c(\bar{p}_h, q_h) \qquad \forall q_h \in \operatorname{Ker}(\widehat{B}_h^t), \tag{6.5}$$

and we assume henceforth that there exists $\gamma_0 > 0$ such that

$$\gamma_0 \|p_{0h}\|_Q \le \|\bar{p}_h\|_Q \qquad \forall \bar{p}_h \in \left(\operatorname{Ker}(\widehat{B}_h^t) \right)^\perp . \tag{6.6}$$

The next proposition ensures the existence and uniqueness of the solution to problem (6.2). It makes the counterpart to the case of numerical integration of the corresponding result given in section 4 and extends to the case $\sigma \ge 0$ the result in [12, Proposition 2.15].

PROPOSITION 6.1. *Assume that there exist positive constant $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ such that*

$$\begin{aligned} &|a_h(\underline{\psi}_h, \underline{\tau}_h)| \le \mathcal{A} \|\underline{\psi}_h\|_V \|\underline{\tau}_h\|_V && \forall \underline{\psi}_h,\, \underline{\tau}_h \in V_h, \\ &|b_h(\underline{\psi}_h, q_h)| \le \mathcal{B} \|\underline{\psi}_h\|_V \|q_h\|_Q && \forall \underline{\psi}_h \in V_h, \quad q_h \in Q_h, \\ &|c_h(q_h, r_h)| \le \mathcal{C} \|q_h\|_Q \|r_h\|_Q && \forall q_h,\, r_h \in Q_h. \end{aligned} \tag{6.7}$$

Suppose also that $a_h(\cdot,\cdot)$ is coercive on $\operatorname{Ker}(\widehat{B}_h)$*, that is,*

$$a_h(\underline{\lambda}_h, \underline{\lambda}_h) \ge a_0 \|\underline{\lambda}_h\|_{0,\Omega}^2 \qquad \forall \underline{\lambda}_h \in \widehat{\mathcal{V}}_h, \tag{6.8}$$

that b_h satisfies the inf-sup condition independently of h, i.e., there exists $k_0 > 0$ such that

$$\sup_{\underline{\tau}_h \in V_h} \frac{b_h(\underline{\tau}_h, q_h)}{\|\underline{\tau}_h\|_V} \ge k_0 \|q_h\|_Q, \tag{6.9}$$

and that $c_h(\cdot,\cdot)$ is a symmetric positive-semidefinite bilinear form on $Q_h \times Q_h$ satisfying (6.6)*. Then, problem* (6.2) *has a unique solution and there exists a positive constant $\mathcal{M}$, independent of h and depending only on $\mathcal{A}$, $\mathcal{C}$, $1/a_0$, $1/k_0$, and $1/\gamma_0$ such that*

$$\|\underline{J}_h\|_V + \|u_h\|_Q \le \mathcal{M} \left(\|g\|_h + \|f\|_h \right), \tag{6.10}$$

where

$$\|g\|_h = \sup_{\underline{\tau}_h \in V_h} \frac{\langle g, \underline{\tau}_h \rangle_h}{\|\underline{\tau}_h\|_V}, \qquad \|f\|_h = \sup_{q_h \in Q_h} \frac{\langle f, q_h \rangle_h}{\|q_h\|_Q}. \tag{6.11}$$

We shall check later on conditions $(6.7)_1$ and (6.8). We merely notice that in the problem at hand $(6.7)_2$ trivially holds taking $\mathcal{B} = 1$ and that (6.9) is automatically satisfied by the RT finite element space. Moreover, using the Cauchy–Schwarz inequality, it is immediate to check that $(6.7)_3$ holds taking $\mathcal{C} = \|\sigma\|_{\infty,\Omega}$. Finally, condition (6.6) trivially holds since $\mathrm{Ker}(\widehat{B}_h^t) = \mathrm{Ker}(B_h^t) = \{\emptyset\}$.

We are in position to state the first main result of the section, which extends to the case $\sigma \geq 0$ [12, Proposition 2.16] and [36, Thm. 11.2].

THEOREM 6.2. *Let* $(\underline{J}, u)$ *be the solution of problem* (2.5)*, and let* $(\underline{J}_h, u_h)$ *be the solution of problem* (6.2)*. Assume that the hypotheses of Proposition* 6.1 *are satisfied; then the following a priori error bound holds:*

$$\begin{aligned}
&\|\underline{J} - \underline{J}_h\|_V + \|u - u_h\|_Q \leq (1 + \mathcal{M}(\mathcal{A} + \mathcal{B})) \inf_{\underline{w}_h \in V_h} \|\underline{J} - \underline{w}_h\|_V \\
&+ (1 + \mathcal{M}(\mathcal{B} + \mathcal{C})) \inf_{r_h \in Q_h} \|u - r_h\|_Q \\
&+ \mathcal{M} \sup_{\underline{v}_h \in V_h} \left[\frac{|a(\underline{J}, \underline{v}_h) - a_h(\underline{J}, \underline{v}_h)|}{\|\underline{v}_h\|_V} + \frac{|b(\underline{v}_h, u) - b_h(\underline{v}_h, u)|}{\|\underline{v}_h\|_V} \right] \\
&+ \mathcal{M} \sup_{q_h \in Q_h} \left[\frac{|b(\underline{J}, q_h) - b_h(\underline{J}, q_h)|}{\|q_h\|_Q} + \frac{|c(u, q_h) - c_h(u, q_h)|}{\|q_h\|_Q} \right] \\
&+ \mathcal{M} \left[\sup_{\underline{v}_h \in V_h} \frac{|\langle g, \underline{v}_h \rangle - \langle g, \underline{v}_h \rangle_h|}{\|\underline{v}_h\|_V} + \sup_{q_h \in Q_h} \frac{|\langle f, q_h \rangle - \langle f, q_h \rangle_h|}{\|q_h\|_Q} \right].
\end{aligned} \tag{6.12}$$

Proof. It is a nontrivial extension of the proof of [12, Proposition 2.11] which holds in the case of exact integration. For *any* couple of functions $(\tilde{\underline{J}}_h, \tilde{u}_h) \in (V_h \times Q_h)$, let us consider the following problem, which is completely equivalent to problem (6.2):

$$\left\{ \begin{array}{rcl}
a_h(\tilde{\underline{J}}_h - \underline{J}_h, \underline{\tau}_h) + b_h(\underline{\tau}_h, \tilde{u}_h - u_h) & = & a_h(\tilde{\underline{J}}_h, \underline{\tau}_h) + b_h(\underline{\tau}_h, \tilde{u}_h) - \langle g, \underline{\tau}_h \rangle \\
 & + & \langle g, \underline{\tau}_h \rangle - \langle g, \underline{\tau}_h \rangle_h \qquad \forall \underline{\tau}_h \in V_h, \\
-b_h(\tilde{\underline{J}}_h - \underline{J}_h, q_h) + c_h(\tilde{u}_h - u_h, q_h) & = & -b_h(\tilde{\underline{J}}_h, q_h) + c_h(\tilde{u}_h, q_h) - \langle f, q_h \rangle \\
 & + & \langle f, q_h \rangle - \langle f, q_h \rangle_h \qquad \forall q_h \in Q_h.
\end{array} \right.$$

Therefore, it turns out that the couple $(\tilde{\underline{J}}_h - \underline{J}_h, \tilde{u}_h - u_h) \in (V_h \times Q_h)$ is the solution of a problem in the form (6.2) with right-hand side $G = G_1 + G_2 \in V_h'$ and $F = F_1 + F_2 \in Q_h'$ (see [12] for a precise definition of these spaces) given by

$$\begin{aligned}
&G_1 : \underline{\tau}_h \to a_h(\tilde{\underline{J}}_h, \underline{\tau}_h) + b_h(\underline{\tau}_h, \tilde{u}_h) - \langle g, \underline{\tau}_h \rangle, \qquad && G_2 : \underline{\tau}_h \to \langle g, \underline{\tau}_h \rangle - \langle g, \underline{\tau}_h \rangle_h, \\
&F_1 : q_h \to -b_h(\tilde{\underline{J}}_h, q_h) + c_h(\tilde{u}_h, q_h) - \langle f, q_h \rangle, \qquad && F_2 : q_h \to \langle f, q_h \rangle - \langle f, q_h \rangle_h.
\end{aligned}$$

Using the stability estimate (6.10), we get

$$\begin{aligned}\|\tilde{\underline{J}}_h - \underline{J}_h\|_V + \|\tilde{u}_h - u_h\|_Q \leq \mathcal{M}\Bigg(&\sup_{\underline{v}_h \in V_h} \frac{\Big|a_h(\tilde{\underline{J}}_h, \underline{v}_h) + b_h(\underline{v}_h, \tilde{u}_h) - \langle g, \underline{v}_h\rangle\Big|}{\|\underline{v}_h\|_V} \\ + \sup_{q_h \in Q_h} \frac{\Big|-b_h(\tilde{\underline{J}}_h, q_h) + c_h(\tilde{u}_h, q_h) - \langle f, q_h\rangle\Big|}{\|q_h\|_Q} &+ \sup_{\underline{v}_h \in V_h} \frac{|\langle g, \underline{v}_h\rangle - \langle g, \underline{v}_h\rangle_h|}{\|\underline{v}_h\|_V} \\ + \sup_{q_h \in Q_h} \frac{|\langle f, q_h\rangle - \langle f, q_h\rangle_h|}{\|q_h\|_Q}\Bigg)&.\end{aligned} \tag{6.13}$$

Let us label the first two terms in the right-hand side of (6.13) by E_1, E_2; next, notice that since $V_h \subset V$ and $Q_h \subset Q$, we have $\langle g, \underline{v}_h\rangle = a(\underline{J}, \underline{v}_h) + b(\underline{v}_h, u)$ and $\langle f, q_h\rangle = -b(\underline{J}, q_h) + c(u, q_h)$. Concerning E_1, we have

$$E_1 = \sup_{\underline{v}_h \in V_h} \frac{\Big|a_h(\tilde{\underline{J}}_h, \underline{v}_h) + b_h(\underline{v}_h, \tilde{u}_h) - a(\underline{J}, \underline{v}_h) - b(\underline{v}_h, u)\Big|}{\|\underline{v}_h\|_V}. \tag{6.14}$$

Add and subtract $a_h(\underline{J}, \underline{v}_h)$ and $b_h(\underline{v}_h, u)$ in (6.14); this can be done since $a_h(\cdot,\cdot)$ and $b_h(\cdot,\cdot)$ are well defined on V and Q. Then, using the triangular and Cauchy–Schwarz inequalities and the continuity of a_h and b_h, we get

$$\begin{aligned}E_1 \leq &\sup_{\underline{v}_h \in V_h} \frac{|a(\underline{J}, \underline{v}_h) - a_h(\underline{J}, \underline{v}_h)|}{\|\underline{v}_h\|_V} + \mathcal{A}\|\underline{J} - \tilde{\underline{J}}_h\|_V + \sup_{\underline{v}_h \in V_h} \frac{|b(\underline{v}_h, u) - b_h(\underline{v}_h, u)|}{\|\underline{v}_h\|_V} \\ &+ \mathcal{B}\|u - \tilde{u}_h\|_Q.\end{aligned}$$

A completely analogous procedure on E_2 yields, adding and subtracting $b_h(\underline{J}, q_h)$ and $c_h(u, q_h)$,

$$\begin{aligned}E_2 \leq &\sup_{q_h \in Q_h} \frac{|b(\underline{J}, q_h) - b_h(\underline{J}, q_h)|}{\|q_h\|_Q} + \mathcal{B}\|\underline{J} - \tilde{\underline{J}}_h\|_V + \sup_{q_h \in Q_h} \frac{|c(u, q_h) - c_h(u, q_h)|}{\|q_h\|_Q} \\ &+ \mathcal{C}\|u - \tilde{u}_h\|_Q.\end{aligned}$$

Concluding, we obtain

$$\|\tilde{\underline{J}}_h - \underline{J}_h\|_V + \|\tilde{u}_h - u_h\|_Q \leq \mathcal{M}(\mathcal{A} + \mathcal{B})\|\underline{J} - \tilde{\underline{J}}_h\|_V + \mathcal{M}(\mathcal{B} + \mathcal{C})\|u - \tilde{u}_h\|_Q + \mathcal{M}\mathcal{E},$$

where $\mathcal{E}$ gathers all the other consistency error terms. Using the triangular inequality

$$\|\underline{J} - \underline{J}_h\|_V + \|u - u_h\|_Q \leq \|\underline{J} - \tilde{\underline{J}}_h\|_V + \|\tilde{\underline{J}}_h - \underline{J}_h\|_V + \|u - \tilde{u}_h\|_Q + \|\tilde{u}_h - u_h\|_Q,$$

we finally get

$$\|\underline{J} - \underline{J}_h\|_V + \|u - u_h\|_Q \leq (1 + \mathcal{M}(\mathcal{A} + \mathcal{B}))\|\underline{J} - \tilde{\underline{J}}_h\|_V + (1 + \mathcal{M}(\mathcal{B} + \mathcal{C}))\|u - \tilde{u}_h\|_Q + \mathcal{M}\mathcal{E},$$

from which, taking the infimum on $\tilde{\underline{J}}_h \in V_h$ and $\tilde{u}_h \in Q_h$, estimate (6.12) follows. □

In view of the error analysis, the following corollary can be derived from Theorem 6.2.

COROLLARY 6.3. *Under the same hypotheses as in Theorem* 6.2, *we have*

$$\begin{aligned}
&\|\underline{J}-\underline{J}_h\|_V + \|u-u_h\|_Q \le (1+\mathcal{M}(2(\mathcal{A}+\mathcal{B})+\gamma+\beta)) \inf_{\underline{w}_h\in V_h} \|\underline{J}-\underline{w}_h\|_V \\
&+ (1+\mathcal{M}(2(\mathcal{B}+\mathcal{C})+\beta+\delta)) \inf_{r_h\in Q_h} \|u-r_h\|_Q \\
&+\mathcal{M} \inf_{\underline{w}_h\in V_h} \sup_{\underline{v}_h\in V_h} \frac{|a(\underline{w}_h,\underline{v}_h)-a_h(\underline{w}_h,\underline{v}_h)|}{\|\underline{v}_h\|_V} + \mathcal{M} \inf_{r_h\in Q_h} \sup_{\underline{v}_h\in V_h} \frac{|b(\underline{v}_h,r_h)-b_h(\underline{v}_h,r_h)|}{\|\underline{v}_h\|_V} \\
&+\mathcal{M} \inf_{\underline{w}_h\in V_h} \sup_{q_h\in Q_h} \frac{|b(\underline{w}_h,q_h)-b_h(\underline{w}_h,q_h)|}{\|q_h\|_Q} + \mathcal{M} \inf_{r_h\in Q_h} \sup_{q_h\in Q_h} \frac{|c(r_h,q_h)-c_h(r_h,q_h)|}{\|q_h\|_Q} \\
&+\mathcal{M}\left[\sup_{\underline{v}_h\in V_h} \frac{|\langle g,\underline{v}_h\rangle - \langle g,\underline{v}_h\rangle_h|}{\|\underline{v}_h\|_V} + \sup_{q_h\in Q_h} \frac{|\langle f,q_h\rangle - \langle f,q_h\rangle_h|}{\|q_h\|_Q}\right],
\end{aligned}$$

where γ, β, and δ are the constants of continuity of $a(\cdot,\cdot)$, $b(\cdot,\cdot)$, and $c(\cdot,\cdot)$, respectively.

Proof. The proof follows by rewriting $(\underline{J},u)$ in the right-hand side of (6.12) as $(\underline{J}-\underline{w}_h+\underline{w}_h,\ u-r_h+r_h)$, respectively, where $\underline{w}_h$ and r_h are the same functions as those with respect to the infima in (6.12). □

We conclude the section with the error estimate that establishes the $\mathcal{O}(h)$ convergence of the family of mixed finite volume schemes (5.7).

THEOREM 6.4. *Assume that the solution $(\underline{J},u)$ of problem* (2.5) *is such that $(\underline{J},u)\in(H^1(\Omega))^2\times H^1(\Omega)$ and $\mathrm{div}\underline{J}\in H^1(\Omega)$. Then there exist two positive constants $\mathcal{C}_1$ and $\mathcal{C}_2$, depending on u, $\underline{J}$, α, σ, and f but independent of h, such that, for h sufficiently small, the solution $(\underline{J}_h,u_h)$ of problem* (6.2) *satisfies*

$$\|\underline{J}-\underline{J}_h\|_V + \|u-u_h\|_Q \le h\,(\mathcal{C}_1+\mathcal{C}_2 h)\,. \tag{6.15}$$

We emphasize that Theorem 6.4 holds irrespectively of the choice of the average $\overline{\alpha}_{l_{kj}}$ in (5.7), the only requirement being that

$$\min_{\underline{x}\in\mathcal{L}_{l_{kj}}}(\alpha(\underline{x})) \le \overline{\alpha}_{l_{kj}} \le \max_{\underline{x}\in\mathcal{L}_{l_{kj}}}(\alpha(\underline{x})) \qquad \forall\mathcal{L}_{l_{kj}}\in\mathcal{L}_h. \tag{6.16}$$

The proof of Theorem 6.4 is carried out in the next section.

7. Error analysis of the mixed finite volume methods.

To prove (6.15) we must first check the basic assumptions of Theorem 6.2 in the case of the family of mixed finite volume methods (5.7) and then give an estimate of the extra terms coming from numerical integration.

7.1. Analysis of the approximate bilinear form.

In this section we check continuity and coercivity on $\mathrm{Ker}(\widehat{B}_h)$ of the approximate bilinear form $a_h(\cdot,\cdot)$ in (6.3).

LEMMA 7.1. *There exist positive constants $\mathcal{A}$ and a_0, independent of h, such that*

$$\begin{aligned}
&|a_h(\underline{p}_h,\underline{q}_h)| \le \mathcal{A}\|\underline{p}_h\|_V\|\underline{q}_h\|_V \qquad \forall\underline{p}_h,\ \underline{q}_h\in V_h, \\
&a_h(\underline{w}_h,\underline{w}_h) \ge a_0\|\underline{w}_h\|^2_{0,\Omega}, \qquad \forall\underline{w}_h\in\widehat{\mathcal{V}}_h.
\end{aligned} \tag{7.1}$$

Proof. To prove (7.1), let us pick up a triangle $T_k\in\mathcal{T}_h$ and set $\underline{p}_h|_{T_k}=\sum_{i=1}^3 P_i\underline{\tau}_i$, $\underline{q}_h|_{T_k}=\sum_{i=1}^3 Q_i\underline{\tau}_i$, $\underline{w}_h|_{T_k}=\sum_{i=1}^3 W_i\underline{\tau}_i$, and $\gamma_{ki}=d_{k,i}/|l_i|$, where P_i, Q_i, and W_i are the fluxes of $\underline{p}_h$, $\underline{q}_h$, and $\underline{w}_h$ across edge l_i and $d_{k,i}$ is the distance between the circumcenter C_k of T_k and edge l_i (see section 5). We also let $\mathcal{R}_{T_k} = \cup_{i\,=\,1,3}\mathcal{L}_{l_i}$. Notice that here we will use a local numbering of the edges and the related quantities. Moreover, for brevity, the subscript k will be always dropped. The element approximate bilinear form $a_h^T(\cdot,\cdot)$ can be written as

$$a_h^T(\underline{p}_h, \underline{q}_h) = \sum_{i=1}^{3} \overline{\alpha}_{l_i} P_i Q_i \gamma_i.$$

Notice that, using (3.1), we have

$$\max_{i=1,3} |\gamma_i| \le \frac{D_T}{\rho_T} \le \mathcal{K},$$

which holds both for acute and obtuse triangles. Therefore, since $\overline{\alpha}$ is an average of α,

$$|a_h^T(\underline{p}_h, \underline{q}_h)| \|\alpha\|_{\infty,\mathcal{R}_T} \sum_{i=1}^{3} |P_i||Q_i| \le \mathcal{K}\|\alpha\|_{\infty,\Omega} \sum_{i=1}^{3} |P_i||Q_i|. \tag{7.2}$$

Let $\underline{x} = F_T(\widehat{\underline{x}}) = B_T\widehat{\underline{x}} + \underline{x}_0$ be the usual affine mapping between the reference unit triangle $\widehat{T}$ and T. Let also $\widehat{h} = \sqrt{2}$ be the diameter of $\widehat{T}$, and let $\mathcal{J}_T = \det(B_T)$ denote the (constant) Jacobian matrix of the transformation. Given any vector-valued function $\widehat{\underline{\psi}} \in (H^1(\widehat{T}))^2$, we associate with $\widehat{\underline{\psi}}$ the function $\underline{\psi}$ defined on T by

$$\underline{\psi} = \frac{1}{\mathcal{J}_T} B_T \widehat{\underline{\psi}} \circ F_T^{-1} \equiv \mathcal{P}\widehat{\underline{\psi}}(\widehat{\underline{x}}),$$

where $\mathcal{P}(\cdot)$ is the *Piola mapping* (see, e.g., [35, (3.17)] or, for a different definition, [12, (1.45)] and is such that

$$\int_{\widehat{l}_i} \widehat{\underline{\psi}} \cdot \widehat{\underline{n}} d\widehat{s} = \int_{l_i} \underline{\psi} \cdot \underline{n} ds, \qquad i = 1, 3, \tag{7.3}$$

as can be easily checked by taking $\widehat{\varphi} = \chi(\widehat{l}_i)$ in formula (3.19) of [35]. Then take $\widehat{\underline{\psi}} \in \mathbb{D}_1 \ \forall \widehat{\underline{x}} \in \widehat{T}$, i.e., $\widehat{\underline{\psi}} = \sum_{i=1}^{3} \widehat{\Psi}_i \widehat{\underline{\tau}}_i$; clearly, each edge flux $\widehat{\Psi}_i$ can be identified with a continuous linear functional on the space $(L^2(\widehat{T}))^2$, denoted as $L_i(\widehat{\underline{\psi}})$ and with norm

$$\|L_i\|_{\mathcal{L}(\mathbb{D}_1;\mathbb{R})} = \sup_{\substack{\widehat{\underline{\psi}} \in \mathbb{D}_1 \\ \widehat{\underline{\psi}} \neq \mathbf{0}}} \frac{\left| \int_{\widehat{l}_i} \widehat{\underline{\psi}} \cdot \widehat{\underline{n}} d\widehat{s} \right|}{\|\widehat{\underline{\psi}}\|_{0,\widehat{T}}}, \qquad i = 1, 3,$$

where, as usual, $\mathcal{L}(\mathcal{V};\mathcal{W})$ is the set of linear continuous functionals from $\mathcal{V}$ into $\mathcal{W}$. It can be checked that $\max_{i=1,3} \|L_i\|_{\mathcal{L}(\mathbb{D}_1;\mathbb{R})} = \sqrt{6}$, so that we get

$$\sum_{i=1}^{3} |\widehat{P}_i||\widehat{Q}_i| \le \sum_{i=1}^{3} \|L_i\|^2_{\mathcal{L}(\mathbb{D}_1;\mathbb{R})} \|\widehat{\underline{p}}_h\|_{0,\widehat{T}} \|\widehat{\underline{q}}_h\|_{0,\widehat{T}} \le 18 \|\widehat{\underline{p}}_h\|_{0,\widehat{T}} \|\widehat{\underline{q}}_h\|_{0,\widehat{T}}. \tag{7.4}$$

Next, we recall that for any function $\widehat{\underline{\psi}} \in (L^2(\widehat{T}))^2$ the following result holds (cf. formula (3.21) in [35] with $l = 0$):

$$\|\widehat{\underline{\psi}}\|_{0,\widehat{T}} \le \|B_T^{-1}\| |\mathcal{J}_T|^{1/2} \|\underline{\psi}\|_{0,T}, \tag{7.5}$$

where $\|\cdot\|$ is the spectral matrix norm. Using (7.3), (7.4), (7.5), standard scaling arguments (see, e.g., [34, p. 86]) and (3.1), we finally obtain

$$|a_h^T(\underline{p}_h, \underline{q}_h)| \leq 36\mathcal{K}^3\|\alpha\|_{\infty,\Omega}\|\underline{p}_h\|_{H(\mathrm{div};T)}\|\underline{q}_h\|_{H(\mathrm{div};T)}, \tag{7.6}$$

from which, summing over all the triangles of $\mathcal{T}_h$ and using the Cauchy–Schwarz inequality, the estimate $(7.1)_1$ follows with $\mathcal{A} = 36\mathcal{K}^3\|\alpha\|_{\infty,\Omega}$. The proof of $(7.1)_2$ is straightforward; for any function $\underline{w}_h$ such that $\underline{w}_h|_T \in (\mathbb{P}_0)^2$ we have

$$a_h^T(\underline{w}_h, \underline{w}_h) \geq \inf_{\underline{x}\in\mathcal{R}_T}(\alpha)\sum_{i=1}^{3} W_i^2\gamma_i \geq \alpha_0\sum_{i=1}^{3} W_i^2\gamma_i, \tag{7.7}$$

where $\alpha_0 = \inf_\Omega(\alpha)$ and W_i is the flux of $\underline{w}_h$ across edge l_i. In [9] it has been proved that $\sum_{i=1}^{3} W_i^2\gamma_i = (\underline{w}_h, \underline{w}_h)_{0,T} = \|\underline{w}_h\|_{0,T}^2$, so that, summing (7.7) over all the triangles in $\mathcal{T}_h$ and using the Cauchy–Schwarz inequality, we finally get $(7.1)_2$ with $a_0 = \alpha_0$ and $\widehat{\mathcal{V}}_h = \{\underline{w} \in (L^2(\Omega))^2 \,|\, \underline{w}|_T \in (\mathbb{P}_0)^2\ \forall T \in \mathcal{T}_h\}$. □

Recalling that the conditions in Proposition 6.1 about b_h and c_h have already been assessed, Theorem 6.2 applies for the convergence analysis of the family of generalized Galerkin methods (5.7).

7.2. Analysis of the quadrature error. In this section we deal with the quadrature errors occurring in the approximate linear and bilinear forms in the finite volume schemes (5.7). For this, we set $g = 0$ and $b_h(\cdot,\cdot) = b(\cdot,\cdot)$. Moreover, to avoid resorting to an interpolation-free approach (see [18], [36, pp. 132–133]) or to some suitable projection operator (see [34, sect. 3.5]), we require an extra regularity for the problem coefficients than asked for in section 2, i.e., assume henceforth that μ, σ, and f are given functions in $W^{1,\infty}(\Omega)$. We shall come back to the issue of coefficient regularity in Remark 7.4. Finally, throughout the section we denote by C a positive constant whose value is not necessarily the same at each occurrence.

We recall the interpolation estimate (see [36, formula (16.19)])

$$|v - \Pi v|_{m,\infty,T} \leq C\frac{h_T^{k+1}}{\rho_T^m}|v|_{k+1,\infty,T}, \tag{7.8}$$

where $v \in W^{k+1,\infty}(T)$ for $k \geq 0$, C is a positive constant, and $\Pi v : T \to \mathbb{R}$ is the local interpolation operator of v associated with the finite element space $\mathbb{P}_k$.

For any $T \in \mathcal{T}_h$, we denote by $\Pi_{0,T}$ the local $\mathbb{P}_0$-interpolation operator of σ such that $\Pi_{0,T}\sigma = \sigma(C_T)$. Using (7.8) with $m = 0$, $k = 0$ and letting C_c be a positive constant yields

$$\begin{aligned}|c(r_h, q_h) - c_h(r_h, q_h)| &\leq \sum_{T\in\mathcal{T}_h}|\sigma - \Pi_{0,T}\sigma|_{0,\infty,T}\left(\int_T |r_h||q_h|dT\right)\\ &\leq \sum_{T\in\mathcal{T}_h} C_c h_T|\sigma|_{1,\infty,T}\|r_h\|_{0,T}\|q_h\|_{0,T} \leq C_c h|\sigma|_{1,\infty,\Omega}\|r_h\|_Q\|q_h\|_Q\end{aligned}$$

so that the quadrature error for $c(\cdot,\cdot)$ can be estimated as

$$\sup_{q_h\in Q_h}\frac{|c(r_h, q_h) - c_h(r_h, q_h)|}{\|q_h\|_Q} \leq C_c h|\sigma|_{1,\infty,\Omega}\|r_h\|_Q. \tag{7.9}$$

In a similar way we obtain

$$\sup_{q_h\in Q_h}\frac{|\langle f, q_h\rangle - \langle f, q_h\rangle_h|}{\|q_h\|_Q} \leq C_f|\Omega|^{1/2}h|f|_{1,\infty,\Omega}. \tag{7.10}$$

Let us now give a bound for the quadrature error associated with $a_h(\cdot,\cdot)$. With this aim, let $\underline{p}_h$ and $\underline{q}_h$ be any two functions in V_h. Using the same notation as in section 7.1, for any $T \in \mathcal{T}_h$ we set $\underline{p}_h|_T = \sum_{i=1}^{3} P_i \underline{\tau}_i$ and $\underline{q}_h|_T = \sum_{i=1}^{3} Q_i \underline{\tau}_i$. Then we recall two approximation results that will be useful in the forthcoming analysis. The first one is an immediate consequence of Theorem 15.3 in [36] with $k = m = 0$ and $p = q = \infty$.

Lemma 7.2. *Let $\widehat{S}$ be an open subset of $\mathbb{R}^2$, and let $\widehat{\Pi}$ be a linear and continuous mapping from $W^{1,\infty}(\widehat{S})$ to $L^\infty(\widehat{S})$ which satisfies $\widehat{\Pi}\widehat{p} = \widehat{p}, \forall \widehat{p} \in \mathbb{P}_0(\widehat{S})$. For any open set S that is affine-equivalent to the set $\widehat{S}$, let Π_S be defined by $\Pi_S v = \widehat{\Pi}\widehat{v} \circ F^{-1}$ for all functions $\widehat{v} \in W^{1,\infty}(\widehat{S})$ and $v \in W^{1,\infty}(S)$ such that $v = \widehat{v} \circ F^{-1}$, where $F : \widehat{S} \mapsto S$ is an invertible affine mapping such that $F(\widehat{S}) = S$. Then there exists a constant C (depending on $\widehat{\Pi}$ and $\widehat{S}$) such that*

$$(7.11) \qquad \|v - \Pi_S v\|_{\infty,S} \le C h_S |v|_{1,\infty,S} \qquad \forall v \in W^{1,\infty}(S),$$

where $h_S = \operatorname{diam} S$.

The second result has been proved in [9]:

$$(7.12) \quad \left| \left(\underline{p}_h, \underline{q}_h\right)_{0,T} - \left(\underline{p}_h, \underline{q}_h\right)_{h,0,T} \right| \le \left(\frac{h_T}{2\sqrt{6}} + \frac{h_T^2}{48} \right) \|\underline{p}_h\|_{H(\mathrm{div};T)} \|\underline{q}_h\|_{H(\mathrm{div};T)}.$$

Let $\Pi_T \alpha = \widetilde{\alpha}_T$ be a suitable average of the function α over T satisfying Lemma 7.2. For instance, we can take $\widetilde{\alpha}_T$ equal to the value of α at the centroid of T. The quadrature error $a^T(\underline{p}_h, \underline{q}_h) - a_h^T(\underline{p}_h, \underline{q}_h)$ can then be split as

$$a^T(\underline{p}_h, \underline{q}_h) - a_h^T(\underline{p}_h, \underline{q}_h) = \mathcal{E}_1(\underline{p}_h, \underline{q}_h) + \mathcal{E}_2(\underline{p}_h, \underline{q}_h) + \mathcal{E}_3(\underline{p}_h, \underline{q}_h),$$

where

$$\mathcal{E}_1(\underline{p}_h, \underline{q}_h) = \left(\alpha \underline{p}_h, \underline{q}_h\right)_{0,T} - \left(\widetilde{\alpha}_T \underline{p}_h, \underline{q}_h\right)_{0,T} = \int_T (\alpha - \widetilde{\alpha}_T) \underline{p}_h \cdot \underline{q}_h dT,$$

$$\mathcal{E}_2(\underline{p}_h, \underline{q}_h) = \left(\widetilde{\alpha}_T \underline{p}_h, \underline{q}_h\right)_{0,T} - \left(\widetilde{\alpha}_T \underline{p}_h, \underline{q}_h\right)_{h,0,T} = \int_T \widetilde{\alpha}_T \underline{p}_h \cdot \underline{q}_h dT - \sum_{i=1}^{3} \widetilde{\alpha}_T P_i Q_i \gamma_i,$$

$$\mathcal{E}_3(\underline{p}_h, \underline{q}_h) = \left(\widetilde{\alpha}_T \underline{p}_h, \underline{q}_h\right)_{h,0,T} - a_h^T(\underline{p}_h, \underline{q}_h) = \sum_{i=1}^{3} (\widetilde{\alpha}_T - \overline{\alpha}_{l_i}) P_i Q_i \gamma_i.$$

Using (7.11) and the Cauchy–Schwarz inequality, we get

$$|\mathcal{E}_1(\underline{p}_h, \underline{q}_h)| \le C h_T |\alpha|_{1,\infty,T} \|\underline{p}_h\|_{H(\mathrm{div};T)} \|\underline{q}_h\|_{H(\mathrm{div};T)},$$

while (7.12) yields

$$|\mathcal{E}_2(\underline{p}_h, \underline{q}_h)| \le \widetilde{\alpha}_T \left(\frac{h_T}{2\sqrt{6}} + \frac{h_T^2}{48} \right) \|\underline{p}_h\|_{H(\mathrm{div};T)} \|\underline{q}_h\|_{H(\mathrm{div};T)}.$$

Recalling that $\widetilde{\alpha}_T$ and $\overline{\alpha}_{l_i}$ are averages of α over $T \subset \mathcal{R}_T$ and $\mathcal{L}_{l_i} \subset \mathcal{R}_T$, respectively, we can immediately conclude that

$$\begin{aligned} |\mathcal{E}_3(\underline{p}_h, \underline{q}_h)| &\le \left(\max_{\underline{x} \in \mathcal{R}_T} (\alpha) - \min_{\underline{x} \in \mathcal{R}_T} (\alpha) \right) \sum_{i=1}^{3} |P_i|\,|Q_i|\,|\gamma_i| \\ &\le |\alpha|_{1,\infty,\mathcal{R}_T} D_T \sum_{i=1}^{3} |P_i|\,|Q_i|\,|\gamma_i \le 36 \mathcal{K}^4 |\alpha|_{1,\infty,\mathcal{R}_T} h_T \|\underline{p}_h\|_{H(\mathrm{div};T)} \|\underline{q}_h\|_{H(\mathrm{div};T)}, \end{aligned}$$

where we have used (3.2) and (7.6) with $\alpha = 1$.

Gathering the previous estimates and overestimating the norms on local sets with the same norms over Ω yields $\forall T \in \mathcal{T}_h$

$$\begin{aligned}&|a^T(\underline{p}_h, \underline{q}_h) - a_h^T(\underline{p}_h, \underline{q}_h)|\\ &\quad \leq \left((C + 36\mathcal{K}^4)|\alpha|_{1,\infty,\Omega} + \|\alpha\|_{\infty,\Omega}\left(\frac{1}{2\sqrt{6}} + \frac{h}{48}\right)\right) h\|\underline{p}_h\|_{H(\mathrm{div};T)}\|\underline{q}_h\|_{H(\mathrm{div};T)}.\end{aligned}$$

Letting

$$\Phi = \Phi(h, \mathcal{T}_h, \alpha) = \left((C + 36\mathcal{K}^4)|\alpha|_{1,\infty,\Omega} + \|\alpha\|_{\infty,\Omega}\left(\frac{1}{2\sqrt{6}} + \frac{h}{48}\right)\right),$$

summing over all the mesh triangles, and using the Cauchy–Schwarz inequality finally gives

$$\sup_{\underline{q}_h \in V_h} \frac{|a(\underline{p}_h, \underline{q}_h) - a_h(\underline{p}_h, \underline{q}_h)|}{\|\underline{q}_h\|_V} \leq \Phi h \|\underline{p}_h\|_{H(\mathrm{div};\Omega)}. \tag{7.13}$$

Taking, in (7.9), r_h equal to the L^2-projection of u, we get

$$\sup_{q_h \in Q_h} \frac{|c(r_h, q_h) - c_h(r_h, q_h)|}{\|q_h\|_Q} \leq C_c h|\sigma|_{1,\infty,\Omega}\|u\|_Q. \tag{7.14}$$

Analogously, taking, in (7.13), $\underline{p}_h$ equal to the RT-interpolant to $\underline{J}$ of lowest degree $\pi_h\underline{J}$ and recalling that (see [36, p. 583])

$$\|\pi_h\underline{J}\|_V \leq C\left(h|\underline{J}|_{1,\Omega} + \|\mathrm{div}\underline{J}\|_{0,\Omega}\right),$$

we get

$$\sup_{\underline{q}_h \in V_h} \frac{|a(\underline{p}_h, \underline{q}_h) - a_h(\underline{p}_h, \underline{q}_h)|}{\|\underline{q}_h\|_V} \leq C\Phi h\left(h|\underline{J}|_{1,\Omega} + \|\mathrm{div}\underline{J}\|_{0,\Omega}\right). \tag{7.15}$$

From Corollary 6.3, using the estimates (7.14), (7.10), and (7.15), letting

$$M = \max\left((1 + \mathcal{M}(2(\mathcal{A} + \mathcal{B}) + \gamma + \beta)), (1 + \mathcal{M}(2(\mathcal{B} + \mathcal{C}) + \beta + \delta))\right),$$

and denoting by C_k, $k = 1, 3$, some positive constants, we finally obtain the convergence result (6.15):

$$\begin{aligned}&\|\underline{J} - \underline{J}_h\|_V + \|u - u_h\|_Q \leq Mh\left[|u|_{1,\Omega} + |\underline{J}|_{1,\Omega} + |\mathrm{div}\underline{J}|_{1,\Omega}\right]\\ &\quad + C_1\Phi h\left(h\|\underline{J}\|_{1,\Omega} + \|\mathrm{div}\underline{J}\|_{0,\Omega}\right)\\ &\quad + C_2 h|\sigma|_{1,\infty,\Omega}\|u\|_Q + C_3 h|\Omega|^{1/2}|f|_{1,\infty,\Omega}.\end{aligned} \tag{7.16}$$

REMARK 7.3. *We remark that the $\mathcal{O}(h)$ convergence proved above for the mixed finite volume methods* (5.7) *is optimal and that it has been obtained without giving up the M-matrix property of the schemes. This feature, together with the reduced computational cost, makes the novel methods quite attractive and competitive with respect to the standard dual mixed approaches with exact integration* [27, 26] *and to* *dual mixed schemes with numerical integration recently proposed for the approximation of problem* (2.1) *in the case* $\sigma = 0$ [9, 10]. *Finally, the genuine finite volume flavor of the novel methods allows for the basic conservation properties (mass and interelement fluxes) to be satisfied, even in the presence of jumps in the coefficient of the problem (as happens, for instance, in porous media flows governed by Darcy's law* [20]*) or steep internal/boundary layers in the scalar unknown* u *(as happens in semiconductor device simulation using the drift-diffusion model* [24, 28, 31, 39]*). Numerical evidences of the accuracy and stability of the new formulation will be exhibited in the next section.*

REMARK 7.4. *The error estimate* (6.15) *has been obtained under stringent regularity assumptions on the problem coefficients. This is quite a standard situation when trying to derive a convergence result (see, e.g.,* [3, sect. 7]*). However, a more careful analysis reveals that the convergence result* (6.15) *still holds under the sole assumption that the coefficients* μ, σ *and the right-hand side* f *be locally smooth functions. Actually, assuming that* $\mu|_T$, $\sigma|_T$ *and* $f|_T$ *belong to* $W^{1,\infty}(T)$ *for all* $T \in \mathcal{T}_h$ *and using the same arguments as above, one can prove the following analogue of* (7.16)*:*

$$\begin{aligned}\|\underline{J} - \underline{J}_h\|_V + \|u - u_h\|_Q \leq Mh &\left[|u|_{1,\Omega} + \left(\sum_{T\in\mathcal{T}_h} |\underline{J}|^2_{1,T} \right)^{1/2} + \left(\sum_{T\in\mathcal{T}_h} |\text{div}\underline{J}|^2_{1,T} \right)^{1/2} \right] \\ &+ C_1 \widetilde{\Phi} h \left[h \left(\sum_{T\in\mathcal{T}_h} |\underline{J}|^2_{1,T} \right)^{1/2} + \|\text{div}\underline{J}\|_{0,\Omega} \right] \\ &+ C_2 h \max_{T\in\mathcal{T}_h} |\sigma|_{1,\infty,T} \|u\|_Q + C_3 h |\Omega|^{1/2} \max_{T\in\mathcal{T}_h} |f|_{1,\infty,T},\end{aligned}$$

where

$$\widetilde{\Phi} = (C + 36\mathcal{K}^4) \max_{T\in\mathcal{T}_h} |\alpha|_{1,\infty,T} + \max_{T\in\mathcal{T}_h} \|\alpha\|_{\infty,T} \left(\frac{1}{2\sqrt{6}} + \frac{h}{48} \right).$$

8. Numerical results. In this section we are going to validate numerically the family of mixed finite volume methods introduced in the present paper. With this aim we first need to characterize the choice of the average $\overline{\alpha}$ of the inverse diffusion coefficient α. Being that this matter is one-dimensional, we restrict our attention on the affine-equivalent interval $\mathcal{I} = [0, h]$ for any $h > 0$.

The obvious candidate for $\overline{\alpha}$ is the mean value of α over $\mathcal{I}$. Since we are interested in the inverse of $\overline{\alpha}$ (see (5.7)) and $\alpha = \mu^{-1}$, it follows that

$$\overline{\alpha}^{-1} = \left(\frac{\displaystyle\int_0^h \mu^{-1}(x)dx}{h} \right)^{-1} \equiv \mathcal{H}_{\mathcal{I}}(\mu), \tag{8.1}$$

where $\mathcal{H}_{\mathcal{I}}(\mu)$ denotes the *harmonic average* of μ over the interval $\mathcal{I}$.

Use of harmonic averaging for the diffusion coefficient μ is quite natural in mixed methods (see [7, 12]) and has been proved in one dimension to provide better results than the mean value, in particular when μ exhibits sharp variations on $\mathcal{I}$ or is even discontinuous. Typical instances of such behavior are flows in porous media (see [20] and the references cited therein) or electron and hole carrier flow in a semiconductor device (see [24, 13]).

It is clear that, except in trivial cases, the evaluation of (8.1) cannot be carried out. This, of course, demands for the use of a suitable quadrature formula. With this aim, define for any function $\phi : \mathcal{I} \rightarrow \mathbb{R}^+$ the *exponential interpolant* to ϕ as

$$\Pi_e\phi(x) = \exp\left\{\Pi_1\left[\ln(\phi(x))\right]\right\} = \phi_0 \left(\frac{\phi_h}{\phi_0}\right)^{\frac{x}{h}}, \qquad x \in \mathcal{I}, \tag{8.2}$$

where $\phi_0 = \phi(0)$, $\phi_h = \phi(h)$, and $\Pi_1\phi$ is the $\mathbb{P}_1$-interpolant to ϕ. The quadrature formula approximating (8.1) can then be defined as

$$\begin{aligned}\overline{\alpha}^{-1} \simeq \left(\frac{\displaystyle\int_0^h \Pi_e\mu^{-1}(x)dx}{h}\right)^{-1} &= \frac{\ln(1/\mu_h) - \ln(1/\mu_0)}{1/\mu_h - 1/\mu_0} \\ &= \left(\frac{\ln(\mu_0) - \ln(\mu_h)}{\mu_0 - \mu_h}\right)\mu_0\mu_h,\end{aligned} \tag{8.3}$$

which clearly satisfies (6.16). Assuming $\alpha \in W^{1,\infty}(\mathcal{I})$, it is easy to prove the following bound for the interpolation error:

$$\|\alpha - \Pi_e\alpha\|_{\infty,\mathcal{I}} \leq Ch\frac{\alpha_M}{\alpha_m}|\alpha|_{1,\infty,\mathcal{I}},$$

where α_M and α_m are the maximum and the minimum values of α over $\mathcal{I}$, respectively. We remark that (8.3) is *exact* if $\mu = e^{ax+b}$, $a, b \in \mathbb{R}$, $x \in \mathcal{I}$, as typically happens in the numerical approximation of the drift-diffusion semiconductor device equations (see [8, 13, 39]).

In this latter application the mixed finite volume scheme (5.7) with the quadrature (8.3) can be regarded as a two-dimensional generalization of the classical exponentially fitted Scharfetter–Gummel method [41]. The resulting scheme can be proved to recover the *exact* solution (u, $\underline{J}$) at the nodes of the dual mesh when $\mu = e^{ax+by}$, $a, b \in \mathbb{R}$, $x, y \in \overline{\Omega}$, $\sigma = 0$, $f = 0$, and suitable Dirichlet–Neumann boundary conditions are assumed in problem (2.1) (see for the proof [40] and [44], respectively, in the case of triangles and rectangles). This nice property is a special instance of the "patch-test" (see [36, Chap. V, sect. 34] and [23]) and turns out to be a sound indication for good behavior of a numerical scheme to deal with advection-dominated flow problems, as previously remarked in [44, 38].

Our second choice for $\overline{\alpha}^{-1}$ is the harmonic average of the piecewise constant extension to μ over $\mathcal{I}$

$$\overline{\alpha}^{-1} \simeq \left(\frac{\displaystyle\int_0^{\overline{x}} \mu_0^{-1}dx + \int_{\overline{x}}^h \mu_h^{-1}dx}{h}\right)^{-1} = \frac{\mu_0\mu_h}{\mu_h\overline{x}/h + \mu_0(1 - \overline{x}/h)}, \tag{8.4}$$

where it is assumed that $\mathcal{I}$ represents the segment connecting the circumcenters of two neighboring triangles T and T' so that $\overline{x}$ is the intersection between $\mathcal{I}$ and the edge common to T and T'. The average (8.4) seems to be quite promising in the presence of discontinuities of μ.

The last choice that we consider for $\overline{\alpha}^{-1}$ employs linear interpolation for μ. This leads to the trapezoidal quadrature formula

$$\overline{\alpha}^{-1} \simeq \frac{\displaystyle\int_0^h \Pi_1\mu(x)dx}{h} = \frac{\mu_0 + \mu_h}{2}. \tag{8.5}$$

Let us now deal with the numerical experiments. We henceforth assume that in (2.1) Ω is the unit square. Two test problems with available exact solutions have been considered; in the first test case five unstructured grids of triangular elements have been employed with average mesh sizes of 2^{-k} for $k = 3, \ldots, 7$. For the second test problem, which is of one-dimensional nature, eight grids of rectangular elements have been used with mesh sizes of 2^{-k} in the x-direction for $k = 3, \ldots, 10$. When applied to this example, the mixed finite volume scheme (5.7) reduces to a five-point difference scheme with a special average of the diffusion coefficient.

Suitably accurate integration formulae have been used to evaluate the $L^2(\Omega)$ and $H(\text{div}; \Omega)$ norms of the discretization errors $u - u_h$ and $\underline{J} - \underline{J}_h$. Precisely, a symmetric formula using 10 nodes on each element of the triangular grid has been employed in the first test case, while the nine-point Simpson formula has been adopted on each element in the quadrilateral grid in the second test case. We emphasize that all the numerical results presented below have been obtained using the midpoint quadrature formula to approximate the right-hand side in (5.2). In all the reported figures the symbols have the following meaning: "*" refers to the scheme using the average (8.5), "o" refers to the use of the average (8.3), and "□" refers to use of the average (8.4).

Test case 1. This test problem basically aims at checking the theoretical convergence properties of the novel method. Therefore, we take as exact solution the smooth function

$$u(x, y) = 16xy(x - 1)(y - 1)$$

such that $\|u\|_{\infty,\Omega} = 1$. The diffusion coefficient μ is

$$\mu(x, y) = \left(\frac{\mu_a + \mu_b}{2}\right)\left(1 + \frac{\mu_b - \mu_a}{\mu_a + \mu_b}\tanh(K(y - x))\right),$$

where $\mu_b > \mu_a > 0$ and K is a suitable constant. Notice that for $\mu_b/\mu_a \gg 1$ and $K \gg 1$, μ tends to a step function with jump equal to $\mu_b - \mu_a$ across the main diagonal of the unit square.

The first set of numerical results refers to the case $\mu_a = 1$, $\mu_b = 10$, and $K = 2$. The maximum values of $|\underline{J}|$ and $\text{div}\underline{J}$ are 36.05 and 129.6, respectively. Figures 4(a) and 4(b) show the L^2-norm and the discrete L^∞-norm of the error $u - u_h$, this latter norm being evaluated as

$$\|u - u_h\|_{\infty,C} = \max_{T_k \in \mathcal{T}_h} |u(\underline{x}_{C_k}) - u_k|.$$

Linear convergence with respect to the mesh size h is clearly observed for all of the considered averages in the case of the L^2-norm, while second-order convergence is exhibited by the method when the discrete L^∞-norm is used to measure convergence. This appears to be a (nodal) superconvergence of the mixed finite schemes and is to be ascribed to the special choice of the point of measurement of the convergence (i.e., the circumcenter of the triangle) which coincides with the center of the stencil of the method.

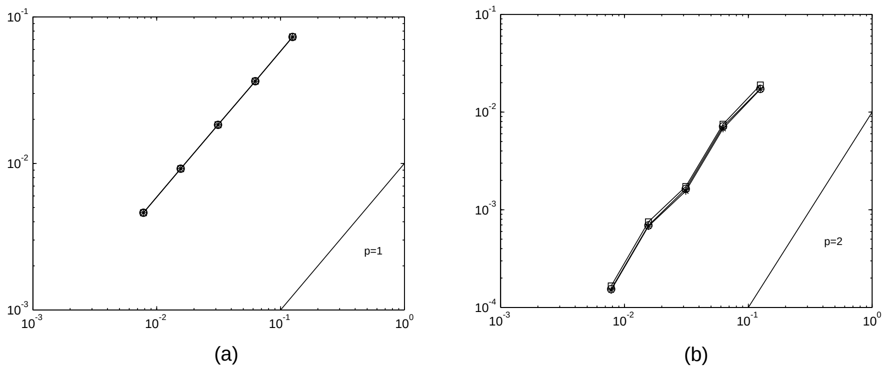


FIG. 4. *Discretization error $u - u_h$ for test case* 1: (a) *L^2-norm, $\mu_a = 1$, $\mu_b = 10$, $K = 2$;* (b) *discrete L^∞-norm, $\mu_a = 1$, $\mu_b = 10$, $K = 2$.*

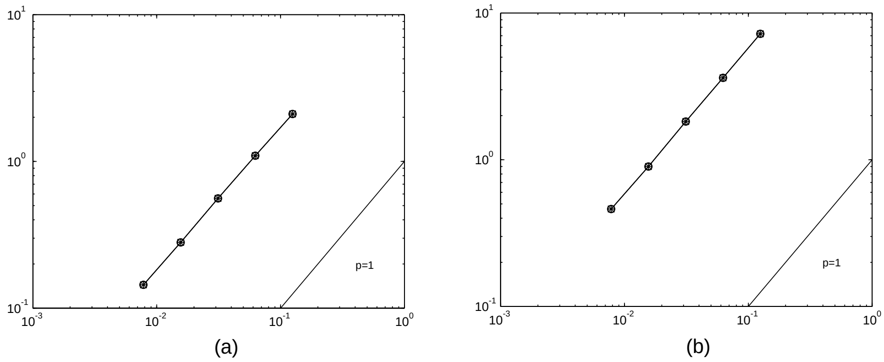


FIG. 5. *Test case* 1: (a) *L^2-norm of the discretization error $\underline{J} - \underline{J}_h$, $\mu_a = 1$, $\mu_b = 10$, $K = 2$;* (b) *graph norm of the discretization error $U - U_h = (u - u_h, \underline{J} - \underline{J}_h)$, $\mu_a = 1$, $\mu_b = 10$, $K = 2$.*

Figures 5(a) and 5(b) show the L^2-norm of the discretization error $\underline{J} - \underline{J}_h$ and the graph norm of the error $U - U_h \equiv (u - u_h, \underline{J} - \underline{J}_h)$ defined as

$$\|U - U_h\|_{V \times Q} = \left(\|u - u_h\|_{0,\Omega}^2 + \|\underline{J} - \underline{J}_h\|_{H(\mathrm{div};\Omega)}^2 \right)^{1/2}.$$

Linear convergence is clearly observed for all of the three averages.

The second set of numerical results refers to the case $\mu_a = 1$, $\mu_b = 1000$, and $K = 100$. The maximum values of $|\underline{J}|$ and $\mathrm{div}\underline{J}$ are 4000 and 16308.8, respectively (clearly much bigger than in the previous case). Figures 6(a) and 6(b) show that linear convergence in the L^2-norm is exhibited by all of the considered averages, while the schemes appear to be second-order accurate in the discrete L^∞-norm. Linear convergence is also observed for the discretization errors $\underline{J} - \underline{J}_h$ and $U - U_h$, as shown in Figures 7(a) and 7(b), respectively.

Test case 2. In the second test case, taken from [7], we check the performances of the three averages introduced in this section when the diffusion coefficient μ is discontinuous. With this aim, we consider the simple one-dimensional problem $-(\mu u')' =$

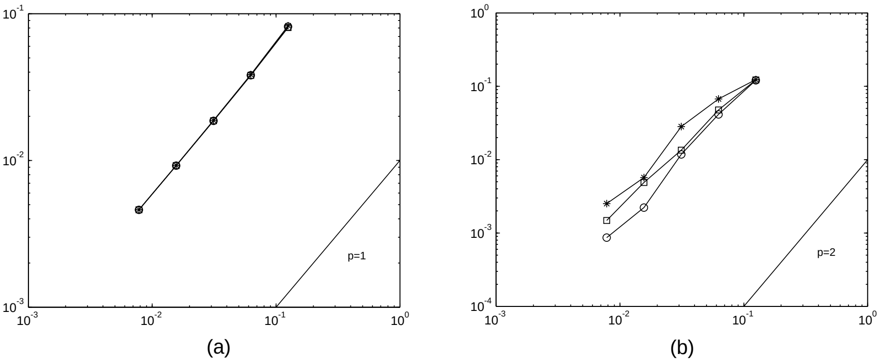


FIG. 6. *Discretization error $u - u_h$ for test case* 1: (a) *L^2-norm, $\mu_a = 1$, $\mu_b = 1000$, $K = 100$*; (b) *discrete L^∞ -norm, $\mu_a = 1$, $\mu_b = 1000$, $K = 100$.*

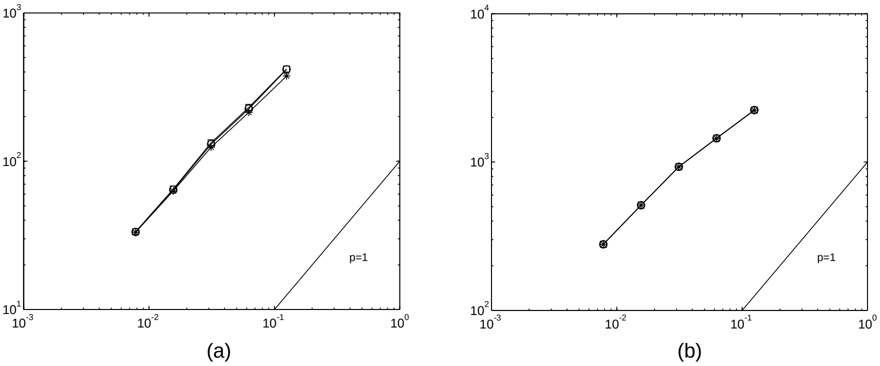


FIG. 7. *Test case* 1: (a) *L^2-norm of the discretization error $\underline{J} - \underline{J}_h$, $\mu_a = 1$, $\mu_b = 1000$, $K = 100$*; (b) *graph norm of the discretization error $U - U_h = (u - u_h, \underline{J} - \underline{J}_h)$, $\mu_a = 1$, $\mu_b = 1000$, $K = 100$.*

M^{-1} on the interval $(0, 1)$ with $u(0) = u(1) = 0$, where

$$M = \frac{\mu_a \beta^2}{2}, \quad \beta = \left(\frac{3}{4\mu_b} + \frac{1}{4\mu_a} - \gamma \right), \quad \mu(x) = \begin{cases} \mu_a, & 0 \le x < 0.5, \\ \mu_b, & 0.5 \le x \le 1, \end{cases}$$

again having $\mu_b > \mu_a > 0$ and having set $\gamma = (1 + 3\mu_a/\mu_b)/(4(\mu_a + \mu_b))$. The exact solution of this latter problem reads

$$u(x) = \frac{1}{M} \begin{cases} -\dfrac{x^2}{2\mu_a} + \beta x, & 0 \le x < 0.5, \\ -\dfrac{x^2}{2\mu_b} + \gamma x + \dfrac{1}{2\mu_b} - \gamma, & 0.5 \le x \le 1, \end{cases} \tag{8.6}$$

$$J(x) = \frac{1}{M} (\beta \mu_a - x), \quad x \in [0, 1], \tag{8.7}$$

and is such that $\|u\|_{\infty,\Omega} = 1$. Notice that the exact solution u, although continuous, does not belong to $H^2(0, 1)$.

Next we analyze a two-dimensional boundary value problem of the form (2.1) but with Dirichlet–Neumann boundary conditions, having as exact solution the function

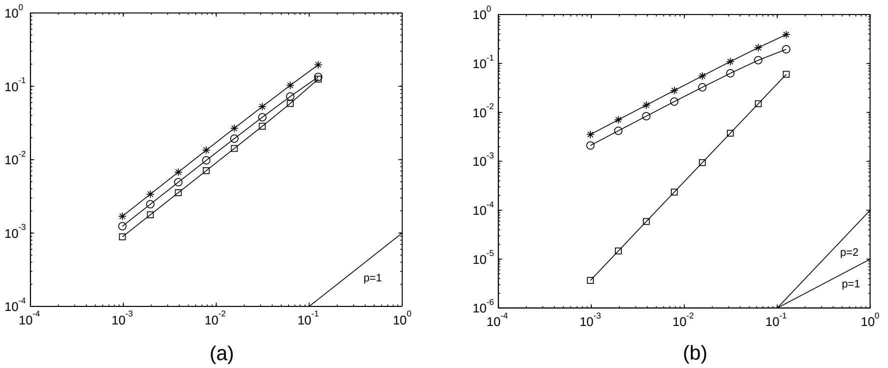


FIG. 8. *Discretization error* $u - u_h$ *for test case* 2: (a) L^2*-norm,* $\mu_a = 1, \mu_b = 100$; (b) *discrete* L^∞*-norm,* $\mu_a = 1, \mu_b = 100$.

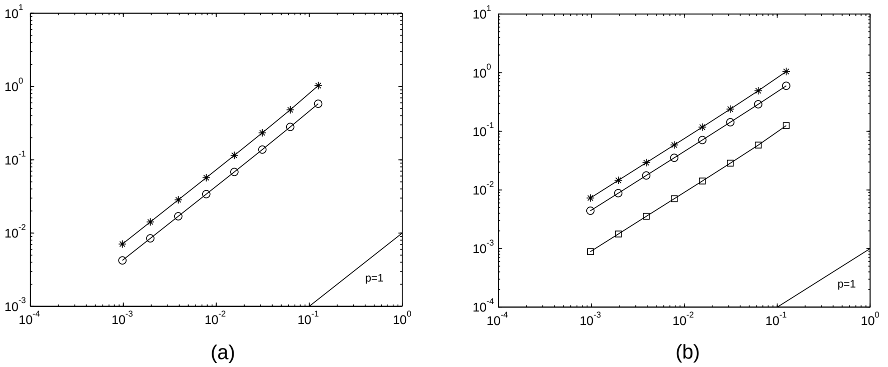


FIG. 9. *Test case* 2: (a) L^2*-norm of the discretization error* $\underline{J} - \underline{J}_h$*,* $\mu_a = 1, \mu_b = 100$; (b) *graph norm of the discretization error* $U - U_h = (u - u_h, \underline{J} - \underline{J}_h)$*,* $\mu_a = 1, \mu_b = 100$.

(8.6). Homogeneous Dirichlet data have been assumed for u along the vertical sides $x = 0$ and $x = 1$, while vanishing fluxes have been imposed for $\underline{J}$ on the rest of the boundary of Ω. The values of the diffusion coefficient μ are $\mu_a = 1$ and $\mu_b = 100$, respectively.

We show in Figures 8(a) and 8(b) the L^2-norm and the discrete L^∞-norm of the error $u - u_h$, this latter norm being evaluated as

$$\|u - u_h\|_{\infty,h} = \max_{T_k \in \mathcal{T}_h} |u(\underline{x}_{B_k}) - u_k|,$$

where $\underline{x}_{B_k}$ is the centroid of the element T_k. Linear convergence with respect to h is observed for all of the considered averages, except for the case of the convergence order in the discrete L^∞-norm of the method using the average (8.4), which proves to be *quadratic* with respect to h. This appears to be a (nodal) superconvergence of the scheme and is to be ascribed to the choice of the average which is *exact* if μ is a piecewise constant function as in the present case. The second-order convergence of the mixed finite volume method using the average (8.4) on quadrilaterals is in agreement with the theoretical results obtained in the literature for the RT finite element of lowest order (see [6, 3, 22, 21]).

Figures 9(a) and 9(b) show the L^2-norm of the discretization error $\underline{J} - \underline{J}_h$ and the graph norm of the error $U - U_h$. Linear convergence is clearly observed for all of the three averages; notice that the curve of the L^2-norm of $\underline{J} - \underline{J}_h$ is not shown for the method using the average (8.4) since in this test case the scheme furnishes the exact solution for $\underline{J}$ (due to the combination of the choice of the average and the fact that the flux is contained in the approximation space). We also point out that the quantity div$\underline{J}_h$ computed by all of the three schemes is *exact* (up to the roundoff error) since the right-hand side f is equal to a constant.

Acknowledgments. The authors gratefully thank the anonymous referees for helping them improve this article.

REFERENCES

[1] D. A. Adams, *Sobolev Spaces*, Academic Press, New York, 1975.

[2] A. Agouzal, J. Baranger, J. F. Maitre, and F. Oudin, *Connection between finite volume and mixed finite element methods for a diffusion problem with nonconstant coefficients. Application to a convection-diffusion problem*, East-West J. Numer. Math., 3 (1995), pp. 237–254.

[3] T. Arbogast and Z. Chen, *On the implementation of mixed methods as nonconforming methods for second-order elliptic problems*, Math. Comp., 64 (1995), pp. 943–972.

[4] T. Arbogast, C. N. Dawson, P. T. Keenan, M. F. Wheeler, and I. Yotov, *Enhanced cell-centered finite differences for elliptic equations on general geometry*, SIAM J. Sci. Comput., 19 (1998), pp. 404–425.

[5] T. Arbogast, M. F. Wheeler, and I. Yotov, *Mixed finite elements for elliptic problems with tensor coefficients as cell-centered finite differences*, SIAM J. Numer. Anal., 34 (1997), pp. 828–852.

[6] D. N. Arnold and F. Brezzi, *Mixed and nonconforming finite element methods: Implementation, postprocessing and error estimates*, RAIRO Modél. Math. Anal. Numér., 19 (1985), pp. 7–32.

[7] I. Babuska and J. E. Osborn, *Generalized finite element methods: Their performance and their relation to mixed methods*, SIAM J. Numer. Anal., 20 (1983), pp. 510–536.

[8] R. E. Bank, J. F. Burgler, W. Fichtner, and R. K. Smith, *Some upwinding techniques for the finite element approximation of convection-diffusion equations*, Numer. Math., 58 (1990), pp. 185–202.

[9] J. Baranger, J. F. Maitre, and F. Oudin, *Application de la théorie des éléments finis mixtes à l'étude d'une classe de schémas aux volumes différences finis pour les problèmes elliptiques*, C. R. Acad. Sci. Paris, 316 (1993), pp. 509–512.

[10] J. Baranger, J. F. Maitre, and F. Oudin, *Connection between finite volume and mixed finite element methods*, M2AN Math. Model. Numer. Anal., 30 (1996), pp. 445–465.

[11] F. Bosisio, S. Micheletti, and R. Sacco, *A discretization scheme for an extended drift-diffusion model including trap-assisted phenomena*, J. Comput. Phys., 159 (2000), pp. 197–212.

[12] F. Brezzi and M. Fortin, *Mixed and Hybrid Finite Element Methods*, Springer-Verlag, New York, 1991.

[13] F. Brezzi, L. D. Marini, and P. Pietra, *Two-dimensional exponential fitting and applications to drift-diffusion models*, SIAM J. Numer. Anal., 26 (1989), pp. 1342–1355.

[14] Z. Cai., J. E. Jones, S. F. McCormick, and T. F. Russell, *Control-volume mixed finite element methods*, Comput. Geosci., 1 (1997), pp. 289–315.

[15] Z. Chen and B. Cockburn, *Error estimates for a finite element method for the drift-diffusion semiconductor device equations*, SIAM J. Numer. Anal., 31 (1994), pp. 1062–1089.

[16] Z. Chen and B. Cockburn, *Analysis of a finite element method for the drift-diffusion semiconductor device equations: The multidimensional case*, Numer. Math., 71 (1995), pp. 1–28.

[17] P. G. Ciarlet and P. A. Raviart, *Maximum principle and uniform convergence for the finite element method*, Comput. Methods Appl. Mech. Engrg., 2 (1973), pp. 17–31.

[18] P. Clément, *Approximation by finite element functions using local regularization*, RAIRO Sér. Rouge Anal. Numér., R-2 (1975), pp. 77–84.

[19] B. Delaunay, *Sur la sphère vide*, Izv. Akad. Nauk. SSSR, Math. Nat. Sci. Div., 6 (1934), pp. 793–800.

[20] R. E. Ewing, O. Saevareid, and J. Shen, *Discretization schemes on triangular grids*, Comput. Methods Appl. Mech. Engrg., 152 (1998), pp. 219–238.

[21] R. E. Ewing, M. M. Liu, and J. Wang, *Superconvergence of mixed finite element approximations over quadrilaterals*, SIAM J. Numer. Anal., 36 (1999), pp. 772–787.

[22] J. P. Hennart and E. del Valle, *On the relationship between nodal schemes and mixed-hybrid finite elements*, Numer. Methods Partial Differential Equations, 9 (1993), pp. 411–430.

[23] J. P. Hennart and E. del Valle, *Mesh-centered finite differences from nodal finite elements for elliptic problems*, Numer. Methods Partial Differential Equations, 14 (1998), pp. 439–465.

[24] J. J. Jerome, *Analysis of Charge Transport. A Mathematical Study of Semiconductor Devices*, Springer-Verlag, Berlin, Heidelberg, 1996.

[25] J. L. Lions and E. Magenes, *Problemes aux limitès non-homogènes et applications*, Dunod, Paris, 1968.

[26] L. D. Marini and P. Pietra, *New mixed finite element schemes for current continuity equations*, COMPEL, 9 (1990), pp. 257–268.

[27] L. D. Marini and P. Pietra, *An abstract theory for mixed approximations of second order elliptic problems*, Mat. Apl. Comput., 8 (1990), pp. 219–239.

[28] P. A. Markowich, *The Stationary Semiconductor Device Equations*, Springer-Verlag, Vienna, 1986.

[29] S. Micheletti and R. Sacco, *Stabilized mixed finite elements for fluid models in semiconductors*, Comput. Visual. Sci., 2 (1999), pp. 139–147.

[30] E. Miglio, A. Quarteroni, and F. Saleri, *Finite element approximation of quasi-*3*D shallow water equations*, Comp. Meth. Appl. Mech. Engrg., 174 (1999), pp. 355–369.

[31] J. Molenaar, *Adaptive multigrid applied to a bipolar transistor problem*, Appl. Numer. Math., 17 (1995), pp. 61–83.

[32] S. J. Polak, W. H. A. Schilders, and H. D. Couperus, *A finite element method with current conservation*, in Proceedings of the Third IEEE International Conference on Simulation of Semiconductor Devices and Processes (Bologna, Italy, 1988), G. Baccarani and M. Rudan, eds., IEEE, Piscataway, NJ, 1990, pp. 1133–1241.

[33] S. J. Polak, *Mixed FEM for* $\triangle u = \alpha u$, in Mathematical Modelling and Simulation of Electrical Circuits and Semiconductor Devices (Oberwolfach, 1988), R. E. Bank, R. Bulirsch, and K. Merten, eds., Birkhäuser Verlag, Basel, 1990, pp. 247–255.

[34] A. Quarteroni and A. Valli, *Numerical Approximation of Partial Differential Equations*, Springer-Verlag, Berlin, 1994.

[35] P. A. Raviart and J. M. Thomas, *A mixed finite element method for second order elliptic problems*, in Mathematical Aspects of the Finite Element Method, Lectures Notes in Math. 606, I. Galligani and E. Magenes, eds., Springer-Verlag, New York, 1977, pp. 292–315.

[36] J. E. Roberts and J. M. Thomas, *Mixed and hybrid methods*, in Handbook of Numerical Analysis, Vol. II, Finite Element Methods (Part I), P. G. Ciarlet and J. L. Lions, eds., North-Holland, Amsterdam, 1991, pp. 523–639.

[37] H. G. Roos, M. Stynes, and L. Tobiska, *Numerical Methods for Singularly Perturbed Differential Equations*, Springer-Verlag, Berlin, Heidelberg, 1996.

[38] R. Sacco, E. Gatti, and L. Gotusso, *The patch-test as a validation of a new finite element for the solution of convection-diffusion equations*, Comput. Methods Appl. Mech. Engrg., 124 (1995), pp. 113–124.

[39] R. Sacco and F. Saleri, *Mixed finite volume methods for semiconductor device simulation*, Numer. Methods Partial Differential Equations, 13 (1997), pp. 215–236.

[40] R. Sacco and F. Saleri, *Stabilized mixed finite volume methods for convection-diffusion problems*, East-West J. Numer. Math., 4 (1997), pp. 291–311.

[41] D. L. Scharfetter and H. K. Gummel, *Large-signal analysis of a silicon Read diode oscillator*, IEEE Trans. Electr. Dev., ED-16 (1969), pp. 64–77.

[42] J. M. Thomas and D. Trujillo, *Finite volume methods for elliptic problems: Convergence on unstructured meshes*, in Numerical Methods in Mechanics, C. Conca and G. Gatica, eds., Longman, Harlow, UK, 1997, pp. 163–174.

[43] R. S. Varga, *Matrix Iterative Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1962.

[44] R. R. P. van Nooyen, *A Petrov-Galerkin mixed finite element method with exponential fitting*, Numer. Methods Partial Differential Equations, 11 (1995), pp. 501–524.

[45] P. S. Vassilevski, S. I. Petrova, and R. D. Lazarov, *Finite difference schemes on triangular cell-centered grids with local refinement*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 1287–1313.

SIAM J. SCI. COMPUT.
Vol. 23, No. 1, pp. 271–303

ON THE EFFICIENT PARALLEL COMPUTATION OF LEGENDRE TRANSFORMS*

MÁRCIA A. INDA[†], ROB H. BISSELING[‡], AND DAVID K. MASLEN[§]

Abstract. In this article, we discuss a parallel implementation of efficient algorithms for computation of Legendre polynomial transforms and other orthogonal polynomial transforms. We develop an approach to the Driscoll–Healy algorithm using polynomial arithmetic and present experimental results on the accuracy, efficiency, and scalability of our implementation. The algorithms were implemented in ANSI C using the BSPlib communications library. We also present a new algorithm for computing the cosine transform of two vectors at the same time.

Key words. orthogonal polynomials, Legendre polynomials, BSP, parallel computation, computational harmonic analysis

AMS subject classifications. 65T50, 65Y05, 42C15

PII. S1064827599355864

1. Introduction.

Discrete Legendre transforms (DLTs) are widely used tools in applied science, commonly arising in problems associated with spherical geometries. Examples of their application include spectral methods for the solution of partial differential equations, e.g., in global weather forecasting [3, 9], shape analysis of molecular surfaces [16], statistical analysis of directional data [18], and geometric quality assurance [17].

A direct method for computing a discrete orthogonal polynomial transform such as the DLT transform for N data values requires a matrix-vector multiplication of $O(N^2)$ arithmetic operations, though several authors [2, 28] have proposed faster algorithms based on approximate methods. In 1989, Driscoll and Healy introduced an exact algorithm that computes such transforms in $O(N \log^2 N)$ arithmetic operations [13, 14, 15]. They implemented the algorithm and analyzed its stability, which depends on the specific orthogonal polynomial sequence used.

Discrete polynomial transforms are computationally intensive, so for large problem sizes the ability to use multiprocessor computers is important, and at least two papers discussing the theoretical parallelizability of the algorithm have already been written [19, 32]. We are, however, unaware of any parallel implementation of the Driscoll–Healy algorithm at the time of writing.

In this paper, we derive a new parallel algorithm that has a lower theoretical time complexity than those of [19, 32], and we present a full implementation of this

*Received by the editors May 12, 1999; accepted for publication (in revised form) February 9, 2001; published electronically June 19, 2001. Parts of this work appeared in preliminary form in *Proceedings of the ECMWF Workshop "Towards TeraComputing—The Use of Parallel Processors in Meteorology,"* W. Zwieflhofer and N. Kreitz, eds., World Scientific, River Edge, NJ, 1999, pp. 87–108. Computer time on the Cray T3E was provided by HPaC, Delft with funding by NCF, The Netherlands. Computer time on the IBM SP was provided by SARA, Amsterdam.

http://www.siam.org/journals/sisc/23-1/35586.html

[†] FOM Institute for Atomic and Molecular Physics (AMOLF), Kruislaan 407, 1098 SJ Amsterdam, The Netherlands (inda@amolf.nl) and Instituto de Matemática, Universidade Federal do Rio Grande do Sul, Av. Bento Gonçalves 9500, 91509-900 Porto Alegre, RS, Brazil. The work of this author was supported by a doctoral fellowship from CAPES, Brazil.

[‡] Department of Mathematics, Utrecht University, P.O. Box 80010, 3508 TA Utrecht, The Netherlands (Rob.Bisseling@math.uu.nl).

[§] Susquehanna Partners, G.P., 401 City Avenue, Suite 220, Bala Cynwyd, PA 19004 (david@maslen.net).

algorithm. Another contribution is the method used to derive the algorithm. We present a method based on polynomial arithmetic to clarify the properties of orthogonal polynomials used by the algorithm and to remove some unnecessary assumptions made in [13, 14, 15].

The remainder of this paper is organized as follows. In section 2, we describe some important properties of orthogonal polynomials and orthogonal polynomial transforms, and we present a derivation of the Driscoll–Healy algorithm. In section 3, we introduce the bulk synchronous parallel (BSP) model and describe a basic parallel algorithm and its implementation. In section 4, we refine the basic algorithm by introducing an intermediate data distribution that reduces the communication to a minimum. In section 5, we present results on the accuracy, efficiency, and scalability of our implementation. We conclude with section 6 and two appendices describing a generalization of the algorithm and the precomputation of the data needed by the algorithm.

2. The Driscoll–Healy algorithm.

2.1. Orthogonal polynomials. A sequence of polynomials $p_0, p_1, p_2, \ldots$ is said to be an *orthogonal polynomial sequence* on the interval $[-1, 1]$ with respect to the weight function $\omega(x)$, if $\deg p_i = i$, and

$$\int_{-1}^{1} p_i(x)p_j(x)\omega(x)dx = 0 \quad \text{for } i \neq j,$$

$$\int_{-1}^{1} p_i(x)^2\omega(x)dx \neq 0 \quad \text{for } i \geq 0.$$

The weight function $\omega(x)$ is nonnegative and integrable on $(-1, 1)$.

Let $x_0, \ldots, x_{N-1}$ be a sequence of distinct real numbers called *sample points*, and let $f_0, \ldots, f_{N-1}$ be a sequence of real values. Then there exists a unique polynomial f of degree less than N such that

$$f(x_j) = f_j, \quad j = 0, \ldots, N-1. \tag{2.1}$$

This polynomial can be obtained by Lagrangian interpolation.

The *expansion transform* corresponding to the orthogonal polynomial sequence $\{p_k\}$ computes the coefficients c_k in the expansion

$$f = \sum_{k=0}^{N-1} c_k p_k, \tag{2.2}$$

where f is a polynomial given by function values f_j in the sample points x_j. (Note that we do not require any special relation between the sample points and the orthogonal polynomials.) The *inverse expansion transform* evaluates f at the sample points x_j, and this can be done by straightforward substitution:

$$f_j = \sum_{k=0}^{N-1} c_k p_k(x_j), \quad j = 0, \ldots, N-1. \tag{2.3}$$

In matrix-vector notation, the latter transform can be written as $\mathbf{f} = \mathbf{Pc}$, where $\mathbf{f} = (f_0, \ldots, f_{N-1})$ and $\mathbf{c} = (c_0, \ldots, c_{N-1})$ are column vectors, and the matrix $\mathbf{P}$ is defined by $P_{jk} = p_k(x_j)$. The matrix $\mathbf{P}$ is invertible, and $\mathbf{P}^{-1}$ represents the expansion transform. In general, $\mathbf{P}$ need not be orthogonal, and hence its inverse and transpose need not be the same.

EXAMPLE 2.1 (discrete Chebyshev transform (DChT)). *The Chebyshev polynomials of the first kind are the sequence of orthogonal polynomials defined recursively by*

$$T_{k+1}(x) = 2x \cdot T_k(x) - T_{k-1}(x), \quad T_0(x) = 1, \quad T_1(x) = x. \tag{2.4}$$

These are orthogonal with respect to the weight function $\omega(x) = \pi^{-1}(1-x^2)^{-\frac{1}{2}}$ *on* $[-1,1]$*, and they satisfy* $T_k(\cos\theta) = \cos k\theta$ *for all real* θ*.*

The DChT is the expansion transform for the Chebyshev polynomials at the Chebyshev points. The Chebyshev points *are the roots of* T_N*, and they are given by*

$$x_j^N = \cos\frac{(2j+1)\pi}{2N}, \quad j = 0, \ldots, N-1. \tag{2.5}$$

We denote the Chebyshev transform by a tilde. More specifically, the coefficient of T_k *in the Chebyshev expansion of a polynomial* f *is denoted by* $\tilde{f}_k$*.*

The inverse Chebyshev transform can straightforwardly be written as

$$f_j = \sum_{k=0}^{N-1} \tilde{f}_k T_k(x_j^N) = \sum_{k=0}^{N-1} \tilde{f}_k \cos\frac{(2j+1)k\pi}{2N}, \quad j = 0, \ldots, N-1. \tag{2.6}$$

Furthermore, it can be shown that the Chebyshev transform itself is given by

$$\tilde{f}_k = \frac{\epsilon_k}{N}\sum_{j=0}^{N-1} f_j T_k(x_j^N) = \frac{\epsilon_k}{N}\sum_{j=0}^{N-1} f_j \cos\frac{(2j+1)k\pi}{2N}, \quad k = 0, \ldots, N-1, \tag{2.7}$$

where

$$\epsilon_k = \begin{cases} 1 & \text{if } k = 0, \\ 2 & \text{if } k > 0. \end{cases} \tag{2.8}$$

In this work, we will study a slightly more general transform which includes weights. Given an orthogonal polynomial sequence $\{p_k\}$, a sequence of sample points $x_0, \ldots, x_{N-1}$, and a sequence of numbers $w_0, \ldots, w_{N-1}$ called *sample weights*, we define the *discrete orthogonal polynomial transform* of a data vector $(f_0, \ldots, f_{N-1})$ to be the vector of sums $(\hat{f}_0, \ldots, \hat{f}_{N-1})$, where

$$\hat{f}_k = \hat{f}(p_k) = \sum_{j=0}^{N-1} w_j f_j p_k(x_j). \tag{2.9}$$

The matrix of the discrete orthogonal polynomial transform (2.9) for the special case with sample weights 1 is $\mathbf{P}^T$.

EXAMPLE 2.2 (discrete cosine transform (DCT)). *The DCT, or DCT*-II *in the terminology of* [35]*, is the discrete orthogonal polynomial transform for the Chebyshev polynomials, with sample weights* 1 *and with the Chebyshev points as sample points. Thus, the matrix representing the DCT is* $\mathbf{P}^T$*. Since the matrix representing the DChT is* $\mathbf{P}^{-1}$*, the DCT is the inverse transpose of the DChT. The relation is even closer: by comparing* (2.9) *for the DCT with* (2.7) *for the DChT we see that the DChT is equivalent to a DCT followed by a multiplication of the kth coefficient by* $\frac{\epsilon_k}{N}$*.*

A DCT can be carried out in $O(N \log N)$ arithmetic operations using a fast Fourier transform (FFT) [1, 35] *or using the recent algorithm of Steidl and Tasche* [33]. *Such an $O(N \log N)$ algorithm is called a fast cosine transform (FCT). This also provides us with a fast Chebyshev transform (FChT). We use an upper bound of the form $\alpha N \log_2 N + \beta N$ for the number of floating point operations (flops) for one FChT of size N, or its inverse. The lower order term is included because we are often interested in small size transforms, for which this term may be dominant.*

Example 2.3 (DLT). *The Legendre polynomials are orthogonal with respect to the uniform weight function 1 on $[-1, 1]$, and they may be defined recursively by*

$$(2.10)\quad P_{k+1}(x) = \frac{2k+1}{k+1} x \cdot P_k(x) - \frac{k}{k+1} P_{k-1}(x), \quad P_0(x) = 1, \quad P_1(x) = x.$$

The Legendre polynomials are one of the most important examples of orthogonal polynomials, as they occur as zonal polynomials in the spherical harmonic expansion of functions on the sphere. Our parallel implementation of the Driscoll–Healy algorithm, to be described later, focuses on the case of Legendre polynomials. For efficiency reasons, we sample these polynomials at the Chebyshev points. In this paper, we call the discrete orthogonal polynomial transform for the Legendre polynomials, with sample weights $1/N$ and with the Chebyshev points as sample points, the DLT.

One of the important properties of orthogonal polynomials we will use is the following lemma.

Lemma 2.4 (Gaussian quadrature). *Let $\{p_k\}$ be an orthogonal polynomial sequence for a nonnegative integrable weight function $\omega(x)$, and let $z_0^N, \ldots, z_{N-1}^N$ be the roots of p_N. Then there exist numbers $w_0^N, \ldots, w_{N-1}^N > 0$, such that for any polynomial f of degree less than $2N$ we have*

$$\int_{-1}^{1} f(x)\omega(x)dx = \sum_{j=0}^{N-1} w_j^N f(z_j^N).$$

The numbers w_j^N are unique and are called the Gaussian weights for the sequence $\{p_k\}$.

Proof. See, e.g., [10, Theorem 6.1]. □

Example 2.5. *The Gaussian weights for the Chebyshev polynomials with weight function $\pi^{-1}(1-x^2)^{-\frac{1}{2}}$ are $w_j^N = 1/N$. So for any polynomial f of degree less than $2N$ we have*

$$(2.11)\qquad \frac{1}{\pi}\int_{-1}^{1} \frac{f(x)dx}{\sqrt{1-x^2}} = \frac{1}{N}\sum_{j=0}^{N-1} f(x_j^N),$$

where $x_j^N = \cos\frac{(2j+1)\pi}{2N}$ are the Chebyshev points.

Another property of orthogonal polynomials that we will need is the existence of a three-term recurrence relation, such as (2.4) for the Chebyshev polynomials and (2.10) for the Legendre polynomials.

Lemma 2.6 (three-term recurrence). *Let $\{p_k\}$ be an orthogonal polynomial sequence for a nonnegative integrable weight function. Then $\{p_k\}$ satisfies a three-term recurrence relation*

$$(2.12)\qquad p_{k+1}(x) = (A_k x + B_k)p_k(x) + C_k p_{k-1}(x),$$

where A_k, B_k, C_k are real numbers with $A_k \neq 0$ and $C_k \neq 0$.

Proof. See, e.g., [10, Theorem 4.1]. □

The Clebsch–Gordan property follows from, and is similar to, the three-term recurrence.

Corollary 2.7 (Clebsch–Gordan). *Let $\{p_k\}$ be an orthogonal polynomial sequence with a nonnegative integrable weight function. Then for any polynomial Q of degree m we have*

$$p_k \cdot Q \in \operatorname{span}_{\mathbf{R}}\{p_{k-m}, \dots, p_{k+m}\}.$$

Proof. Rewrite the recurrence (2.12) in the form $x \cdot p_k = A_k^{-1}(p_{k+1} - B_k p_k - C_k p_{k-1})$, and use induction on m. □

Iterating the three-term recurrence also gives a more general recurrence between polynomials in an orthogonal polynomial sequence. Define the *associated polynomials* $Q_{l,m}, R_{l,m}$ for the orthogonal polynomial sequence $\{p_l\}$ by the following recurrences on m, which are shifted versions of the recurrence for p_l. See, e.g., [4, 5].

$$\begin{aligned} Q_{l,m}(x) &= (A_{l+m-1}x + B_{l+m-1})Q_{l,m-1}(x) + C_{l+m-1}Q_{l,m-2}(x), \\ Q_{l,0}(x) &= 1, \quad Q_{l,1}(x) = A_l x + B_l, \\ R_{l,m}(x) &= (A_{l+m-1}x + B_{l+m-1})R_{l,m-1}(x) + C_{l+m-1}R_{l,m-2}(x), \\ R_{l,0}(x) &= 0, \quad R_{l,1}(x) = C_l. \end{aligned} \tag{2.13}$$

Lemma 2.8 (generalized three-term recurrence). *The associated polynomials satisfy* $\deg Q_{l,m} = m$, $\deg R_{l,m} \le m - 1$, *and for* $l \ge 1$ *and* $m \ge 0$,

$$p_{l+m} = Q_{l,m} \cdot p_l + R_{l,m} \cdot p_{l-1}. \tag{2.14}$$

Proof. Equation (2.14) follows by induction on m with the case $m = 1$ being the original three-term recurrence (2.12). □

In the case where the p_l are the Legendre polynomials, the associated polynomials should not be confused with the associated Legendre functions, which in general are not polynomials.

2.2. Derivation of the Driscoll–Healy algorithm. The Driscoll–Healy algorithm [13, 14] allows one to compute orthogonal polynomial transforms at any set of N sample points, in $O(N \log^2 N)$ arithmetic operations. The core of this algorithm consists of an algorithm to compute orthogonal polynomial transforms in the special case where the sample points are the Chebyshev points and the sample weights are $1/N$. For simplicity we restrict ourselves to this special case, and, furthermore, we assume that N is a power of 2. In Appendix A, we sketch extensions to more general problems.

Using the relation

$$f \cdot p_{l+m} = Q_{l,m} \cdot (f \cdot p_l) + R_{l,m} \cdot (f \cdot p_{l-1}), \tag{2.15}$$

derived from the three-term recurrence (2.14), we may formulate a strategy for computing all the polynomials $f \cdot p_l$, $0 \le l < N$, in $\log_2 N$ stages.

- At stage 0, compute $f \cdot p_0$ and $f \cdot p_1$.
- At stage 1, use (2.15) with $l = 1$ and $m = N/2 - 1$ or $m = N/2$ to compute
$$\begin{aligned} f \cdot p_{\frac{N}{2}} &= Q_{1,\frac{N}{2}-1} \cdot (f \cdot p_1) + R_{1,\frac{N}{2}-1} \cdot (f \cdot p_0), \\ f \cdot p_{\frac{N}{2}+1} &= Q_{1,\frac{N}{2}} \cdot (f \cdot p_1) + R_{1,\frac{N}{2}} \cdot (f \cdot p_0). \end{aligned}$$
- In general, at each stage $k, 1 \le k < \log_2 N$, similarly as before, use (2.15) with $l = 2q(N/2^k) + 1$, $0 \le q < 2^{k-1}$, and $m = N/2^k - 1$ or $N/2^k$, to compute the polynomial pairs

$$f \cdot p_{\frac{N}{2^k}},\ f \cdot p_{\frac{N}{2^k}+1};\ f \cdot p_{\frac{3N}{2^k}},\ f \cdot p_{\frac{3N}{2^k}+1};\ \cdots\ ; f \cdot p_{\frac{(2^k-1)N}{2^k}}, f \cdot p_{\frac{(2^k-1)N}{2^k}+1}.$$

The problem with this strategy is that computing a full representation of each polynomial $f \cdot p_l$ generates much more data at each stage than is needed to compute the final output. To overcome this problem, the Driscoll–Healy algorithm uses *Chebyshev truncation operators* to discard unneeded information at the end of each stage. Let $f = \sum_{k \geq 0} b_k T_k$ be a polynomial, of any degree, written in the basis of Chebyshev polynomials, and let n be a positive integer. Then the truncation operator $\mathcal{T}_n$ applied to f is defined by

$$\mathcal{T}_n f = \sum_{k=0}^{n-1} b_k T_k. \tag{2.16}$$

The important properties of $\mathcal{T}_n$ are given in Lemma 2.9.

LEMMA 2.9. *Let f and Q be polynomials. Then the following hold.*

1. $\mathcal{T}_1 f = \int_{-1}^{1} f(x)\omega(x)dx$, *where* $\omega(x) = \pi^{-1}(1-x^2)^{-\frac{1}{2}}$.
2. *If* $m \leq n$, *then* $\mathcal{T}_m \mathcal{T}_n = \mathcal{T}_m$.
3. *If* $\deg Q \leq m \leq n$, *then* $\mathcal{T}_{n-m}(f \cdot Q) = \mathcal{T}_{n-m}[(\mathcal{T}_n f) \cdot Q]$.

Proof. Part 1 follows from the orthogonality of Chebyshev polynomials, as $\mathcal{T}_1 f$ is just the constant term of f in its expansion in Chebyshev polynomials. Part 2 is a trivial consequence of the definition of truncation operators. For part 3 we assume that $f = \sum_{k \geq 0} b_k T_k$ is a polynomial and that $\deg Q \leq m \leq n$. By Corollary 2.7, $T_k \cdot Q$ is in the linear span of $T_{k-m}, \ldots, T_{k+m}$, so $\mathcal{T}_{n-m}(T_k \cdot Q) = 0$ for $k \geq n$. Therefore,

$$\mathcal{T}_{n-m}(f \cdot Q) = \mathcal{T}_{n-m}\left(\sum_{k \geq 0} b_k T_k \cdot Q\right) = \mathcal{T}_{n-m}\left(\sum_{k=0}^{n-1} b_k T_k \cdot Q\right) = \mathcal{T}_{n-m}[(\mathcal{T}_n f) \cdot Q]. \qquad \square$$

As a corollary of part 1 of Lemma 2.9, we see how we can retrieve the discrete orthogonal polynomial transform from the $f \cdot p_l$'s computed by the strategy above by using a simple truncation.

COROLLARY 2.10. *Let f be the unique polynomial of degree less than N such that $f(x_j^N) = f_j$, $0 \leq j < N$. Let $\{p_l\}$ be an orthogonal polynomial sequence. Then*

$$\hat{f}_l = \mathcal{T}_1(f \cdot p_l), \quad 0 \leq l < N,$$

where the $\hat{f}_l$ form the discrete orthogonal polynomial transform of f of size N with respect to the sample points x_j^N and sample weights $1/N$.

Proof. This follows from the definition of discrete orthogonal polynomial transforms, the Gaussian quadrature rule (2.11) for Chebyshev polynomials applied to the function $f \cdot p_l$, and Lemma 2.9,

$$\hat{f}_l = \frac{1}{N}\sum_{j=0}^{N-1} f(x_j^N) p_l(x_j^N) = \frac{1}{\pi}\int_{-1}^{1} \frac{f(x)p_l(x)}{\sqrt{1-x^2}}dx = \mathcal{T}_1(f \cdot p_l). \qquad \square$$

The key property of the truncation operators $\mathcal{T}_n$ is the "aliasing" property (part 3 of Lemma 2.9), which states that we may use a truncated version of f when computing a truncated product of f and Q. For example, if we wish to compute the truncated product $\mathcal{T}_1(f \cdot p_l)$ with $l, \deg f < N$ then, because $\deg p_l = l$, we may apply part 3 of Lemma 2.9 with $m = l$ and $n = l + 1$ to obtain

$$\hat{f}_l = \mathcal{T}_1(f \cdot p_l) = \mathcal{T}_1[(\mathcal{T}_{l+1} f) \cdot p_l].$$

Thus we need to know only the first $l+1$ Chebyshev coefficients of f to compute $\hat{f}_l$.

The Driscoll–Healy algorithm follows the strategy described at the start of this section, but it computes truncated polynomials

$$Z_l^K = \mathcal{T}_K(f \cdot p_l) \tag{2.17}$$

for various values of l and K, instead of the original polynomials $f \cdot p_l$. The input is the polynomial f, and the output is $\hat{f}_l = \mathcal{T}_1(f \cdot p_l) = Z_l^1$, $0 \le l < N$.

Each stage of the algorithm uses truncation operators to discard unneeded information, which keeps the problem size down. Instead of using the generalized three-term recurrence (2.15) directly, each stage uses truncated versions. Specifically, (2.15) with $m = K-1, K$ and part 3 of Lemma 2.9 with $m = K$ and $n = 2K$ imply the following recurrences for the Z_l^K:

$$Z_{l+K-1}^K = \mathcal{T}_K[Z_l^{2K} \cdot Q_{l,K-1} + Z_{l-1}^{2K} \cdot R_{l,K-1}], \tag{2.18}$$

$$Z_{l+K}^K \quad = \mathcal{T}_K[Z_l^{2K} \cdot Q_{l,K} + Z_{l-1}^{2K} \cdot R_{l,K}]. \tag{2.19}$$

The algorithm proceeds in $\log_2 N + 1$ stages, as shown in Algorithm 2.1. The organization of the computation is illustrated in Figure 2.1.

Algorithm 2.1 Polynomial version of the Driscoll–Healy algorithm.

```
INPUT  $(f_0, \ldots, f_{N-1})$: Polynomial defined by $f_j = f(x_j^N)$; $N$ is a power of 2.
OUTPUT  $(\hat{f}_0, \ldots, \hat{f}_{N-1})$: Transformed polynomial with $\hat{f}_l = \mathcal{T}_1(f \cdot p_l) = Z_l^1$.
STAGES
        0. Compute $Z_0^N \leftarrow f \cdot p_0$ and $Z_1^N \leftarrow \mathcal{T}_N(f \cdot p_1)$.
        k. for $k = 1$ to $\log_2 N - 1$ do
               $K \leftarrow \frac{N}{2^k}$
               for $l = 1$ to $N - 2K + 1$ step $2K$ do
                 (a) Use recurrence (2.18) and (2.19) to compute new polynomials.
                     $Z_{l+K-1}^K \leftarrow \mathcal{T}_K \left(Z_l^{2K} \cdot Q_{l,K-1} + Z_{l-1}^{2K} \cdot R_{l,K-1}\right)$
                     $Z_{l+K}^K \leftarrow \mathcal{T}_K \left(Z_l^{2K} \cdot Q_{l,K} + Z_{l-1}^{2K} \cdot R_{l,K}\right)$
                 (b) Truncate old polynomials.
                     $Z_{l-1}^K \leftarrow \mathcal{T}_K Z_{l-1}^{2K}$
                     $Z_l^K \leftarrow \mathcal{T}_K Z_l^{2K}$
  $\log_2 N$. for $l = 0$ to $N - 1$ do
               $\hat{f}_l \leftarrow Z_l^1$
```

2.3. Data representation and recurrence procedure. To complete our description of the Driscoll–Healy algorithm, we still need to specify how to represent the polynomials in the algorithm and to describe the methods used to multiply two polynomials and to apply the truncation operators $\mathcal{T}_K$. This is done in the following subsections.

2.3.1. Chebyshev representation of polynomials. Truncation of a polynomial requires no computation if the polynomial is represented by the coefficients of its expansion in Chebyshev polynomials. Therefore, we use the Chebyshev coefficients z_n^l defined by

$$Z_l^K = \sum_{n=0}^{K-1} z_n^l T_n \tag{2.20}$$

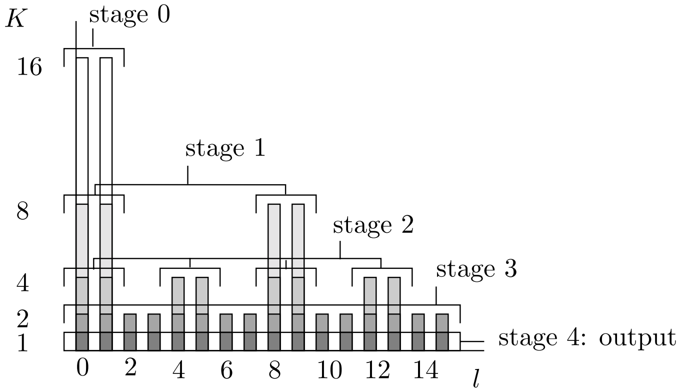


FIG. 2.1. *Computation of the truncated polynomials Z_l^K for $N = 16$. Each bar represents the polynomials Z_l^K for one value of l. The height of the bar represents the initial number of Chebyshev coefficients. At each stage, the number of coefficients is reduced as indicated by the gray scales.*

to represent all the polynomials Z_l^K appearing in the algorithm. Such a representation of a polynomial is called the *Chebyshev representation*.

The input polynomial f of degree less than N is given as the vector $\mathbf{f} = (f_0, \ldots, f_{N-1})$ of values $f_j = f(x_j^N)$. This is called the *point-value representation* of f. In stage 0, we convert $Z_0^N = \mathcal{T}_N(f \cdot p_0) = f \cdot p_0$ and $Z_1^N = \mathcal{T}_N(f \cdot p_1)$ to their Chebyshev representations. For $f \cdot p_0$ this can be done by a Chebyshev transform on the vector of function values with the input values multiplied by the constant p_0. For $f \cdot p_1$ we also use a Chebyshev transform of size N, even though $f \cdot p_1$ may have degree N, rather than $N - 1$. This poses no problem, because applying part 4 of Lemma 2.11 from the next subsection with $h = f \cdot p_1$ and $K = N$ proves that $f \cdot p_1$ agrees with Z_1^N at the sampling points x_j^N. Stage 0 becomes the following.

```
Stage 0. Compute the Chebyshev representation of $Z_0^N$ and $Z_1^N$.
         (a) $(z_0^0, \ldots, z_{N-1}^0) \leftarrow \text{Chebyshev}(f_0 p_0, \ldots, f_{N-1} p_0)$
         (b) $(z_0^1, \ldots, z_{N-1}^1) \leftarrow \text{Chebyshev}(f_0 p_1(x_0^N), \ldots, f_{N-1} p_1(x_{N-1}^N))$
```

Stage 0 takes a total of $2\alpha N \log_2 N + 2\beta N + 2N$ flops, where the third term represents the $2N$ flops needed to multiply f with p_0 and p_1.

2.3.2. Recurrence using Chebyshev transforms. To apply the recurrences (2.18) and (2.19) efficiently, we do the following.

1. Apply inverse Chebyshev transforms of size $2K$ to bring the polynomials Z_{l-1}^{2K}, Z_l^{2K} into point-value representation at the points x_j^{2K}, $0 \leq j < 2K$.
2. Perform the multiplications and additions.
3. Apply a forward Chebyshev transform of size $2K$ to bring the result into Chebyshev representation.
4. Truncate the results to degree less than K.

This procedure replaces the polynomial multiplications in the recurrences (2.18) and (2.19) by a slightly different operation. Because the multiplications are made in only $2K$ points, whereas the degree of the resulting polynomial could be $3K - 1$, we must verify that the end result is the same. To describe the operation formally, we introduce the *Lagrange interpolation operators* $\mathcal{S}_n$ for positive integers n. For any polynomial h, the Lagrange interpolation polynomial $\mathcal{S}_n h$ is the polynomial of degree less than n which agrees with h at the points $x_0^n, \dots, x_{n-1}^n$. The important properties of $\mathcal{S}_n$ are given in Lemma 2.11.

Lemma 2.11. *Let g and h be polynomials. Then the following hold.*

1. *If $\deg h < n$, then $\mathcal{S}_n h = h$.*
2. *$\mathcal{S}_n(g \cdot h) = \mathcal{S}_n((\mathcal{S}_n g) \cdot (\mathcal{S}_n h))$.*
3. *Let $m \leq n$. If $\deg h \leq m + n$, then $\mathcal{T}_{n-m} h = \mathcal{T}_{n-m}\mathcal{S}_n h$.*
4. *If $\deg h = n$, then $\mathcal{S}_n h = \mathcal{T}_n h$.*

Proof. Parts 1 and 2 are easy. To prove part 3 assume that $\deg h \leq m + n$. By long division, there is a polynomial Q of degree at most m such that $h = \mathcal{S}_n h + T_n \cdot Q$. Applying $\mathcal{T}_{n-m}$ and using part 3 of Lemma 2.9, we obtain

$$\mathcal{T}_{n-m}\mathcal{S}_n h = \mathcal{T}_{n-m} h - \mathcal{T}_{n-m}[T_n \cdot Q] = \mathcal{T}_{n-m} h - \mathcal{T}_{n-m}[(\mathcal{T}_n T_n) \cdot Q] = \mathcal{T}_{n-m} h,$$

since $\mathcal{T}_n T_n = 0$. For part 4 we note that $\deg \mathcal{S}_n h < n$, and we use part 3 with $m = 0$ to obtain $\mathcal{S}_n h = \mathcal{T}_n \mathcal{S}_n h = \mathcal{T}_n h$. □

From the recurrences (2.18) and (2.19) and part 3 of Lemma 2.11 with $m = K$ and $n = 2K$, it follows that

$$Z_{l+K-1}^K = \mathcal{T}_K[\mathcal{S}_{2K}(Z_l^{2K} \cdot Q_{l,K-1}) + \mathcal{S}_{2K}(Z_{l-1}^{2K} \cdot R_{l,K-1})], \tag{2.21}$$

$$Z_{l+K}^K \quad = \mathcal{T}_K[\mathcal{S}_{2K}(Z_l^{2K} \cdot Q_{l,K}) + \mathcal{S}_{2K}(Z_{l-1}^{2K} \cdot R_{l,K})]. \tag{2.22}$$

These equations are exactly the procedure described above. The inner loop of stage k of Algorithm 2.1 becomes the following.

```
(a) Compute the Chebyshev representation of $Z^K_{l+K-1}$ and $Z^K_{l+K}$.
    $(z_0^{l+K-1}, \dots, z_{K-1}^{l+K-1};\ z_0^{l+K}, \dots, z_{K-1}^{l+K})$
        $\leftarrow \mathrm{Recurrence}_l^K(z_0^{l-1}, \dots, z_{2K-1}^{l-1};\ z_0^l, \dots, z_{2K-1}^l)$
(b) Compute the Chebyshev representation of $Z^K_{l-1}$ and $Z^K_l$.
    Discard $(z_K^{l-1}, \dots, z_{2K-1}^{l-1})$ and $(z_K^l, \dots, z_{2K-1}^l)$
```

Algorithm 2.2 describes in detail the recurrence procedure, which takes $4(\alpha \cdot 2K \log_2 2K + \beta \cdot 2K) + 12K = 8\alpha K \log_2 K + (8\alpha + 8\beta + 12)K$ flops.

2.4. Early termination. At late stages in the Driscoll–Healy algorithm, the work required to apply the recursion amongst the Z_l^K is larger than that required to finish the computation using a naive matrix-vector multiplication. It is then more efficient to use the vectors Z_l^K computed so far directly to obtain the final result, as follows.

Let $q_{l,m}^n$ and $r_{l,m}^n$ denote the Chebyshev coefficients of the polynomials $Q_{l,m}$ and $R_{l,m}$, respectively, so that

$$Q_{l,m} = \sum_{n=0}^{m} q_{l,m}^n T_n, \quad R_{l,m} = \sum_{n=0}^{m-1} r_{l,m}^n T_n. \tag{2.23}$$

The problem of finishing the computation at the end of stage $k = \log_2(N/M)$, when $K = M$, is equivalent to finding $\hat{f}_l = z_0^l$ for $0 \leq l < N$, given the data z_n^l, z_n^{l-1}, $0 \leq n < M$, $l = 1, M + 1, 2M + 1, \dots, N - M + 1$. Our method of finishing the computation is to use part 1 of Lemma 2.12, which follows. Part 2 of this lemma can be used to halve the number of computations in the common case, where the polynomial recurrence (2.12) has a coefficient $B_k = 0$ for all k.

Algorithm 2.2 Recurrence procedure using the Chebyshev transform.

```
CALL Recurrence$_l^K(\tilde{f}_0,\dots,\tilde{f}_{2K-1};\ \tilde{g}_0,\dots,\tilde{g}_{2K-1})$.
INPUT $\tilde{\mathbf{f}} = (\tilde{f}_0,\dots,\tilde{f}_{2K-1})$ and $\tilde{\mathbf{g}} = (\tilde{g}_0,\dots,\tilde{g}_{2K-1})$: First $2K$ Chebyshev coefficients of input
polynomials $Z_{l-1}^{2K}$ and $Z_l^{2K}$; $K$ is a power of 2.
OUTPUT $\tilde{\mathbf{u}} = (\tilde{u}_0,\dots,\tilde{u}_{K-1})$ and $\tilde{\mathbf{v}} = (\tilde{v}_0,\dots,\tilde{v}_{K-1})$: First $K$ Chebyshev coefficients of output
polynomials $Z_{l+K-1}^K$ and $Z_{l+K}^K$.
STEPS
    1. Transform $\tilde{\mathbf{f}}$ and $\tilde{\mathbf{g}}$ to point-value representation.
       $(f_0,\dots,f_{2K-1}) \leftarrow \text{Chebyshev}^{-1}(\tilde{f}_0,\dots,\tilde{f}_{2K-1})$
       $(g_0,\dots,g_{2K-1}) \leftarrow \text{Chebyshev}^{-1}(\tilde{g}_0,\dots,\tilde{g}_{2K-1})$
    2. Perform the recurrence.
       for $j = 0$ to $2K-1$ do
           $u_j \leftarrow Q_{l,K-1}(x_j^{2K})\, g_j + R_{l,K-1}(x_j^{2K})\, f_j$
           $v_j \leftarrow Q_{l,K}(x_j^{2K})\, g_j + R_{l,K}(x_j^{2K})\, f_j$
    3. Transform $\mathbf{u}$ and $\mathbf{v}$ to Chebyshev representation.
       $(\tilde{u}_0,\dots,\tilde{u}_{2K-1}) \leftarrow \text{Chebyshev}(u_0,\dots,u_{2K-1})$
       $(\tilde{v}_0,\dots,\tilde{v}_{2K-1}) \leftarrow \text{Chebyshev}(v_0,\dots,v_{2K-1})$
    4. Discard $(\tilde{u}_K,\dots,\tilde{u}_{2K-1})$ and $(\tilde{v}_K,\dots,\tilde{v}_{2K-1})$.
```

LEMMA 2.12.

1. *If $l \geq 1$ and $0 \leq m < M$, then*

$$\hat{f}_{l+m} = \sum_{n=0}^{m} \frac{1}{\epsilon_n}(z_n^l q_{l,m}^n + z_n^{l-1} r_{l,m}^n). \tag{2.24}$$

2. *If p_l satisfies a recurrence of the form $p_{l+1}(x) = A_l x p_l(x) + C_l p_{l-1}(x)$, then*

$$\begin{aligned} q_{l,m}^n &= 0 \quad \textit{if } n-m \textit{ is odd, and} \\ r_{l,m}^n &= 0 \quad \textit{if } n-m \textit{ is even.} \end{aligned}$$

Proof. Applying $\mathcal{T}_{M-m}$ to both sides of (2.15) and using part 3 of Lemma 2.9 with $n = M$ gives $Z_{l+m}^{M-m} = \mathcal{T}_{M-m}(Z_l^M \cdot Q_{l,m} + Z_{l-1}^M \cdot R_{l,m})$. Truncating again, now using $\mathcal{T}_1$, we see that $\hat{f}_{l+m} = Z_{l+m}^1$ is the constant term of the Chebyshev expansion of $Z_l^M \cdot Q_{l,m} + Z_{l-1}^M \cdot R_{l,m}$. To find this constant term expressed in the Chebyshev coefficients of Z_l^M, Z_{l-1}^M and of $Q_{l,m}, R_{l,m}$, we substitute the expansions (2.20) and (2.23) and rewrite the product of sums by using the identity $T_j \cdot T_k = \frac{1}{2}(T_{|j-k|} + T_{j+k})$. For the second part, we assume that p_l satisfies the given recurrence. Then $Q_{l,m}$ is odd or even according to whether m is odd or even, and $R_{l,m}$ is even or odd according to whether m is odd or even, which can be verified by induction on m. This implies that the Chebyshev expansion of $Q_{l,m}$ must contain only odd or even coefficients, respectively, and the reverse must hold for $R_{l,m}$. □

Assuming that the assumptions of part 2 of the lemma are valid, i.e., each term of (2.24) has either $q_{l,m}^n = 0$ or $r_{l,m}^n = 0$, and that the factor $1/\epsilon_n$ has been absorbed in the precomputed values $q_{l,m}^n$ and $r_{l,m}^n$, the total number of flops needed to compute $\hat{f}_{l+m}$ is $2m+1$.

2.5. Complexity of the algorithm. Algorithm 2.3 gives the sequential Driscoll–Healy algorithm in its final form. The total number of flops can be computed as follows. Stage 0 takes $2\alpha N \log_2 N + (2\beta + 2)N$ flops. Stage k invokes $N/(2K)$ times the recurrence procedure, which has cost $8\alpha K \log_2 K + (8\alpha + 8\beta + 12)K$ flops, so that the total cost of that stage is $4\alpha N \log_2 K + (4\alpha + 4\beta + 6)N$ flops. Adding the costs for $K = N/2, \ldots, M$ gives $2\alpha N[\log_2^2 N - \log_2^2 M] + (2\alpha + 4\beta + 6)N[\log_2 N - \log_2 M]$ flops. In the last stage, output values have to be computed for $m = 1, \ldots, M - 2$, for each of the N/M values of l. This gives a total of $\frac{N}{M} \sum_{m=1}^{M-2} (2m + 1) = NM - 2N$ flops. Summing the costs gives

$$\begin{aligned} T_{\text{Driscoll-Healy}} =& N[2\alpha(\log_2^2 N - \log_2^2 M) + (4\alpha + 4\beta + 6) \log_2 N \\ & - (2\alpha + 4\beta + 6) \log_2 M + M + 2\beta]. \end{aligned} \tag{2.25}$$

Algorithm 2.3 Driscoll–Healy algorithm.

```
INPUT  $\mathbf{f} = (f_0, \ldots, f_{N-1})$: Real vector with $N$ a power of 2.
OUTPUT  $\hat{\mathbf{f}} = (\hat{f}_0, \ldots, \hat{f}_{N-1})$: Discrete orthogonal polynomial transform of $\mathbf{f}$.
STAGES
            0. Compute the Chebyshev representation of $Z_0^N$ and $Z_1^N$.
               (a) $(z_0^0, \ldots, z_{N-1}^0) \leftarrow \text{Chebyshev}(f_0 p_0, \ldots, f_{N-1} p_0)$
               (b) $(z_0^1, \ldots, z_{N-1}^1) \leftarrow \text{Chebyshev}(f_0 p_1(x_0^N), \ldots, f_{N-1} p_1(x_{N-1}^N))$
            $k$. for $k = 1$ to $\log_2 \frac{N}{M}$ do
                   $K \leftarrow \frac{N}{2^k}$
                   for $l = 1$ to $N - 2K + 1$ step $2K$ do
                     (a) Compute the Chebyshev representation of $Z_{l+K-1}^K$ and $Z_{l+K}^K$
                           $(z_0^{l+K-1}, \ldots, z_{K-1}^{l+K-1};\ z_0^{l+K}, \ldots, z_{K-1}^{l+K})$
                              $\leftarrow \text{Recurrence}_l^K(z_0^{l-1}, \ldots, z_{2K-1}^{l-1};\ z_0^l, \ldots, z_{2K-1}^l)$
                     (b) Compute the Chebyshev representation of $Z_{l-1}^K$ and $Z_l^K$.
                           Discard $(z_K^{l-1}, \ldots, z_{2K-1}^{l-1})$ and $(z_K^l, \ldots, z_{2K-1}^l)$
$\log_2 \frac{N}{M} + 1$. Compute the remaining values.
            for $l = 1$ to $N - M + 1$ step $M$ do
                   $\hat{f}_{l-1} \leftarrow z_0^{l-1}$
                   $\hat{f}_l \leftarrow z_0^l$
                   for $m = 1$ to $M - 2$ do
                         $\hat{f}_{l+m} \leftarrow z_0^l q_{l,m}^0 + z_0^{l-1} r_{l,m}^0 + \frac{1}{2} \sum_{n=1}^{m} (z_n^l q_{l,m}^n + z_n^{l-1} r_{l,m}^n)$
```

The optimal stage at which to halt the Driscoll–Healy algorithm and complete the computation using Lemma 2.12 depends on α and β and can be obtained theoretically. The derivative of (2.25) as a function of M equals zero if and only if

$$M \ln^2 2 - 4\alpha \ln M = (2\alpha + 4\beta + 6) \ln 2. \tag{2.26}$$

In our implementation $\alpha = 2.125$ and $\beta = 5$; thus the minimum is $M = 128$. In practice, the optimal choice of M will depend not only on the number of flops performed, but also on the architecture of the machine used. The machine-tuned basic linear algebra subprograms (BLAS) exploit the memory hierarchy of a computer, and using the BLAS may cause a shift in the optimal value for M. For example, early termination can be implemented by using a level 2 BLAS operation for matrix-vector multiplication, which is more efficient than the level 1 vector operations of a straightforward implementation of the Driscoll–Healy algorithm. This will increase the optimal value for M.

3. The basic parallel algorithm and its implementation.

We designed our parallel algorithm using the BSP model, which provides a simple and effective way of developing portable parallel algorithms. The BSP model does not favor any specific computer architecture, and it includes a simple cost function that enables us to choose between algorithms without actually having to implement them.

In the following subsections, we give a brief description of the BSP model, and then we present the framework in which we develop our parallel algorithm, including the data structures and data distributions used. This leads to a basic parallel algorithm. From now on we concentrate on the DLT instead of the more general discrete orthogonal polynomial transform.

3.1. The BSP model. In the BSP model [34], a computer consists of a set of p processors, each with a private memory, connected by a communication network that allows processors to access the memories of other processors. In the model, algorithms consist of a sequence of supersteps. In the variant of the model we use, a *superstep* is either a number of computation steps or a number of communication steps. Global synchronization barriers (i.e., places of the algorithm where all processors must synchronize with each other) precede and/or follow a communication superstep. Using supersteps imposes a sequential structure on parallel algorithms, and this greatly simplifies the design process.

A BSP computer can be characterized by four global parameters: p, the number of processors; s, the computing speed in flop/s; g, the communication time per data element sent or received, measured in flop time units; and l, the synchronization time, also measured in flop time units. Algorithms can be analyzed by using the parameters p, g, and l; the parameter s just scales the time. In this work, we are able to avoid all synchronizations at the end of computation supersteps. Therefore, the time of a computation superstep is simply w, the maximum amount of work (in flops) of any processor. The time of a communication superstep is $hg + l$, where h is the maximum number of data elements sent or received by any processor. The total execution time of an algorithm (in flops) can be obtained by adding the times of the separate supersteps. This yields an expression of the form $a + bg + cl$. For further details and some basic techniques, see [6]. BSPlib [21] is a standard library which enables parallel programming in BSP style. Available implementations are the Oxford BSP toolset [22] and the Paderborn University BSP library [8].

3.2. Data structures and data distributions. At each stage k, $1 \le k \le \log_2 \frac{N}{M}$, the number of intermediate polynomial pairs doubles as the number of expansion coefficients halves. Thus, at every stage of the computation, all the intermediate polynomials can be stored in two arrays of size N. We use an array $\mathbf{f}$ to store the Chebyshev coefficients of the polynomials Z_l^{2K} and an array $\mathbf{g}$ to store the coefficients of Z_{l+1}^{2K} for $l = 0, 2K, \ldots, N - 2K$ with $K = N/2^k$ in stage k. We also need some extra work space to compute the coefficients of the polynomials Z_{l+K}^{2K} and Z_{l+K+1}^{2K}. For this we use two auxiliary arrays, $\mathbf{u}$ and $\mathbf{v}$, of size N.

The data flow of the algorithm (see Figure 3.1) suggests that we distribute all the vectors by blocks, i.e., assign one block of consecutive vector elements to each processor. This works well if p is a power of two, which we will assume from now on. Since both N and p are thus powers of two, each processor obtains exactly N/p elements. For the general case, the block distribution is defined as follows.

Definition 3.1 (block distribution). *Let* $\mathbf{f}$ *be a vector of size* N. *We say that* $\mathbf{f}$ *is* block distributed *over* p *processors if, for all* j, *the element* f_j *is stored in* $\mathrm{Proc}(j \operatorname{div} b)$ *and has local index* $j' = j \bmod b$, *where* $b = \lceil N/p \rceil$ *is the block size.*

The precomputed data required to perform the recurrence of stage k are stored in two-dimensional arrays $\mathbf{Q}$ and $\mathbf{R}$, each of size $2\log_2(N/M) \times N$. Each pair of rows

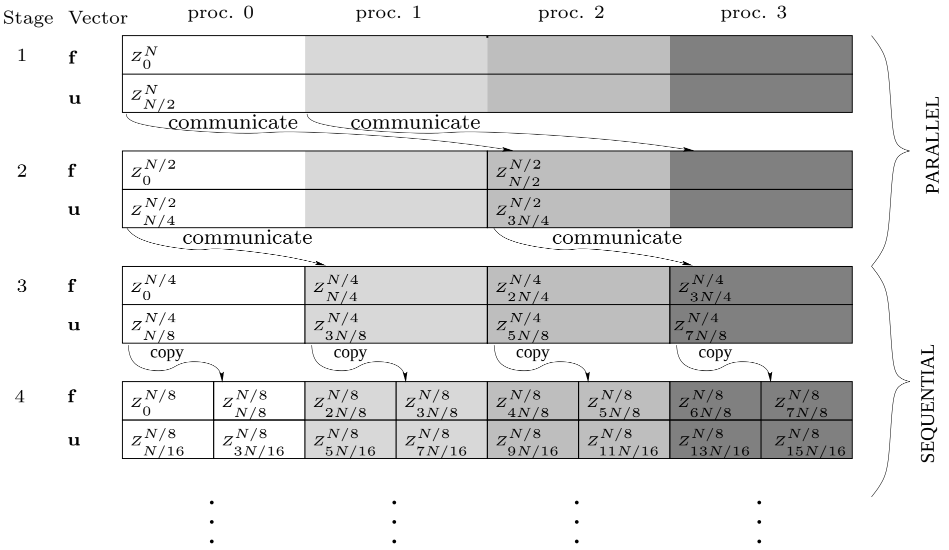


FIG. 3.1. *Main data structure and data distribution in the parallel fast Legendre transform (FLT) algorithm for $p = 4$. Arrays* **f** *and* **g** *contain the Chebyshev coefficients of the polynomials Z_l^{2K} and Z_{l+1}^{2K}, which are already available at the start of the stage. Arrays* **u** *and* **v** *contain Z_{l+K}^{2K} and Z_{l+K+1}^{2K}, which become available at the end of the stage. Arrays* **g** *and* **v** *are not depicted. Each array is divided into four local subarrays by using the block distribution.*

k	$K-1, K$	proc. 0	proc. 1	proc. 2	proc. 3
1	31	$l = 0$			
	32	$j = 0, \dots, 63$			
2	15	$l = 0$		$l = 32$	
	16	$j = 0, \dots, 31$		$j = 0, \dots, 31$	
3	7	$l = 0$	$l = 16$	$l = 32$	$l = 48$
	8	$j = 0, \dots, 15$	$j = 0, \dots, 15$	$j = 0, \dots, 15$	$j = 0, \dots, 15$

FIG. 3.2. *Data structure and distribution of the precomputed data needed in the recurrence with $N = 64$, $M = 8$, and $p = 4$. Data are stored in two-dimensional arrays* **Q** *and* **R**; *one such array is shown. Each pair of rows in an array stores the data needed for one stage k.*

in **Q** stores data needed for one stage k by

$$\begin{aligned} \mathbf{Q}[2k-2, l+j] &= Q_{l+1,K-1}(x_j^{2K}), \\ \mathbf{Q}[2k-1, l+j] &= Q_{l+1,K}(x_j^{2K}) \end{aligned} \tag{3.1}$$

for $l = 0, 2K, \dots, N-2K$, $j = 0, 1, \dots, 2K-1$, where $K = N/2^k$. Thus polynomials $Q_{l+1,K-1}$ are stored in row $2k-2$, and polynomials $Q_{l+1,K}$ are stored in row $2k-1$. This is shown in Figure 3.2. The polynomials $R_{l+1,K-1}$ and $R_{l+1,K}$ are stored in the same way in array **R**. Note that the indexing of the implementation arrays starts at zero. Each row of **Q** and **R** is distributed by the block distribution, i.e., $\mathbf{Q}[i,j], \mathbf{R}[i,j] \in \text{Proc}(j \text{ div } \frac{N}{p})$, so that the recurrence is a local operation.

The termination coefficients $q_{l,m}^n$ and $r_{l,m}^n$ for $l = 1, M+1, 2M+1, \dots, N-M+1$, $m = 1, 2, \dots, M-2$, and $n = 0, 1, \dots, m$ are stored in a two-dimensional array **T** of size $N/M \times (M(M-1)/2 - 1)$. The coefficients for one value of l are stored in row $(l-1)/M$ of **T**. Each row has the same internal structure: the coefficients are stored in increasing order of m, and coefficients with the same m are ordered by increasing n. (This format is commonly used to store lower triangular matrices.) By part 2 of

	$m = 1$	$m = 2$	$m = 3$	$m = 4$	$m = 5$	$m = 6$	
$l = 1$	$r^0\ q^1$	$q^0\ r^1\ q^2$	$r^0\ q^1\ r^2\ q^3$	$q^0\ r^1\ q^2\ r^3\ q^4$	$r^0\ q^1\ r^2\ q^3\ r^4\ q^5$	$q^0\ r^1\ q^2\ r^3\ q^4\ r^5\ q^6$	proc. 0
$l = 9$	$r^0\ q^1$	$q^0\ r^1\ q^2$	$r^0\ q^1\ r^2\ q^3$	$q^0\ r^1\ q^2\ r^3\ q^4$	$r^0\ q^1\ r^2\ q^3\ r^4\ q^5$	$q^0\ r^1\ q^2\ r^3\ q^4\ r^5\ q^6$	
$l = 17$	$r^0\ q^1$	$q^0\ r^1\ q^2$	$r^0\ q^1\ r^2\ q^3$	$q^0\ r^1\ q^2\ r^3\ q^4$	$r^0\ q^1\ r^2\ q^3\ r^4\ q^5$	$q^0\ r^1\ q^2\ r^3\ q^4\ r^5\ q^6$	proc. 1
$l = 25$	$r^0\ q^1$	$q^0\ r^1\ q^2$	$r^0\ q^1\ r^2\ q^3$	$q^0\ r^1\ q^2\ r^3\ q^4$	$r^0\ q^1\ r^2\ q^3\ r^4\ q^5$	$q^0\ r^1\ q^2\ r^3\ q^4\ r^5\ q^6$	
$l = 33$	$r^0\ q^1$	$q^0\ r^1\ q^2$	$r^0\ q^1\ r^2\ q^3$	$q^0\ r^1\ q^2\ r^3\ q^4$	$r^0\ q^1\ r^2\ q^3\ r^4\ q^5$	$q^0\ r^1\ q^2\ r^3\ q^4\ r^5\ q^6$	proc. 2
$l = 41$	$r^0\ q^1$	$q^0\ r^1\ q^2$	$r^0\ q^1\ r^2\ q^3$	$q^0\ r^1\ q^2\ r^3\ q^4$	$r^0\ q^1\ r^2\ q^3\ r^4\ q^5$	$q^0\ r^1\ q^2\ r^3\ q^4\ r^5\ q^6$	
$l = 49$	$r^0\ q^1$	$q^0\ r^1\ q^2$	$r^0\ q^1\ r^2\ q^3$	$q^0\ r^1\ q^2\ r^3\ q^4$	$r^0\ q^1\ r^2\ q^3\ r^4\ q^5$	$q^0\ r^1\ q^2\ r^3\ q^4\ r^5\ q^6$	proc. 3
$l = 57$	$r^0\ q^1$	$q^0\ r^1\ q^2$	$r^0\ q^1\ r^2\ q^3$	$q^0\ r^1\ q^2\ r^3\ q^4$	$r^0\ q^1\ r^2\ q^3\ r^4\ q^5$	$q^0\ r^1\ q^2\ r^3\ q^4\ r^5\ q^6$	

FIG. 3.3. *Data structure and distribution of the precomputed data for termination with $N = 64$, $M = 8$, and $p = 4$. The coefficients $q^n_{l,m}$ and $r^n_{l,m}$ are stored in a two-dimensional array* $\mathbf{T}$. *In the picture, q^n denotes $q^n_{l,m}$, and r^n denotes $r^n_{l,m}$.*

Lemma 2.12, either $q^n_{l,m} = 0$ or $r^n_{l,m} = 0$ for each n and m, so we need to store only the value that can be nonzero. Since this depends on whether $n - m$ is even or odd, we obtain an alternating pattern of $q^n_{l,m}$'s and $r^n_{l,m}$'s. Figure 3.3 illustrates this data structure.

The termination stage can be kept local if $M \leq N/p$. This requires that each row of $\mathbf{T}$ is assigned to one processor, namely, to the processor that holds the subvectors for the corresponding value of l. Each column of $\mathbf{T}$ is in the block distribution, i.e., $\mathbf{T}[i, j] \in \text{Proc}(i \text{ div } \frac{N}{pM})$. As a result, the N/M rows of $\mathbf{T}$ are distributed in consecutive blocks of rows.

3.3. The basic parallel algorithm. To formulate our basic parallel algorithm, we introduce the following conventions and subroutines.

Processor identification. The total number of processors is p. The processor identification number is s with $0 \leq s < p$.

Supersteps. Labels indicate a superstep and its type: (Comp) computation superstep, (Comm) communication superstep, and (CpCm) subroutine containing both computation and communication supersteps. Global synchronizations are stated explicitly. Supersteps inside loops are executed repeatedly, though they are numbered only once.

Indexing. All the indices are global. This means that array elements have a unique index which is independent of the processor that owns it. This enables us to describe variables and gain access to arrays in an unambiguous manner, even though the array is distributed and each processor has only part of it.

Vectors and subroutine calls. All vectors are indicated in boldface. To specify part of a vector we write its first element in boldface, e.g., $\mathbf{f_j}$; the vector size is explicitly written as a parameter.

Communication. Communication between processors is done by using

$$\mathbf{g_j} \leftarrow \text{Put}(pid, n, \mathbf{f_i}).$$

This operation puts n elements of vector $\mathbf{f}$, starting from element i, into processor *pid* and stores them there in vector $\mathbf{g}$ starting from element j.

Copying a vector. The operation

$$\mathbf{g_j} \leftarrow \text{Copy}(n, \mathbf{f_i})$$

denotes the copy of n elements of vector $\mathbf{f}$, starting from element i, to a vector $\mathbf{g}$ starting from element j.

Subroutine name ending in 2. Subroutines with a name ending in 2 perform an operation on two vectors instead of one. For example,

$$(\mathbf{f_i}, \mathbf{g_j}) \leftarrow \text{Copy2}(n, \mathbf{u_k}, \mathbf{v_l})$$

is an abbreviation for

$$\begin{aligned} \mathbf{f_i} &\leftarrow \text{Copy}(n, \mathbf{u_k}) \\ \mathbf{g_j} &\leftarrow \text{Copy}(n, \mathbf{v_l}). \end{aligned}$$

FChT. The subroutine

$$\text{BSP\_FChT}(s_0, s_1, p_1, \mathit{sign}, n, \mathbf{f})$$

replaces the input vector $\mathbf{f}$ of size n by its Chebyshev transform if $\mathit{sign} = 1$, or by its inverse Chebyshev transform if $\mathit{sign} = -1$. A group of p_1 processors starting from processor s_0 work together; s_1 with $0 \leq s_1 < p_1$ denotes the processor number within the group. The original processor number equals $s = s_0 + s_1$. For a group of size $p_1 = 1$, this subroutine reduces to the sequential FChT.

Truncation. The subroutine

$$\mathbf{f} \leftarrow \text{BSP\_Trunc}(s_0, s_1, p_1, n, \mathbf{u})$$

truncates two polynomials of degree less than n which are stored as vectors $\mathbf{f}$ and $\mathbf{u}$ of length n. The subroutine copies the first half of $\mathbf{u}$ into the second half of $\mathbf{f}$. A group of p_1 processors starting from processor s_0 work together, similar to the BSP_FChT operation. For a group of size $p_1 = 1$, the subroutine reduces to a sequential truncation of one or more complete polynomials. In Figure 3.1, the truncation operation is depicted by arrows. Algorithm 3.1 describes subroutine BSP_Trunc2 which carries out two truncation operations simultaneously. In doing so, we save one synchronization.

Algorithm 3.1 Truncation procedure for the FLT.

```
CALL  $(\mathbf{f}, \mathbf{g}) \leftarrow \text{BSP\_Trunc2}(s_0, s_1, p_1, n, \mathbf{u}, \mathbf{v})$.
DESCRIPTION
              if $p_1 = 1$ then
$1^{\text{Comp}}$            Sequential truncation.
                  $(\mathbf{f}_{\frac{\mathbf{n}}{\mathbf{2}}}, \mathbf{g}_{\frac{\mathbf{n}}{\mathbf{2}}}) \leftarrow \text{Copy2}(\frac{n}{2}, \mathbf{u}, \mathbf{v})$
              else
$2^{\text{Comm}}$            Parallel truncation.
                  if $s_1 < \frac{p_1}{2}$ then
                      $(\mathbf{f}_{\mathbf{s_1}\frac{\mathbf{n}}{\mathbf{p_1}}+\frac{\mathbf{n}}{\mathbf{2}}}, \mathbf{g}_{\mathbf{s_1}\frac{\mathbf{n}}{\mathbf{p_1}}+\frac{\mathbf{n}}{\mathbf{2}}}) \leftarrow \text{Put2}(s_0 + s_1 + \frac{p_1}{2}, \frac{n}{p_1}, \mathbf{u}_{\mathbf{s_1}\frac{\mathbf{n}}{\mathbf{p_1}}}, \mathbf{v}_{\mathbf{s_1}\frac{\mathbf{n}}{\mathbf{p_1}}})$
                  Synchronize
```

Algorithm 3.2 Basic parallel algorithm for the FLT.

CALL BSP_FLT$(s, p, N, M, \mathbf{f})$.

ARGUMENTS

s: Processor identification; $0 \le s < p$.
p: Number of processors; p is a power of 2 with $p < N$.
N: Transform size; N is a power of 2 with $N \ge 4$.
M: Termination block size; M is a power of 2 with $M \le \min(N/2, N/p)$.
$\mathbf{f} = (f_0, \dots, f_{N-1})$: Real vector of size N (block distributed).

OUTPUT $\mathbf{f} \leftarrow \hat{\mathbf{f}}$.

DESCRIPTION

```
1^Comp    Stage 1: Initialization.
          for $j = s\frac{N}{p}$ to $(s+1)\frac{N}{p} - 1$ do
              $g_j \leftarrow x_j^N f_j$
              $u_j \leftarrow (\mathbf{Q}[0,j] \cdot x_j^N + \mathbf{R}[0,j]) \cdot f_j$
              $v_j \leftarrow (\mathbf{Q}[1,j] \cdot x_j^N + \mathbf{R}[1,j]) \cdot f_j$
2^CpCm    Stage 1: Chebyshev transform.
          BSP_FChT2$(0, s, p, 1, N, \mathbf{f}, \mathbf{g})$
          BSP_FChT2$(0, s, p, 1, N, \mathbf{u}, \mathbf{v})$
3^CpCm    Stage 1: Truncation.
          $(\mathbf{f}, \mathbf{g}) \leftarrow$ BSP_Trunc2$(0, s, p, N, \mathbf{u}, \mathbf{v})$
          for $k = 2$ to $\log_2 \frac{N}{M}$ do
              $K \leftarrow \frac{N}{2^k}$
              $p_1 \leftarrow \max(\frac{p}{2^{k-1}}, 1)$
              $s_0 \leftarrow (s \text{ div } p_1)p_1$
              $s_1 \leftarrow s \bmod p_1$
4^Comp        Stage $k$: Copy.
              $(\mathbf{u}_{\mathbf{s}\frac{\mathbf{N}}{\mathbf{p}}}, \mathbf{v}_{\mathbf{s}\frac{\mathbf{N}}{\mathbf{p}}}) \leftarrow$ Copy2$(\frac{N}{p}, \mathbf{f}_{\mathbf{s}\frac{\mathbf{N}}{\mathbf{p}}}, \mathbf{g}_{\mathbf{s}\frac{\mathbf{N}}{\mathbf{p}}})$
              for $l = s_0\frac{N}{p}$ to $(s_0 + 1)\frac{N}{p} - \frac{2K}{p_1}$ step $\frac{2K}{p_1}$ do
5^CpCm            Stage $k$: Inverse Chebyshev transform.
                  BSP_FChT2$(s_0, s_1, p_1, -1, 2K, \mathbf{u}_\mathbf{l}, \mathbf{v}_\mathbf{l})$
6^Comp            Stage $k$: Recurrence.
                  for $j = s_1\frac{N}{p}$ to $s_1\frac{N}{p} + \frac{2K}{p_1} - 1$ do
                      $a1 \leftarrow \mathbf{Q}[2k-2, l+j] \cdot v_{l+j} + \mathbf{R}[2k-2, l+j] \cdot u_{l+j}$
                      $a2 \leftarrow \mathbf{Q}[2k-1, l+j] \cdot v_{l+j} + \mathbf{R}[2k-1, l+j] \cdot u_{l+j}$
                      $u_{l+j} \leftarrow a1$
                      $v_{l+j} \leftarrow a2$
7^CpCm            Stage $k$: Chebyshev transform.
                  BSP_FChT2$(s_0, s_1, p_1, 1, 2K, \mathbf{u}_\mathbf{l}, \mathbf{v}_\mathbf{l})$
8^CpCm            Stage $k$: Truncation.
                  $(\mathbf{f}, \mathbf{g}) \leftarrow$ BSP_Trunc2$(s_0, s_1, p_1, 2K, \mathbf{u}_\mathbf{l}, \mathbf{v}_\mathbf{l})$
9^Comp    Stage $\log_2 \frac{N}{M} + 1$: Termination.
          for $l = s\frac{N}{p}$ to $(s+1)\frac{N}{p} - M$ step $M$ do
              $\mathbf{f}_\mathbf{l} \leftarrow$ Terminate$(l, M, \mathbf{f}_\mathbf{l}, \mathbf{g}_\mathbf{l})$
```

The basic parallel algorithm for the FLT is presented as Algorithm 3.2. At each stage $k \le \log_2(N/M)$ of the algorithm, there are 2^{k-1} independent problems. For $k \le \log_2 p$, there are more processors than problems, so the processors will have to work in groups. Each group of $p_1 = p/2^{k-1} > 1$ processors handles one subvector of size $2K$, $K = N/2^k$; each processor handles a block of $2K/p_1 = N/p$ vector components. In this case, the l-loop has only one iteration, namely, $l = s_0 \cdot N/p$, and the j-loop has N/p iterations, starting with $j = s_1 \cdot N/p$, so that the indices $l+j$ start with $(s_0 + s_1)N/p = s \cdot N/p$ and end with $(s_0 + s_1)N/p + N/p - 1 = (s+1)N/p - 1$. Interprocessor communication is needed, but it occurs only in two instances:

- inside the parallel FChTs (in supersteps 2, 5, 7); see section 4;
- at the end of each stage (in supersteps 3, 8).

For $k > \log_2 p$, the length of the subvectors involved becomes $2K \leq N/p$. In that case, $p_1 = 1$, $s_0 = s$, and $s_1 = 0$, and each processor has one or more complete problems to deal with, so that the processors can work independently and without communication. Note that the index l runs only over the local values sN/p, $sN/p+2K$, …, $(s+1)N/p - 2K$ instead of over all values of l.

The original stages 0 and 1 of Algorithm 2.3 are combined into one stage and then performed efficiently as follows. First, in superstep 1, the polynomials Z_1^N, $Z_{N/2}^N$, and $Z_{N/2+1}^N$ are computed directly from the input vector $\mathbf{f}$. This is possible because the point-value representation of $Z_1^N = \mathcal{T}_N(f \cdot P_1) = \mathcal{T}_N(f \cdot x)$ needed by the recurrences is the vector of $f_j \cdot x_j^N, 0 \leq j < N$; see subsection 2.3.1. In superstep 2, polynomials $Z_0^N = \mathbf{f}, Z_1^N = \mathbf{g}, Z_{N/2}^N = \mathbf{u}$, and $Z_{N/2+1}^N = \mathbf{v}$ are transformed to Chebyshev representation; then, in superstep 3, they are truncated to obtain the input for stage 2.

The main loop works as follows. In superstep 4, the polynomials Z_l^{2K}, with $K = N/2^k$ and $l = 0, 2K, \ldots, N-2K$, are copied from the array $\mathbf{f}$ into the auxiliary array $\mathbf{u}$, where they are transformed into the polynomials Z_{l+K}^{2K} in supersteps 5–7. Similarly, the polynomials Z_{l+1}^{2K} are copied from $\mathbf{g}$ into $\mathbf{v}$ and then transformed into the polynomials Z_{l+K+1}^{2K}. Note that $\mathbf{u}$ corresponds to the lower value of l, so that in the recurrence the components of $\mathbf{u}$ must be multiplied by values from $\mathbf{R}$. In superstep 8, all the polynomials are truncated by copying the first K Chebyshev coefficients of Z_{l+K}^{2K} into the memory space of the last K Chebyshev coefficients of Z_l^{2K}. The same happens to polynomials Z_{l+K+1}^{2K} and Z_{l+1}^{2K}.

The termination procedure, superstep 9, is a direct implementation of Lemma 2.12 using the data structure $\mathbf{T}$ described in subsection 3.2. Superstep 9 is a computation superstep, provided the condition $M \leq N/p$ is satisfied. This usually holds for the desired termination block size M. In certain situations, however, one would like to terminate even earlier, with a block size larger than N/p. This extension is discussed in [23].

4. Improvements of the parallel algorithm.

4.1. FChT of two vectors, FChT2. The efficiency of the FLT algorithm depends strongly on the FCT algorithm used to perform the Chebyshev transform. There exists a substantial amount of literature on this topic and many implementations of sequential FCTs are available; see, e.g., [1, 29, 30, 33]. Parallel algorithms or implementations have been less intensively studied; see [31] for a recent discussion.

In the FLT algorithm, the Chebyshev transforms always come in pairs, which led us to develop an algorithm that computes two Chebyshev transforms at the same time. The new algorithm is based on the FCT algorithm given by Van Loan [35, Algorithm 4.4.6] and the standard algorithm for computing the FFTs of two real input vectors at the same time (see, e.g., [29]). The new algorithm employs a complex FFT, which is advantageous in the sequential case because as a separate module the complex FFT can easily be replaced, for instance, by a newer, more efficient FFT. Even in the parallel case, where the parallel FFT module needs to be modified to reduce the communication cost of the FLT, we can still make use of the techniques developed for the parallel complex FFT and reuse parts of the FFT code; see subsection 5.4.

The Chebyshev transform is computed as follows. Let $\mathbf{x}$ and $\mathbf{y}$ be the input vectors of length N. We view $\mathbf{x}$ and $\mathbf{y}$ as the real and imaginary parts of a complex vector $(\mathbf{x} + i\,\mathbf{y})$. The algorithm has three phases. Phase 1, the packing of the input data into an auxiliary complex vector $\mathbf{z}$ of length N, is a simple permutation:

$$\left\{ \begin{array}{ll} z_j & = (x_{2j} + i\, y_{2j}), \\ z_{N-j-1} & = (x_{2j+1} + i\, y_{2j+1}), \qquad 0 \le j < N/2. \end{array} \right. \tag{4.1}$$

In phase 2, the complex FFT creates a complex vector $\mathbf{Z}$ of length N:

$$Z_k = \sum_{j=0}^{N-1} z_j e^{\frac{2\pi ijk}{N}}, \qquad 0 \le k < N. \tag{4.2}$$

This phase takes $4.25N \log_2 N$ flops if we use a radix-4 algorithm [35]. Finally, in phase 3 we obtain the Chebyshev transform by

$$(\tilde{x}_k + i\tilde{y}_k) = \frac{\epsilon_k}{2N}(e^{\frac{\pi ik}{2N}} Z_k + e^{-\frac{\pi ik}{2N}} Z_{N-k}), \qquad 0 \le k < N. \tag{4.3}$$

The value Z_N in (4.3) is defined as $Z_N = Z_0$ by periodic extension of (4.2). Phase 3 is efficiently performed by computing the components k and $N-k$ together and using symmetry properties. The cost of phase 3 is $10N$ flops. The total cost of the FChT2 algorithm is thus $4.25N \log_2 N + 10N$, giving an average $\alpha = 2.125$ and $\beta = 5$ for a single transform.

The verification that (4.1)–(4.3) indeed produce the Chebyshev transform is best made in two steps. First, we prove that

$$e^{\frac{\pi ik}{2N}} Z_k = \sum_{j=0}^{N/2-1} [(x_{2j} + iy_{2j})e^{\frac{\pi ik(4j+1)}{2N}} + (x_{2j+1} + iy_{2j+1})e^{-\frac{\pi ik(4j+3)}{2N}}], \tag{4.4}$$

and

$$e^{-\frac{\pi ik}{2N}} Z_{N-k} = \sum_{j=0}^{N/2-1} [(x_{2j} + iy_{2j})e^{-\frac{\pi ik(4j+1)}{2N}} + (x_{2j+1} + iy_{2j+1})e^{\frac{\pi ik(4j+3)}{2N}}]. \tag{4.5}$$

Second, we add (4.4) to (4.5) and multiply the result by $\frac{\epsilon_k}{2N}$ to obtain the desired equality (2.7).

The inverse Chebyshev transform is obtained by inverting the procedure described above. The phases are performed in the reverse order, and the operation of each phase is replaced by its inverse. The cost of the inverse FChT algorithm is the same as that of the FChT algorithm.

Efficient parallelization of this algorithm requires breaking open the parallel FFT inside the FChT2 and merging parts of the FFT with the surrounding computations. We explain this process in the following subsection.

4.2. Parallel FFT within the scope of the parallel FChT2. The FFT is a well-known method for computing the discrete Fourier transform (4.2) of a complex vector of length N in $O(N \log N)$ operations. It can concisely be written as a decomposition of the Fourier matrix F_N,

$$F_N = A_N \cdots A_8 A_4 A_2 P_N, \tag{4.6}$$

where F_N is an $N \times N$ complex matrix, P_N is an $N \times N$ permutation matrix corresponding to the so-called *bit reversal permutation*, and the $N \times N$ matrices A_L are defined by

$$A_L = I_{N/L} \otimes B_L, \quad L = 2, 4, 8, \ldots, N, \tag{4.7}$$

which is shorthand for a block-diagonal matrix $\mathrm{diag}(B_L, \ldots, B_L)$ with N/L copies of the $L \times L$ matrix B_L on the diagonal. The matrix B_L is known as the $L \times L$ *butterfly matrix.*

This matrix decomposition naturally leads to the *radix*-2 *FFT* algorithm [11, 35]. In a radix-2 FFT of size N, the input vector $\mathbf{z}$ is permuted by P_N and then multiplied successively by all the matrices A_L. The multiplications are carried out in $\log_2 N$ stages, each with N/L times a butterfly computation. One butterfly computation modifies $L/2$ pairs $(z_j, z_{j+L/2})$ at distance $L/2$ by adding a multiple of $z_{j+L/2}$ to z_j and subtracting the same multiple.

Parallel radix-2 FFTs have already been discussed in the literature; see, e.g., [25]. For simplicity, we restrict ourselves in our exposition to FFT algorithms where $p \leq \sqrt{N}$. This class of algorithms uses the block distribution to perform the short distance butterflies with $L \leq N/p$ and the cyclic distribution to perform the long distance butterflies with $L > N/p$. Figure 4.1(a) gives an example of the cyclic distribution, which is defined as follows.

Definition 4.1 (cyclic distribution). *Let* $\mathbf{z}$ *be a vector of size* N. *We say that* $\mathbf{z}$ *is* cyclically distributed *over* p *processors if, for all* j, *the element* z_j *is stored in* $\mathrm{Proc}(j \bmod p)$ *and has local index* $j' = j \operatorname{div} p$.

Using such a parallel FFT algorithm, we obtain a basic parallel FChT2 algorithm for two vectors $\mathbf{x}$ and $\mathbf{y}$ of size N.

1. PACK vectors $\mathbf{x}$ and $\mathbf{y}$ as the auxiliary complex vector $\mathbf{z}$ by permuting them, using (4.1).
2. TRANSFORM vector $\mathbf{z}$ using an FFT of size N.
   (a) Perform a bit reversal permutation in $\mathbf{z}$.
   (b) Perform the short distance butterflies of size $L = 2, 4, \ldots, N/p$.
   (c) Permute $\mathbf{z}$ to the cyclic distribution.
   (d) Perform the long distance butterflies of size $L = 2N/p, 4N/p, \ldots, N$.
   (e) Permute $\mathbf{z}$ to the block distribution.
3. EXTRACT the transforms from vector $\mathbf{z}$ and store them in vectors $\mathbf{x}$ and $\mathbf{y}$.
   (a) Permute $\mathbf{z}$ to put components j and $N - j$ in the same processor.
   (b) Compute the new values of $\mathbf{z}$ using (4.3).
   (c) Permute $\mathbf{z}$ to block distribution, and store the result in vectors $\mathbf{x}$ and $\mathbf{y}$.

The time complexity of this basic algorithm will be reduced by a sequence of improvements as detailed in the following subsections.

4.2.1. Combining permutations. By breaking open the FFT phase inside the parallel FChT2 algorithm, we can combine the packing permutation (1) and the bit reversal (2(a)), thus saving one complete permutation of BSP cost $2\frac{N}{p}g + l$. The same can be done for (2(e)) and (3(a)).

4.2.2. Increasing the symmetry of the cyclic distribution. We can eliminate permutation (2(e))/(3(a)) completely by restricting the number of processors slightly further to $p \leq \sqrt{N/2}$ and permuting the vector $\mathbf{z}$ in phase (2(e)) from block distribution to a slightly modified cyclic distribution, the zig-zag cyclic distribution, shown in Figure 4.1(b) and defined as follows.

Definition 4.2 (zig-zag cyclic distribution). *Let* $\mathbf{z}$ *be a vector of size* N. *We say that* $\mathbf{z}$ *is* zig-zag cyclically distributed *over* p *processors if, for all* j, *the element* z_j *is stored in* $\mathrm{Proc}(j \bmod p)$ *if* $j \bmod 2p < p$ *and in* $\mathrm{Proc}(-j \bmod p)$ *otherwise, and has local index* $j' = j \operatorname{div} p$.

In this distribution, both the components j and $j + L/2$ needed by the butterfly

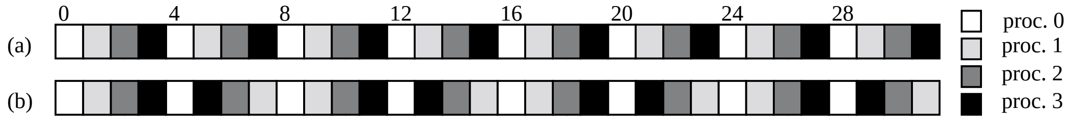


FIG. 4.1. (a) *Cyclic distribution and* (b) *zig-zag cyclic distribution for a vector of size* 32 *distributed over four processors.*

operations with $L > N/p$ and the components j and $N - j$ needed by the extract operation are in the same processor; thus we avoid the permutation (2(e))/(3(a)) above, saving another $2\frac{N}{p}g + l$ in BSP costs.

4.2.3. Reversing the stages for the inverse FFT. To be able to apply the same ideas to the inverse transform we perform the inverse FFT by reversing the stages of the FFT and inverting the butterflies, instead of taking the more common approach of using the same FFT algorithm but replacing the powers of $e^{\frac{2\pi i}{N}}$ by their conjugates. Thus we save $6\frac{N}{p}g + 3l$, both in the Chebyshev transform and its inverse.

4.2.4. Reducing the number of flops. Wherever possible we take pairs of stages $A_{2L}A_L$ together and perform them as one operation. The butterflies have the form $B_{2L}(I_2 \otimes B_L)$, which is a $2L \times 2L$ matrix consisting of 4×4 blocks, each an $L/2 \times L/2$ diagonal submatrix. This matrix is a symmetrically permuted version of the radix-4 butterfly matrix [35]. This approach gives both the efficiency of a radix-4 FFT algorithm and the flexibility of treating the parallel FFT within the radix-2 framework; for example, it is possible to redistribute the data after any number of stages and not only after an even number. The radix-4 approach reduces α from 2.5 to 2.125. An additional benefit of radix-4 butterflies is better use of the cache memory: 34 flops are performed on a quadruple of data, instead of 10 flops on a pair of data. Thus 8.5 flops are carried out per data word loaded into the cache, instead of five flops. This effect may be even more important than the reduction in flop count.

Since we do not use the upper half of the Chebyshev coefficients computed in the forward transform, we can alter the algorithm to avoid computing them. This saves $4N$ flops in (4.3).

4.3. Main loop. The discussion that follows is only relevant in the parallel part of the main loop, i.e., stages $k \le \log_2 p$, so we will restrict ourselves to these stages. Recall that in these stages a group of $p_1 = p/2^{k-1} > 1$ processors handles only one subproblem of size $2K = 2N/2^k$ corresponding to $l = s_0\frac{N}{p}$.

4.3.1. Modifying the truncation/copy operation. It is possible to reorganize the main loop of the FLT algorithm such that the end of stage k and the start of stage $k+1$ are merged into one, more efficient procedure. The current sequence of operations is as follows.

1. Permute from zig-zag cyclic to block distribution in stage k.
2. Truncate at the end of stage k.
3. Copy at the beginning of stage $k + 1$.
4. Permute from block to zig-zag cyclic distribution in stage $k + 1$.

In the new approach, we aim at removing permutations 1 and 4 by keeping the data in the zig-zag cyclic distribution of stage k during part of stage $k + 1$ as well. In the following discussion, we treat only operations on the arrays $\mathbf{f}$ and $\mathbf{u}$ because the operations on $\mathbf{g}$ and $\mathbf{v}$ are similar. The values of K and p_1 used in the discussion are those of stage k. Now assume that the second half of the $2K$ elements of $\mathbf{f_l}$ has been discarded. Recall that the second half of $\mathbf{u_l}$ has not even been computed; see subsection 4.2.4. Therefore, $\mathbf{f_l}$ and $\mathbf{u_l}$ are now in the zig-zag cyclic distribution of K elements (instead of $2K$) over p_1 processors. The new sequence of operations, illustrated in Figure 4.2, is as follows.

1. Prepare a working copy of the data needed at stage $k+1$.
   - (a) Copy vector $\mathbf{u_l}$ of size K into vector $\mathbf{u_{l+K}}$.
   - (b) Copy vector $\mathbf{f_l}$ of size K into vector $\mathbf{u_l}$.
2. Redistribute the data needed at stage $k+2$.
   - (a) Put the first $K/2$ elements of vector $\mathbf{u_l}$ into vector $\mathbf{f_l}$ in the zig-zag cyclic distribution over the first $\frac{p_1}{2}$ processors.
   - (b) Put the first $K/2$ elements of vector $\mathbf{u_{l+K}}$ into vector $\mathbf{f_{l+K}}$ in the zig-zag cyclic distribution over the next $\frac{p_1}{2}$ processors.

The synchronization at the end of the redistribution can be removed by buffering and delaying the communications until the next synchronization. The new approach reduces the BSP cost of the truncation/copy operation from $6\frac{N}{p}g + 3l$ to $\frac{N}{p}g$.

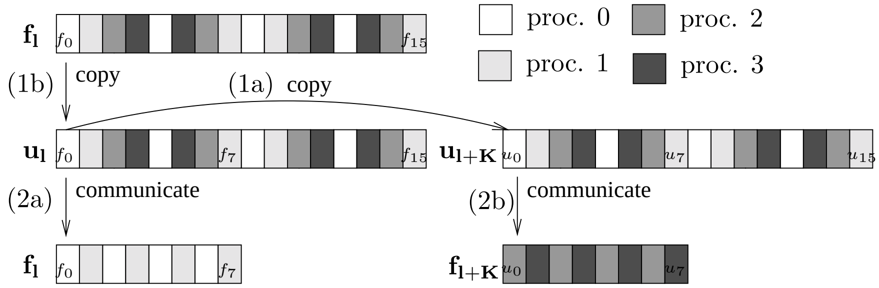


FIG. 4.2. *New truncation/copy operation of vectors* $\mathbf{f_l}$ *and* $\mathbf{u_l}$ *for* $K = 16$ *and* $p_1 = 4$.

As a result, vectors $\mathbf{u_l}$ and $\mathbf{u_{l+K}}$ contain all the data needed at stage $k+1$, and vectors $\mathbf{f_l}$ and $\mathbf{f_{l+K}}$ contain half the data needed at stage $k+2$; stage $k+1$ will produce the other half. We now verify that the operations on $\mathbf{u_l}$ and $\mathbf{u_{l+K}}$ at the start of stage $k+1$ remain local and hence do not require communication. The first operation is the inverse of operation (4.3), which acts on an array of size K. The pairs (u_{l+j}, u_{l+K-j}) and (u_{l+K+j}, u_{l+2K-j}) involved in this operation are indeed local. After that, the long distance butterflies of the inverse FFT have to be performed, and then the short distance ones have to be performed. The short distance butterflies, of size $L \leq K/(p_1/2)$, will be done after a suitable redistribution as in the original algorithm, using the block distribution over $p_1/2$ processors. The long distance butterflies operate on pairs $(u_{l+j}, u_{l+j+L/2})$ and $(u_{l+K+j}, u_{l+K+j+L/2})$ with $L \geq 4K/p_1$. The restriction $p \leq \sqrt{N/2}$ implies $p_1 \leq \sqrt{K}$ and hence $2p_1 \leq 2K/p_1$, which means that the period of the zig-zag cyclic distribution over p_1 processors does not exceed the minimum butterfly distance. As a result, the pairs involved are local.

4.3.2. Moving the bit reversal to the precomputation. Another major improvement is to avoid the packing/bit reversal permutation (1)/(2(a)) in the FChT2 just following the recurrence and its inverse preceding the recurrence, thus saving another $4\frac{N}{p}g + 2l$ in communication costs. This is done by storing the recurrence coefficients permuted by the packing/bit reversal permutation. This works because one permutation is the inverse of the other, so that the auxiliary vector $\mathbf{z}$ is in the same ordering immediately before and after the permutations.

4.4. Time complexity. After all the improvements, the total communication and synchronization cost is approximately $(5\frac{N}{p}\log_2 p)g + (2\log_2 p)l$. Only two communication supersteps remain: the zig-zag cyclic to block redistribution inside the inverse FFT, which can be combined with the redistribution of the truncation, and the block to zig-zag cyclic redistribution inside the FFT. To obtain this complexity, we ignored lower order terms and special cases occurring at the start and the end of the algorithm.

The total cost of the optimized algorithm without early termination is

$$T_{\mathrm{FLT,par}} \approx 4.25\frac{N}{p}\log_2^2 N + \left(5\frac{N}{p}\log_2 p\right) g + (2\log_2 p)\, l. \tag{4.8}$$

5. Experimental results and discussion. In this section, we present results on the accuracy and scalability of the implementation of the Legendre transform algorithm. We implemented the algorithm in ANSI C using the BSPlib library [21]. Our programs are completely self-contained, and we did not rely on any system-provided numerical software such as BLAS, FFTs, etc. Nonetheless, we used an optimized FFT package [27] to illustrate how to optimize the computation supersteps of the code; see section 5.4. We tested our programs using the Oxford BSP toolset [22] implementation of BSPlib running on two different machines:

1. a *Cray T3E* with up to 64 processors, each having a theoretical peak speed of 600 Mflop/s, with double precision (64-bit) accuracy of 1.0×10^{-15};
2. an *IBM RS/6000 SP* with up to 8 processors, each having a theoretical peak speed of 640 Mflop/s, which uses the more common IEEE 754 floating point arithmetic with double precision accuracy of 2.2×10^{-16}.

To make a consistent comparison of the results, we compiled all test programs using the `bspfront` driver with options `-O3 -flibrary-level 2 -bspfifo 10000 -fcombine-puts` (on the IBM we had to add the option `-fcombine-puts-buffer 256K,8M,256K`) and measured the elapsed execution times on exclusively dedicated CPUs using the system clock.

5.1. Accuracy. We tested the accuracy of our implementation by measuring the error obtained when transforming a random input vector $\mathbf{f}$ with elements uniformly distributed between 0 and 1. The relative error is defined as $||\hat{\mathbf{f}}^* - \hat{\mathbf{f}}||_2/||\hat{\mathbf{f}}||_2$, where $\hat{\mathbf{f}}^*$ is the FLT and $\hat{\mathbf{f}}$ is the exact DLT (computed by (2.9), using the stable three-term recurrence (2.10) and quadruple precision); $||\cdot||_2$ indicates the L^2-norm.

Table 5.1 shows the relative errors of the sequential algorithm for various problem sizes using double precision except in the precomputation of the last column, which is carried out in quadruple precision. This could not be done for the Cray T3E because quadruple precision is not available there. Note, however, that it is possible to precompute the values on another computer. The results show that the error of the FLT algorithm is comparable with the error of the DLT provided that the precomputed values are accurate. Therefore, it is best to perform the precomputation in increased precision. This can be done at little extra cost because the precomputation is done only once and its cost can be amortized over many FLTs. See [19, 20] for a discussion of other techniques that can be used to obtain more accurate results.

The errors of the parallel implementation are of the same order as in the sequential case. The only part of the parallel implementation that differs from the sequential implementation in this respect is the FFT, and then only if the butterfly stages cannot be paired in the same way. Varying the termination block size between 2 and 128 also does not significantly change the magnitude of the error.

Table 5.1

Relative errors for the sequential FLT algorithm. (QP indicates that the precomputation is carried out in quadruple precision.)

N	Cray T3E		IBM SP (IEEE 754)		
	DLT	FLT	DLT	FLT	FLT-QP
512	7.0×10^{-14}	1.4×10^{-12}	7.7×10^{-14}	4.3×10^{-12}	1.5×10^{-14}
1024	3.5×10^{-13}	2.1×10^{-11}	3.0×10^{-13}	3.1×10^{-11}	2.3×10^{-13}
8192	1.2×10^{-11}	5.4×10^{-9}	1.3×10^{-11}	3.5×10^{-9}	1.3×10^{-11}
65536	2.7×10^{-10}	5.5×10^{-7}	2.7×10^{-10}	9.4×10^{-8}	1.6×10^{-10}

Table 5.2

Execution time (in ms) of various sequential Legendre transform algorithms.

	Cray T3E				IBM SP			
N	DLT	FLT	FLT	FLT	DLT	FLT	FLT	FLT
		$M = \frac{N}{2}$	$M = 64$	$M = 2$		$M = \frac{N}{2}$	$M = 64$	$M = 2$
16	0.013	0.028	−−	0.078	0.018	0.016	−−	0.041
32	0.050	0.053	−−	0.187	0.046	0.030	−−	0.100
64	0.219	0.114	−−	0.436	0.182	0.069	−−	0.239
128	1.199	0.328	0.328	1.027	0.729	0.186	0.186	0.560
256	5.847	1.394	1.123	2.576	3.109	0.612	0.549	1.297
512	23.497	5.712	3.340	6.034	12.483	2.231	1.568	3.013
1024	93.702	21.559	8.525	14.147	49.917	8.141	3.966	6.889

5.2. Efficiency of the sequential implementation. We measured the efficiency of our optimized FLT algorithm by comparing its execution time with the execution time of the direct DLT algorithm (i.e., a matrix-vector multiplication). Table 5.2 shows the times obtained by the direct algorithm and the FLT with various termination values: $M = N/2$ is the maximum termination value that our program can handle, and the resulting algorithm is similar to the seminaive algorithm [12]; $M = 64$ is the empirically determined value that makes the algorithm perform best for $N \leq 8192$ on the Cray T3E and for $N \leq 16384$ on the IBM SP (for larger values of N, the choice $M = 128$ gives slightly better results); $M = 2$ yields the pure FLT algorithm without early termination.

The results show that for the choice $M = N/2$ the behavior of the FLT is similar to that of the DLT. On the other extreme, the pure FLT algorithm with $M = 2$ becomes faster than the DLT algorithm at $N = 128$. Choosing the optimum value $M = 64$ further improves the performance of the FLT. This optimum is close to the theoretical optimum $M = 128$ calculated in section 2.4.

Another advantage of the FLT is its reduced need of storage space for the precomputed data. While the DLT must store $O(N^2)$ precomputed values, the FLT needs to store only $O(N \log N)$ precomputed values. Furthermore, these values can be computed in only $O(N \log^2 N)$ operations using Algorithm B.1, which is presented in Appendix B. Table 5.3 lists the storage and precomputation requirements for the DLT and the FLT for various input sizes, assuming a precomputation cost of $4N(N-2)$ for the direct transform and $12.75N \log_2^2 N + 76.25N \log_2 N$ for the fast transform; cf. (B.3).

5.3. Scalability of the parallel implementation. We tested the scalability of our optimized parallel implementation using our optimized sequential implementation as basis for comparison.

Tables 5.4 and 5.5 show the execution times on the Cray T3E for up to 64 proces-

Table 5.3

Storage and precomputation requirements for the direct and FLT algorithms.

	Storage space (in words)		Precomputation cost (in flops)	
N	DLT	FLT ($M = 2$)	DLT	FLT ($M = 2$)
16	256	96	896	8,144
64	4,096	640	15,872	58,656
256	65,536	3,584	260,096	365,056
1024	1,048,576	18,432	4,186,112	2,086,400

Table 5.4

Execution times (in ms) for the FLT algorithm on a Cray T3E for $M = 2$ (top) and $M = 64$ (bottom).

N	seq	$p = 1$	$p = 2$	$p = 4$	$p = 8$	$p = 16$	$p = 32$	$p = 64$
512	6.04	6.30	3.65	2.36	1.82	2.16	−−	−−
1024	14.15	14.42	8.18	4.19	3.23	2.71	−−	−−
8192	206.61	205.30	103.47	52.96	27.66	15.61	9.48	9.10
65536	4515.10	4518.30	2325.80	1180.00	569.47	244.02	126.74	67.17
512	3.37	3.58	2.29	1.67	−−	−−	−−	−−
1024	8.53	8.84	5.34	3.28	2.47	−−	−−	−−
8192	151.49	155.10	77.50	40.64	21.69	12.93	8.48	8.26
65536	3670.10	3732.80	1932.00	970.58	469.07	194.77	102.26	54.60

sors and the IBM SP for up to eight processors, respectively. The tables list timing results for the sequential and parallel algorithms with $p < \sqrt{N}$ and $M = 2, 64$. The table for the Cray starts at a lower value of N because on the Cray parallelism is already advantageous for much smaller problem sizes. In general, the Cray T3E delivers better scalability than the IBM SP, but the execution on the IBM SP is faster. Qualitatively, this is in accordance with the BSP parameters for these machines given in Table 5.6: the parameters g and l are smaller for the Cray T3E, which means relatively faster communication and synchronization, whereas the parameter s is larger for the IBM SP, which means faster computation.

The BSP parameters presented in Table 5.6 reflect the way we implemented the communication subroutines of our programs. Our implementations divide the communication supersteps into three phases. In phase 1, the data are locally rearranged in such a way that the elements to be sent to the same processor are packed together. In phase 2, packets of data are exchanged between the processors. In phase 3, the data are unpacked locally so that the elements get to their final destination. This scheme generally saves communication time for regular communication patterns, because the overhead of sending corresponding address information together with the actual data is drastically reduced. Furthermore, we implemented the communication subroutines using `hpputs` (high performance put operations that dispense with the use of buffers). Therefore, the l and g values of Table 5.6 were measured for large data packets sent by using `hpputs`. For more details, see [23, Appendix A].

Figure 5.1 shows the behavior of our algorithm in terms of flop rate per processor:

$$F(N,p) = \frac{4.25N \log_2^2 N + 34.5N \log_2 N}{p \cdot T(N,p)}. \tag{5.1}$$

Here, the numerator represents the number of flops of the basic sequential FLT algorithm with only the two main terms included and without optimizations such as early termination. This gives a convenient reference count for the FLT, similar to the

Table 5.5
Execution times (in ms) for the FLT algorithm on an IBM SP for $M = 2$ (top) and $M = 64$ (bottom).

N	seq	$p = 1$	$p = 2$	$p = 4$	$p = 8$
8192	83.71	85.36	60.51	53.96	73.26
16384	233.89	242.91	134.45	99.67	102.77
32768	840.56	865.57	449.76	229.25	187.90
65536	2159.20	2201.80	1193.90	625.40	353.87
8192	56.85	57.81	48.18	47.30	70.93
16384	176.97	179.74	107.79	83.52	96.70
32768	640.49	651.51	359.49	187.04	168.59
65536	1729.30	1747.60	1016.10	540.95	314.75

Table 5.6
BSP parameters measured using a modified version of the program **bspbench** *from the package* **BSPEDUpack** [7].

p	Cray T3E ($s = 34.9$ Mflop/s)				IBM SP ($s = 202$ Mflop/s)			
	g		l		g		l	
	(flops)	(μs)	(flops)	(μs)	(flops)	(μs)	(flops)	(μs)
2	1.14	0.0328	479	13.72	82.2	0.407	215203	1066
8	2.14	0.0613	1377	39.48	92.0	0.456	868640	4300
64	3.05	0.0876	3861	110.88	——	——	——	——

common count of $5N \log_2 N$ for the FFT. Furthermore, $T(N, p)$ denotes the execution time of the parallel FLT algorithm. Ideally, the flop rates should be high and remain constant with an increase in p. As already pointed out, in absolute numbers, the IBM SP delivers more Mflops per second, but the Cray T3E maintains better flop rates as a function of p.

Normalizing $F(N, p)$ against the flop rate of the sequential algorithm, $F^{\text{seq}}(N)$ (the entry labeled "seq" in Figure 5.1), gives the absolute efficiency of a parallel algorithm:

$$E^{\text{abs}}(N, p) = \frac{F(N, p)}{F^{\text{seq}}(N)}. \tag{5.2}$$

The absolute efficiency can be used as a measure of the scalability of the parallel algorithm. Ideally, $E^{\text{abs}}(N, p) = 1$. As a rule of thumb, efficiencies larger than 0.8 are considered very good, while efficiencies of 0.5 are reasonable. From the nearly horizontal lines of $F(N, p)$ for large N, it is clear that the algorithm scales very well asymptotically (i.e., when N is large). The experimental results also show that on the Cray T3E with up to 64 processors a size of $N \geq 8192$ already gives reasonable to very good efficiencies, whereas on the IBM SP with up to eight processors, N must at least be equal to 32768. Note that on the Cray T3E, efficiencies larger than one are observed for $N \geq 16384$. This is a well-known phenomenon related to cache size.

The DLT is often used as part of a larger spherical harmonic transform. This means that, in practical applications, many independent FLTs of small size will be performed by a group of processors that could be only slightly larger than the number of transforms. For this reason, it is important that the FLT algorithm scales well for small N and p. Indeed, our algorithm already delivers reasonable to good efficiencies on the Cray T3E with up to eight processors for N as small as 512; on the IBM SP, larger problem sizes are needed.

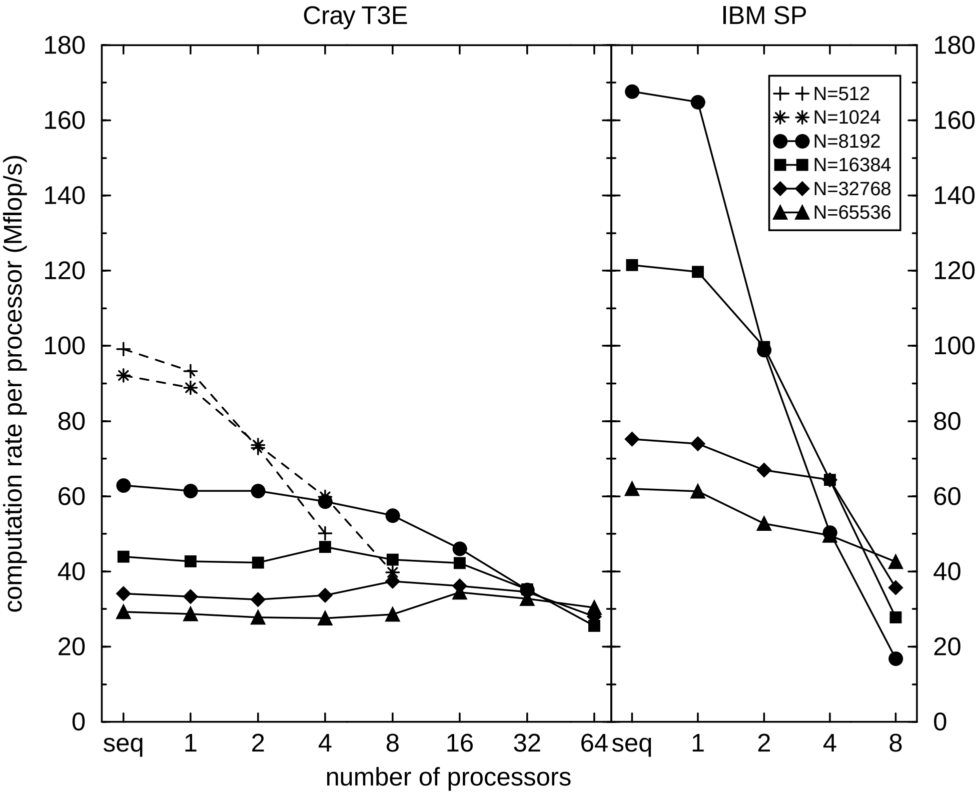


Fig. 5.1. *Mflop/s rate per processor of the FLT algorithm with termination parameter $M = 64$.*

5.4. Further optimizations. It is easy to modify our FLT algorithm to enable the use of complete FFTs instead of our own butterfly routines. In this subsection, we discuss the necessary modifications and demonstrate the possible gains by comparing our plain implementation with an optimized version that uses Ooura's FFT package `fft4g.c` [27]. This competitive FFT is on average 2.6 times faster than our butterflies and our FFT.

The sequential stages of our parallel algorithm involve complete FFTs, provided we do not move the bit-reversal permutation to the precomputation; see section 4.3.2. These FFTs can readily be replaced by highly optimized versions. Similarly, the short distance butterflies of the parallel stages can be replaced by a local bit-reversal permutation followed by a complete local FFT of size N/p. In stages 2 to $\log_2 p$, the extra permutation can be moved to the precomputation, so that it comes for free. (It is even possible to skip the extra permutation of the first stage; see [23, Section 2.3.2].)

Figure 5.2 shows the effect on the Cray T3E of optimizing the computation supersteps of the FLT algorithm. For small N, the main savings are already obtained by terminating early, while for large N they are achieved by using complete, faster FFTs. We also observe that computation is dominant for large values of N/p, which leaves in principle plenty of room for improvement of the FLT by optimizing the computation supersteps. For small N/p, however, such optimization will not have much

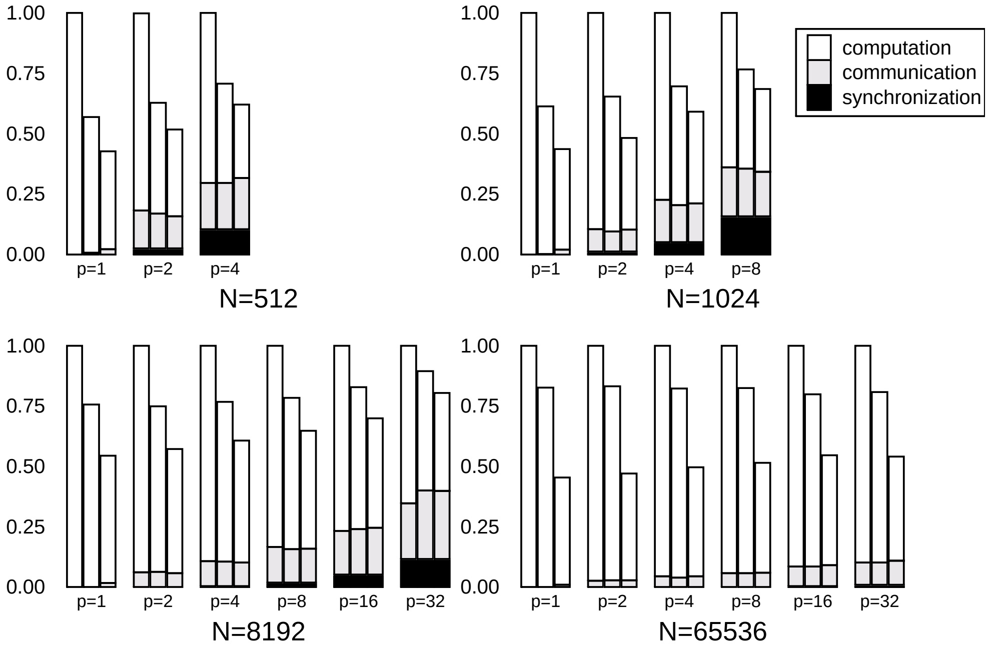


FIG. 5.2. *Gains achieved by optimizing the computational supersteps of the parallel FLT. Vertical bars represent the time of an FLT of length N executed on p processors of a Cray T3E, normalized with respect to our plain implementation with $M = 2$. The first bar of each triple represents the plain implementation with $M = 2$; the second bar represents the plain implementation optimized by early termination with $M = 64$; and the third bar represents the highly optimized version (with $M = 64$ and Ooura's FFT). Each bar is split into three parts, representing computation, communication, and synchronization time.*

effect because the dominant cost is that of communication and synchronization; we believe that these costs have already been reduced to the minimum.

Optimizing the FFTs of the sequential stages and the short distance butterflies of the parallel stages already covers most of the $O(N \log^2 N)$ computation operations of the algorithm. The only parts not yet optimized are as follows: the recurrence and pack/extract operations of the FChTs, with a total of $O(N \log N)$ flops, which can be carried out using level 1 BLAS; the $O(NM)$ termination routine which can be based on level 2 BLAS; and the long distance butterflies with a total of $O(N \log^2 p)$ flops. Since $p \ll N/p$ in practice, the long distance butterfly stages have a relatively small cost; if, however, $p \approx N/p$, this cost becomes significant. Fortunately, the long distance butterfly stages in the zig-zag cyclic distribution can also be carried out using FFTs at the expense of an extra $O(N \log p)$ flops and some local permutations. To do so, we first need to permute the vector to move the even elements to the front and then apply to both vector halves the method described in [23, sections 2.4 and 2.6], which performs cyclically distributed long distance butterflies using FFTs.

6. Conclusions and future work. As part of this work, we developed and implemented a sequential algorithm for the DLT, based on the Driscoll–Healy algorithm. This implementation is competitive for large problem sizes. Its complexity $O(N \log^2 N)$ is considerably lower than the $O(N^2)$ matrix-vector multiplication algorithms which are still much in use today for the computation of Legendre transforms.

Its accuracy is similar, provided the precomputation is performed in increased precision. The new algorithm is a promising approach for compute-intensive applications such as weather forecasting.

The main aim of this work was to develop and implement a parallel Legendre transform algorithm. Our experimental results show that the performance of our parallel algorithm scales well with the number of processors for medium to large problem sizes. The overhead of our parallel program consists mainly of communication, and this is limited to two redistributions of the full data set and one redistribution of half the set in each of the first $\log_2 p$ stages of the algorithm. Two full redistributions are already required by an FFT and an inverse FFT, indicating that our result is close to optimal. Our parallelization approach was first to derive a basic algorithm that uses block and cyclic data distributions and then to optimize this algorithm by removing permutations and redistributions wherever possible. To facilitate this we proposed a new data distribution, which we call the zig-zag cyclic distribution.

Within the framework of this work, we also developed a new algorithm for the simultaneous computation of two Chebyshev transforms. This is useful in the context of the FLT because the Chebyshev transforms always come in pairs, but such a double FChT also has many applications in its own right, as does the corresponding double FCT. Our algorithm has the additional benefit of easy parallelization. Our FFT, FChT, and FLT programs will be made available in the public domain as the package BSPFTpack, which can be obtained through the same Webpage as BSPEDUpack [7].

We view the present FLT as a good starting point for the use of fast Legendre algorithms in practical applications. However, to make our FLT algorithm directly useful in such applications, further work must be done: an inverse FLT must be developed; the FLT must be adapted to the more general case of the spherical harmonic transform where associated Legendre functions are used (this can be done by changing the initial values of the recurrences of the precomputed values and multiplying the results by normalization factors); and alternative choices of sampling points must be made possible. Lesur and Gubbins [26] have studied the accuracy of the generalization of the FLT to the spherical harmonic transform, and they found numerical instabilities for higher order transforms. Future research should investigate how techniques such as precomputation in quadruple precision and our method of truncation improve the accuracy in the general case. Driscoll, Healy, and Rockmore [15] have already shown how a variant of the Driscoll–Healy algorithm may be used to compute Legendre transforms at any set of sample points (see Appendix A), though the set of points chosen affects the stability of the algorithm.

Appendix A. Related transforms and algorithms.

The derivation of the Driscoll–Healy algorithm given in section 2 depends only on the properties of truncation operators $\mathcal{T}_n$ given in Lemma 2.9 and on the existence of an efficient algorithm for applying the truncation operators. In particular, Lemmas 2.9 and 2.11 hold as stated when the weight function $\omega(x) = \pi^{-1}(1-x^2)^{\frac{1}{2}}$ is changed, when the truncation operators are defined using a polynomial sequence which is orthogonal with respect to the new weight function and which starts with the polynomial 1, and when the Lagrange interpolation operators are defined using the roots of the polynomials from the sequence. In theory, this can be used to develop new algorithms for computing orthogonal polynomial transforms, though with different sample weights w_j. In practice, however, the existence of efficient Chebyshev and cosine transform algorithms makes these the only reasonable choice in the definition of the truncation operators. This situation may change with the advent of other fast transforms.

Theoretically, the basic algorithm works, with minor modifications, in the following general situation. We are given operators $\mathcal{T}_n^r$ for $1 \leq n \leq r$ such that the following hold.

1. $\mathcal{T}_n^r$ is a mapping from the space of polynomials of degree less than $2r$ to the space of polynomials of degree less than n.
2. If $m \leq n \leq r$, then $\mathcal{T}_m^n \mathcal{T}_n^r = \mathcal{T}_m^r$.
3. If $\deg Q \leq m \leq n \leq r$, then $\mathcal{T}_{n-m}^r(f \cdot Q) = \mathcal{T}_{n-m}^n\left[(\mathcal{T}_n^r f) \cdot Q\right]$.

The problem now is, given an input polynomial f of degree less than N, to compute the quantities $\mathcal{T}_1^N(f \cdot p_l)$ for $0 \leq l < N$, where $\{p_l\}$ is a sequence of orthogonal polynomials. This problem may be treated using the same algorithms as in section 2, but with the truncation operators $\mathcal{T}_n$ replaced by $\mathcal{T}_n^r$, where $r \leq N$ depends on the stage of the algorithm. Using $r = N$ retrieves our original algorithm. The generalized algorithm uses the quantities $Z_l^K = \mathcal{T}_K^N(f \cdot p_l)$, and the recurrences in this context are

$$Z_{l+K-1}^K = \mathcal{T}_K^{2K}[Z_l^{2K} \cdot Q_{l,K-1} + Z_{l-1}^{2K} \cdot R_{l,K-1}], \tag{A.1}$$

$$Z_{l+K}^K \quad = \mathcal{T}_K^{2K}[Z_l^{2K} \cdot Q_{l,K} + Z_{l-1}^{2K} \cdot R_{l,K}], \tag{A.2}$$

cf. (2.18) and (2.19).

This generalization of our approach may be used to derive the original algorithm of Driscoll and Healy in the exact form it was presented [13, 14], which uses the cosine transforms in the points $\cos(j\pi/K)$. For more details, see [24].

Driscoll, Healy, and Rockmore [15] describe another variant of the Driscoll–Healy algorithm that may be used to compute the Legendre transform of a polynomial sampled at the Gaussian points, i.e., at the roots of the Legendre polynomial P_N. Their method replaces the initial Chebyshev transform used to find the polynomial Z_0^N in Chebyshev representation by a Chebyshev transform taken at the Gaussian points. Once Z_0^N has been found in Chebyshev representation, the rest of the computation is the same.

The Driscoll–Healy algorithm can also be used for input vectors of arbitrary size, not only powers of two. Furthermore, at each stage, we can split the problem into an arbitrary number of subproblems, not only into two. This requires that Chebyshev transforms of suitable sizes are available.

Appendix B. The precomputed data.

In this appendix we describe algorithms for generating the point values of $Q_{l,m}, R_{l,m}$ used in the recurrence of the FLT algorithm and for generating the coefficients $q_{l,m}^n, r_{l,m}^n$ used in its termination stage.

Lemma B.1. *Let $l \geq 1$, $j \geq 0$, and $k \geq 1$. Then the associated polynomials $Q_{l,m}, R_{l,m}$ satisfy the recurrences*

$$Q_{l,j+k} = Q_{l+k,j} Q_{l,k} + R_{l+k,j} Q_{l,k-1}, \tag{B.1}$$

$$R_{l,j+k} = Q_{l+k,j} R_{l,k} + R_{l+k,j} R_{l,k-1}. \tag{B.2}$$

Proof. The proof is by induction on j. The proof for $j = 0$ follows immediately from the definition (2.13), since $Q_{l+k,0} Q_{l,k} + R_{l+k,0} Q_{l,k-1} = 1 \cdot Q_{l,k} + 0 = Q_{l,k}$ and similarly for $R_{l,k}$. The case $j = 1$ also follows immediately from the definition. For

$j > 1$, we have

$$\begin{aligned} Q_{l+k,j}Q_{l,k} + R_{l+k,j}Q_{l,k-1} &= \left[Q_{l+k+j-1,1}Q_{l+k,j-1} + R_{l+k+j-1,1}Q_{l+k,j-2}\right] Q_{l,k} \\ &\quad + \left[Q_{l+k+j-1,1}R_{l+k,j-1} + R_{l+k+j-1,1}R_{l+k,j-2}\right] Q_{l,k-1} \\ &= Q_{l+k+j-1,1}\left[Q_{l+k,j-1}Q_{l,k} + R_{l+k,j-1}Q_{l,k-1}\right] \\ &\quad + R_{l+k+j-1,1}\left[Q_{l+k,j-2}Q_{l,k} + R_{l+k,j-2}Q_{l,k-1}\right] \\ &= Q_{l+k+j-1,1}Q_{l,k+j-1} + R_{l+k+j-1,1}Q_{l,k+j-2} \\ &= Q_{l,k+j}, \end{aligned}$$

where we have used the case $j = 1$ to prove the first and last equality and the induction hypothesis for the cases $j-1, j-2$ to prove the third equality. In the same way we may show that $Q_{l+k,j}R_{l,k} + R_{l+k,j}R_{l,k-1} = R_{l,k+j}$. □

This lemma is the basis for the computation of the data needed in the recurrences of the Driscoll–Healy algorithm. The basic idea of the Algorithm B.1 is to start with polynomials of degree $0, 1$, given in only one point, and then repeatedly double the number of points by performing a Chebyshev transform, adding zero terms to the Chebyshev expansion, and transforming back, and also double the maximum degree of the polynomials by applying the lemma with $j = K-1, K$ and $k = K$.

Algorithm B.1 Precomputation of the point values.

INPUT N: a power of 2.

OUTPUT $Q_{l,m}(x_j^{2^k})$, $R_{l,m}(x_j^{2^k})$ for $1 \le k \le \log_2 N$, $0 \le j < 2^k$, $m = 2^{k-1}, 2^{k-1}-1$, and $l = 1, 2^{k-1}+1, \ldots, N - 2^{k-1} + 1$.

```
STAGES
 0. for $l = 1$ to $N$ do
        $Q_{l,0}(0) \leftarrow 1, \quad R_{l,0}(0) \leftarrow 0, \quad Q_{l,1}(0) \leftarrow B_l, \quad R_{l,1}(0) \leftarrow C_l$
 k. for $k = 1$ to $\log_2 N$ do
        $K \leftarrow 2^{k-1}$
        for $m = K-1$ to $K$ do
            for $l = 1$ to $N-K+1$ step $K$ do
                $(q_{l,m}^0, \ldots, q_{l,m}^{K-1}) \leftarrow \text{Chebyshev}(Q_{l,m}(x_0^K), \ldots, Q_{l,m}(x_{K-1}^K))$
                $(r_{l,m}^0, \ldots, r_{l,m}^{K-1}) \leftarrow \text{Chebyshev}(R_{l,m}(x_0^K), \ldots, R_{l,m}(x_{K-1}^K))$
                $(q_{l,m}^K, \ldots, q_{l,m}^{2K-1}) \leftarrow (0, \ldots, 0)$
                if $m = K$ then $q_{l,m}^K \leftarrow A_l A_{l+1} \cdots A_{l+m-1}/2^{m-1}$
                $(r_{l,m}^K, \ldots, r_{l,m}^{2K-1}) \leftarrow (0, \ldots, 0)$
                $(Q_{l,m}(x_0^{2K}), \ldots, Q_{l,m}(x_{2K-1}^{2K})) \leftarrow \text{Chebyshev}^{-1}(q_{l,m}^0, \ldots, q_{l,m}^{2K-1})$
                $(R_{l,m}(x_0^{2K}), \ldots, R_{l,m}(x_{2K-1}^{2K})) \leftarrow \text{Chebyshev}^{-1}(r_{l,m}^0, \ldots, r_{l,m}^{2K-1})$
        for $l = 1$ to $N-2K+1$ step $2K$ do
            for $j = 0$ to $2K-1$ do
                $Q_{l,2K}(x_j^{2K}) \leftarrow Q_{l+K,K}(x_j^{2K})Q_{l,K}(x_j^{2K}) + R_{l+K,K}(x_j^{2K})Q_{l,K-1}(x_j^{2K})$
                $R_{l,2K}(x_j^{2K}) \leftarrow Q_{l+K,K}(x_j^{2K})R_{l,K}(x_j^{2K}) + R_{l+K,K}(x_j^{2K})R_{l,K-1}(x_j^{2K})$
                $Q_{l,2K-1}(x_j^{2K}) \leftarrow Q_{l+K,K-1}(x_j^{2K})Q_{l,K}(x_j^{2K}) + R_{l+K,K-1}(x_j^{2K})Q_{l,K-1}(x_j^{2K})$
                $R_{l,2K-1}(x_j^{2K}) \leftarrow Q_{l+K,K-1}(x_j^{2K})R_{l,K}(x_j^{2K}) + R_{l+K,K-1}(x_j^{2K})R_{l,K-1}(x_j^{2K})$
```

Note that $\deg R_{l,m} \le m-1$, so the Chebyshev coefficients $r_{l,m}^n$ with $n \ge m$ are zero, which means that the polynomial is fully represented by its first m Chebyshev coefficients. In the case of the $Q_{l,m}$, the coefficients are zero for $n > m$. If $n = m$, however, the coefficient is nonzero, and this is a problem if $m = K$. The Kth coefficient which was set to zero must then be corrected and set to its true value, which can be computed easily by using (2.13) and (2.4).

The FLT algorithm requires the numbers

$$Q_{l,K}(x_j^{2K}),\quad Q_{l,K-1}(x_j^{2K}),\quad R_{l,K}(x_j^{2K}),\quad R_{l,K-1}(x_j^{2K}),\quad 0 \le j < 2K,$$

for $l = r \cdot 2K + 1$, $0 \le r < \frac{N}{2K}$, for all K with $M \le K \le N/2$. After the m-loop in stage $k = \log_2 K + 1$ of Algorithm B.1, we have obtained these values for $l = rK + 1$, $0 \le r < N/K$. We need only the values for even r, so the others can be discarded. The algorithm must be continued until $K = N/2$, i.e., $k = \log_2 N$.

The total number of flops of the precomputation of the point values is

$$T_{\text{precomp, point}} = 6\alpha N \log_2^2 N + (2\alpha + 12\beta + 12) N \log_2 N. \tag{B.3}$$

Comparing with the cost (2.25) of the Driscoll–Healy algorithm itself and considering only the highest order term, we see that the precomputation costs about three times as much as the Driscoll–Healy algorithm without early termination. This one-time cost, however, can be amortized over many subsequent executions of the algorithm.

Parallelizing the precomputation of the point values can be done most easily by using the block distribution. This is similar to our approach in deriving a basic parallel version of the Driscoll–Healy algorithm. In the early stages of the precomputation, each processor handles a number of independent problems, one for each l. At the start of stage k, such a problem involves K points. In the later stages, each problem is assigned to one processor group. The polynomials $Q_{l,K}$, $Q_{l,K-1}$, $R_{l,K}$, $R_{l,K-1}$, and $Q_{l+K,K}$, $Q_{l+K,K-1}$, $R_{l+K,K}$, $R_{l+K,K-1}$ are all distributed in the same manner, so that the recurrences are local. The Chebyshev transforms and the addition of zeros may require communication. For the addition of zeros, this is caused by the desire to maintain a block distribution while doubling the number of points. The parallel precomputation algorithm can be optimized following similar ideas as in the optimized main algorithm.

The precomputation of the coefficients $q_{l,m}^n, r_{l,m}^n$ required to terminate the Driscoll–Healy algorithm early, as in Lemma 2.12, is based on the following recurrences.

LEMMA B.2. *Let $l \ge 1$ and $m \ge 2$. The coefficients $q_{l,m}^n$ satisfy the recurrences*

$$q_{l,m}^n = \frac{1}{2} A_{l+m-1}(q_{l,m-1}^{n+1} + q_{l,m-1}^{n-1}) + B_{l+m-1} q_{l,m-1}^n + C_{l+m-1} q_{l,m-2}^n \text{ for } n \ge 2,$$

$$q_{l,m}^1 = A_{l+m-1}(q_{l,m-1}^0 + \frac{1}{2} q_{l,m-1}^2) + B_{l+m-1} q_{l,m-1}^1 + C_{l+m-1} q_{l,m-2}^1,$$

$$q_{l,m}^0 = \frac{1}{2} A_{l+m-1} q_{l,m-1}^1 + B_{l+m-1} q_{l,m-1}^0 + C_{l+m-1} q_{l,m-2}^0,$$

subject to the boundary conditions $q_{l,0}^0 = 1, q_{l,1}^0 = B_l, q_{l,1}^1 = A_l$, and $q_{l,m}^n = 0$ for $n > m$. The $r_{l,m}^n$ satisfy the same recurrences but with boundary conditions $r_{l,1}^0 = C_l$ and $r_{l,m}^n = 0$ for $n \ge m$.

Proof. These recurrences are the shifted three-term recurrences (2.13) rewritten in terms of the Chebyshev coefficients of the polynomials by using the equations $x \cdot T_n = (T_{n+1} + T_{n-1})/2$ for $n > 0$ and $x \cdot T_0 = T_1$. □

For a fixed l, we can compute the $q_{l,m}^n$ and $r_{l,m}^n$ by increasing m, starting with the known values for $m = 0, 1$, and finishing with $m = M - 2$. For each m, we need to compute only the $q_{l,m}^n$ with $n \le m$ and the $r_{l,m}^n$ with $n < m$. The total number of flops of the precomputation of the Chebyshev coefficients in the general case is

$$T_{\text{precomp, term}} = 7M^2 - 16M - 15. \tag{B.4}$$

When the initial values B_l are identically zero, the coefficients can be packed in alternating fashion into array $\mathbf{T}$, as shown in Figure 3.3. In that case the cost is considerably lower, namely, $2.5M^2 - 3.5M - 12$.

The precomputed Chebyshev coefficients can be used to save the early stages in Algorithm B.1. If we continue the precomputation of the Chebyshev coefficients two steps more and finish with $m = M$ instead of $m = M - 2$, we then can switch directly to the precomputation of the point values at stage $K = M$, just after the forward Chebyshev transforms.

Parallelizing the precomputation of the Chebyshev coefficients is straightforward, since the computation for each l is independent. Therefore, if $M \leq N/p$, both the termination and its precomputation are local operations.

Acknowledgments. We thank Dennis Healy, Daniel Rockmore, and Peter Kostelec for useful discussions. We thank the anonymous referees for valuable suggestions for improvement.

REFERENCES

[1] N. Ahmed, T. Natarajan, and K. R. Rao, *Discrete cosine transform*, IEEE Trans. Comput., 23 (1974), pp. 90–93.

[2] B. K. Alpert and V. Rokhlin, *A fast algorithm for the evaluation of Legendre expansions*, SIAM J. Sci. Statist. Comput., 12 (1991), pp. 158–179.

[3] S. R. M. Barros and T. Kauranne, *On the parallelization of global spectral weather models*, Parallel Comput., 20 (1994), pp. 1335–1356.

[4] P. Barrucand and D. Dickinson, *On the associated Legendre polynomials*, in Orthogonal Expansions and Their Continuous Analogues, D. T. Haimo, ed., Southern Illinois University Press, Carbondale, IL, 1968, pp. 43–50.

[5] S. Belmehdi, *On the associated orthogonal polynomials*, J. Comput. Appl. Math., 32 (1990), pp. 311–319.

[6] R. H. Bisseling, *Basic techniques for numerical linear algebra on bulk synchronous parallel computers*, in Numerical Analysis and Its Applications, Lecture Notes in Comput. Sci. 1196, Springer-Verlag, Berlin, 1997, pp. 46–57.

[7] R. H. Bisseling, *BSPEDUpack,* http://www.math.uu.nl/people/bisseling/software.html, 2000.

[8] O. Bonorden, B. Juurlink, I. von Otte, and I. Rieping, *The Paderborn University BSP (PUB) library—design, implementation and performance*, in Proceedings of the 13th International Parallel Processing Symposium and the 10th Symposium on Parallel and Distributed Processing, CD-ROM, IEEE Computer Society, Los Alamitos, CA, 1999.

[9] G. L. Browning, J. J. Hack, and P. N. Swarztrauber, *A comparison of three numerical methods for solving differential equations on the sphere*, Monthly Weather Review, 117 (1989), pp. 1058–1075.

[10] T. S. Chihara, *An Introduction to Orthogonal Polynomials*, Gordon and Breach, New York, 1978.

[11] J. W. Cooley and J. W. Tukey, *An algorithm for the machine calculation of complex Fourier series*, Math. Comp., 19 (1965), pp. 297–301.

[12] G. A. Dilts, *Computation of spherical harmonic expansion coefficients via FFTs*, J. Comput. Phys., 57 (1985), pp. 439–453.

[13] J. R. Driscoll and D. M. Healy, Jr., *Computing Fourier transforms and convolutions on the* 2*-sphere*, Adv. Appl. Math., 15 (1994), pp. 202–250.

[14] J. R. Driscoll and D. M. Healy, Jr., *Computing Fourier transforms and convolutions on the* 2*-sphere*, extended abstract in Proceedings of the 30th IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, Los Alamitos, CA, 1989, pp. 344–349.

[15] J. R. Driscoll, D. M. Healy, Jr., and D. N. Rockmore, *Fast discrete polynomial transforms with applications to data analysis for distance transitive graphs*, SIAM J. Comput., 26 (1997), pp. 1066–1099.

[16] B. S. Duncan and A. J. Olson, *Approximation and characterization of molecular surfaces*, Biopolymers, 33 (1993), pp. 219–229.

[17] D. M. Healy, Jr. and P. T. Kim, *Spherical Deconvolution with Application to Geometric Quality Assurance*, Tech. report, Department of Mathematics and Computer Science, Dartmouth College, Hanover, NH, 1993.

[18] D. M. Healy, Jr. and P. T. Kim, *An empirical Bayes approach to directional data and efficient computation on the sphere*, Ann. Statist., 24 (1996), pp. 232–254.

[19] D. M. Healy, Jr., S. S. B. Moore, and D. Rockmore, *Efficiency and Stability Issues in the Numerical Computation of Fourier Transforms and Convolutions on the* 2*-Sphere*, Tech. report PCS-TR94-222, Department of Mathematics and Computer Science, Dartmouth College, Hanover, NH, 1994.

[20] D. M. Healy, Jr., D. Rockmore, P. J. Kostelec, and S. S. B. Moore, *FFTs for the* 2*-Sphere—Improvements and Variations*, Tech. report, Department of Mathematics and Computer Science, Dartmouth College, Hanover, NH, 1998.

[21] J. M. D. Hill, B. McColl, D. C. Stefanescu, M. W. Goudreau, K. Lang, S. B. Rao, T. Suel, T. Tsantilas, and R. H. Bisseling, *BSPlib: The BSP programming library*, Parallel Comput., 24 (1998), pp. 1947–1980.

[22] J. M. D. Hill, S. R. Donaldson, and A. McEwan, *Installation and User Guide for the Oxford BSP Toolset (v*1.4*) Implementation of BSPlib*, Tech. report, Oxford University Computing Laboratory, Oxford, UK, 1998.

[23] M. A. Inda, *Constructing Parallel Algorithms for Discrete Transforms: From FFTs to Fast Legendre Transforms*, Ph.D. thesis, Department of Mathematics, Utrecht University, Utrecht, The Netherlands, 2000.

[24] D. K. Maslen, *A Polynomial Approach to Orthogonal Polynomial Transforms*, Preprint MPI/95-9, Max-Planck-Institut für Mathematik, Bonn, Germany, 1995.

[25] W. F. McColl, *Scalability, portability and predictability: The BSP approach to parallel programming*, Fut. Gen. Comp. Syst., 12 (1996), pp. 265–272.

[26] V. Lesur and D. Gubbins, *Evaluation of fast spherical transforms for geophysical applications*, Geophys. J. Int., 139 (1999), pp. 547–555.

[27] T. Ooura, *General purpose FFT (fast Fourier/cosine/sine transform) package,* http://momonga.t.u-tokyo.ac.jp/õoura/fft.html, 1998.

[28] S. A. Orszag, *Fast eigenfunction transforms*, in Science and Computers, G.-C. Rota, ed., Academic Press, Orlando, FL, 1986, pp. 23–30.

[29] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed., Cambridge University Press, Cambridge, UK, 1992.

[30] K. R. Rao and P. Yip, *Discrete Cosine Transform: Algorithms, Advantages, and Applications*, Academic Press, San Diego, CA, 1990.

[31] N. Shalaby, *Parallel Discrete Cosine Transforms: Theory and Practice*, Tech. report TR-34-95, Center for Research in Computing Technology, Harvard University, Cambridge, MA, 1995.

[32] N. Shalaby and S. L. Johnsson, *Hierarchical load balancing for parallel fast Legendre transforms*, in Proceedings of the 8th SIAM Conference on Parallel Processing for Scientific Computing, M. Heath et al., eds., SIAM, Philadelphia, 1997.

[33] G. Steidl and M. Tasche, *A polynomial approach to fast algorithms for discrete Fourier-cosine and Fourier-sine transforms*, Math. Comp., 56 (1991), pp. 281–296.

[34] L. G. Valiant, *A bridging model for parallel computation*, Comm. ACM, 33 (1990), pp. 103–111.

[35] C. Van Loan, *Computational Frameworks for the Fast Fourier Transform*, SIAM, Philadelphia, 1992.

SIAM J. Sci. Comput.
Vol. 23, No. 1, pp. 304–322

A LOCAL SPACE-TIME ADAPTIVE SCHEME IN SOLVING TWO-DIMENSIONAL PARABOLIC PROBLEMS BASED ON DOMAIN DECOMPOSITION METHODS[*]

HONGYI YU[†]

Abstract. We describe a local space-time adaptive refinement and derefinement method for solving systems of parabolic partial differential equations. Solutions are calculated on local space-time meshes using a domain decomposition finite element algorithm in space and an extrapolation algorithm in time. An a posteriori spatial error estimate controls a local spatial refinement strategy and an a posteriori temporal error estimate monitors temporal refinement and order of the integration. A multiplicative Schwarz algorithm is used to coordinate solutions between overlapping grids. We use nearly minimal fine meshes which follow the dynamics of the problems in an efficient manner.

Key words. local AMR scheme, domain decomposition, extrapolation methods, multiplicative Schwarz algorithm

AMS subject classification. 65Y20

PII. S1064827500315360

1. Introduction.

Adaptive local refinement methods have been effectively used to solve a wide range of systems of partial differential equations [1, 3, 4, 5, 6, 7, 15, 16, 17, 18, 19]. Two basic strategies are structured and unstructured mesh refinement. Structured methods are further divided into cellular and noncellular methods, where cellular means that the fine mesh is aligned with the coarse mesh (see Figure 1a) and noncellular mesh means that the fine mesh can be skewed with respect to the coarse mesh (see Figure 1b).

Berger and Oliger [5] developed a noncellular overlapping uniform grid algorithm which allows alignment of fine grids with evolving dynamic structures. However, the data transition between coarse and fine grids is complicated. Multiple levels of overlapping refined grids, as shown in Figure 1b, require complicated data structures and data transition as well. Additionally, when using implicit methods in time, grids must not cross domain boundaries.

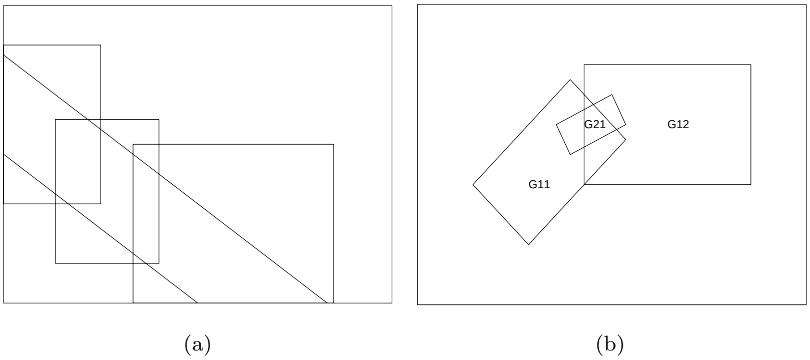
Arney and Flaherty [3] use cellular meshes to form fine grids with piecewise polygonal shapes. Data transition between fine and coarse grids is simplified with cellular refinement. However, the data structure is still complicated, since the refined grids are generally not rectangles. Thus, information associated with each node has to include pointers to the nodes above and below the node in the mesh. Further complications arise if high-order elements are used.

Moore and Flaherty [16] use rectangular meshes which are aligned with the coarse mesh to form overlapping patches. The problem is solved on the patches using a Schwarz alternating algorithm. The data structure is simplified in this case, but more work is required due to the large overlap between the rectangular meshes and over-refinement resulting from grids aligned with domain boundaries rather than solution features (see Figure 1a).

Another approach is to use unstructured meshes with triangular elements. These methods are more economical than the structured mesh methods in the sense that a

[*] Received by the editors October 18, 1999; accepted for publication (in revised form) October 31, 2000; published electronically June 19, 2001.
http://www.siam.org/journals/sisc/23-1/31536.html

[†] 10243 Parkwood Dr. #6, Cupertino, CA 95014 (hongyi.yu@silvaco.com).



(a) (b)

FIG. 1. *Cellular and noncellular overlapping grids.*

minimum region required for local solution is created. But the data structures are even more complicated, and so more memory is needed.

Domain decomposition methods have also become popular. The classical Schwarz alternating method has been extended to more than two subdomains and additive and multiplicative Schwarz methods have been developed [8, 11, 13, 21]. The methods have been widely used to solve elliptic partial differential equations as well as some simple, linear parabolic partial differential equations [11, 9, 10, 13]. The advantage of these methods is that each subdomain can be sent to a separate processor with communications needed only at the interfaces.

All the methods mentioned above use uniform time stepping when applied to solving parabolic partial differential equations. Herein, we develop a local space-time adaptive mesh refinement procedure coupled with a domain decomposition multiplicative Schwarz algorithm on structured meshes which has the advantage of using simple spatial grids, nearly minimal spatial overrefinement, and a variable time step over the spatial regions. By nearly minimal spatial overrefinement, we mean that the spatial overrefinement and resolution procedure are performed only in the regions that cover and are nearly as large as the regions that require a resolution as indicated by a posterior error estimator. Examples in section 5 show that our methods automatically create fine spatial meshes and small time steps (or fine space-time meshes) in regions of high activity based on the error estimates. Coarse spatial meshes and large time steps (or coarse space-time meshes) are used in regions of low activity.

Note: For clarity, we sometimes separate spatial meshes and time steps from space-time meshes as we did in the last paragraph.

In section 2, the model problem is defined and the underlying space discretization algorithm is presented. In section 3, we discuss the one- and two-level overlapping domain decomposition and the multiplicative Schwarz algorithm. We will give a detailed description of the local adaptive mesh refinement (AMR) procedure in section 4. Some examples that show the effectiveness of the method are presented in section 5. Concluding remarks are given in section 6.

2. The model problem and discretization.

Consider systems of nonlinear parabolic equations of the form

$$u_t + f(t, x, y, u, u_x, u_y) = [D_1(t, x, y, u)u_x]_x + [D_2(t, x, y, u)u_y]_y \tag{1}$$

for $(x, y) \in \Omega, t > 0$, together with the initial conditions

$$u(x, y, 0) = u_0(x, y), \quad (x, y) \in \overline{\Omega}, \tag{2}$$

and the mixed boundary conditions

$$u(t, x, y) = g_D(t, x, y), \quad (x, y) \in \partial\Omega_D, \quad t > 0, \tag{3}$$

$$D_1 u_x \eta^1 + D_2 u_y \eta^2 = g_N(t, x, y), \quad (x, y) \in \partial\Omega_N, \quad t > 0, \tag{4}$$

where (η^1, η^2) is the unit outer normal of $\partial\Omega$ and $\partial\Omega_D \bigcup \partial\Omega_N = \partial\Omega$, and $\Omega = [x_l, x_r] \times [y_l, y_r]$. The system is assumed to be well posed, i.e., D_1, D_2 are positive definite.

We solve the mixed boundary value problem (1)–(4) using a Galerkin finite element method with a hierarchical polynomial basis of degree $p \geq 1$ in space coupled with a variable-order, variable-step extrapolation method in time.

The Galerkin form of (1)–(4) is to find $u \in H_E^1$ such that

$$(u_t, v) + A(u, v) + (f, v) = \int_{\partial\Omega_N} v^T g_N ds \quad \forall v \in H_0^1, \quad t > 0, \tag{5}$$

$$A(u, v) = A(u^0, v) \qquad \forall v \in H_0^1, \quad t = 0, \tag{6}$$

where

$$(v, u) = \int_\Omega v(t, x, y)^T u(t, x, y) dx dy$$

and

$$A(u, v) = (D_1 u_x, v_x) + (D_2 u_y, v_y).$$

As usual, the Sobolev space $H^1(\Omega)$ consists of functions having square integrable first partial derivatives. The subscripts E and 0 further restrict this space to functions that satisfy (3) and a homogeneous version of (3), respectively.

Introduce partitions

$$\Omega_x = \{x_l = x_0 < x_1 < \cdots < x_{n_x} = x_r\},$$

$$\Omega_y = \{y_l = y_0 < y_1 < \cdots < y_{n_y} = y_r\}$$

of Ω into $m = n_x n_y$ elements. Let

$$h_x = \frac{x_r - x_l}{n_x}, \quad h_y = \frac{y_r - y_l}{n_y}, \quad h = \sqrt{h_x^2 + h_y^2}, \text{ and } \Delta^h = \Omega_x \times \Omega_y.$$

We approximate $H^1(\Omega)$ by a finite-dimensional subspace $S^{p,h}$ of piecewise polynomials of degree p, $p \geq 1$, defined on Δ^h. The finite element Galerkin approximation $U(t, x, y) \in S_E^{p,h}$ to $u(t, x, y) \in H_E^1(\Omega)$ is obtained by solving

$$(U_t, V) + A(U, V) + (f, V) = \int_{\partial\Omega_N} V^T g_N ds \quad \forall V \in S_0^{p,h}, \quad t > 0, \tag{7}$$

$$A(U, V) = A(u_0, V) \quad \forall V \in S_0^{p,h}, \quad t = 0. \tag{8}$$

A hierarchical basis is constructed for $S^{p,h}$ with rectangular elements using the tensor products of Legendre polynomials. We refer to Szabo and Babuska [20] for details.

3. Domain decomposition and multiplicative Schwarz methods.

3.1. One- and two-level decompositions with small overlap. Let Δ^H be a coarse mesh or H-level subdivision of Ω and let $\{\Omega_i\}_{i=1}^M$ be the elements of Δ^H with $M = N_x N_y$, where N_x and N_y are the number of subdivisions in the x- and y-directions, respectively. We further divide each coarse element Ω_i into smaller rectangles of lengths h_x and h_y, where $h_x = \frac{H_x}{r}$, $h_y = \frac{H_y}{r}$, and r is an integer (we take $r = 3$). The mesh Δ^h with $h = \sqrt{h_x^2 + h_y^2}$ is called the fine mesh or h-level subdivision of Ω. To obtain an overlapping decomposition, we extend each coarse element $\Omega_i = (x_i, x_{i+1}) \times (y_i, y_{i+1})$ to a slightly larger extended element $\Omega_i^{ext} = (x_i, x_{i+1} + \tilde{h}_x) \times (y_i, y_{i+1} + \tilde{h}_y)$, where

$$\tilde{h}_x = \begin{cases} h_x & \text{if } x_{i+1} < x_r, \\ 0 & \text{if } x_{i+1} = x_r, \end{cases} \qquad \tilde{h}_y = \begin{cases} h_y & \text{if } y_{i+1} < y_r, \\ 0 & \text{if } y_{i+1} = y_r. \end{cases}$$

We also require that $\Omega_i^{ext} \supset \overline{\Omega_i^{ext}} \bigcap \partial\Omega_N$, i.e., the Neumann boundary is included in $\bigcup_i \Omega_i^{ext}$.

The purpose of creating $\{\Omega_i^{ext}\}_{i=1}^M$ is to form overlapping subdomains of Ω. Using the extended elements, we can always form four overlapping subdomains as follows:

Let $m = (k-1)N_x + j$, the element number with $1 \le k \le N_y$ and $1 \le j \le N_x$. Define

$$\tilde{\Omega}_1 = \left\{ \bigcup \Omega_m^{ext}; \quad k, j \text{ are odd} \right\}, \qquad \tilde{\Omega}_2 = \left\{ \bigcup \Omega_m^{ext}; \quad k \text{ is odd and } j \text{ is even} \right\},$$

$$\tilde{\Omega}_4 = \left\{ \bigcup \Omega_m^{ext}; \quad k,\ j \text{ are even} \right\}, \qquad \tilde{\Omega}_3 = \left\{ \bigcup \Omega_m^{ext}; \quad k \text{ is even and } j \text{ is odd} \right\}.$$

Then we have

$$\bigcup_{i=1}^{4} \tilde{\Omega}_i = \bigcup_{i=1}^{M} \Omega_i^{ext}.$$

Each $\tilde{\Omega}_i$ is a union of disjoint extended elements, as illustrated in Figure 2 shown with their fine element structures. For convenience, each extended element Ω_i^{ext} with its fine element structure is called a local grid. Thus, $\tilde{\Omega}_1$ consists of 16 local grids, $\tilde{\Omega}_2$ and $\tilde{\Omega}_3$ both have 12, and $\tilde{\Omega}_4$ has 9 local grids. The overlaps between different $\tilde{\Omega}_i$'s have a width of one fine element. For each $\tilde{\Omega}_j, 1 \le j \le J$ (here $J = 4$), we define

$$V_j = \{v \in V^h; \quad v(x,y) = 0, (x,y) \notin \tilde{\Omega}_j\} \subset V^h$$

with $V^h = S_0^{p,h}$. If we denote $S_0^{p,H}$ by V_0, then we have the following two decompositions of V^h:

$$V^h = V_1 + \cdots + V_J$$

and

$$V^h = V_0 + V_1 + \cdots + V_J,$$

which will be referred to as the one-level and two-level uniformly overlapping decompositions of V^h, or as the decomposition without and with the coarse mesh space, respectively.

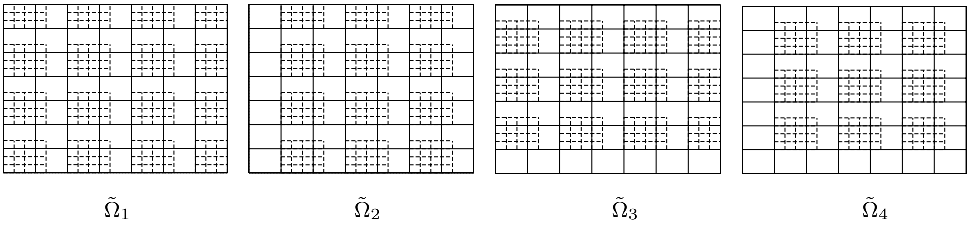


FIG. 2. *Partition of the entire domain into overlapping subdomains.*

We have eliminated the condition

$$\text{(9)} \qquad Distance\left(\partial\Omega_i^{ext}\bigcap\Omega, \partial\Omega_i\bigcap\Omega\right) \geq cH \quad \forall i, \quad c > 0,$$

proposed by Cai [9] and therefore achieved smaller overlaps between the local grids. The systems are made smaller, and in most cases the computational speed is increased [22]. The experiments also showed that the size of the overlap affects neither the accuracy of the solution nor the convergence of the multiplicative Schwarz algorithm.

3.2. Multiplicative Schwarz algorithms. We briefly describe the multiplicative Schwarz algorithm based on the two decompositions of V^h as described in section 3.1. We assume a set of triplets $\{W_i, P_i, w_i | i = 1, 2, \ldots, m\}$, where W_i are some subspaces of a normed linear space W, P_i are mappings from W to W_i, and $w_i \in W_i$. The multiplicative Schwarz algorithm can be described as follows:

MULTIPLICATIVE SCHWARZ $\{W_i, P_i, w_i\}$.

```
Given an initial guess $u^0 \in W$,
for $n = 0, 1, \ldots, n_{max}$;
    for $i = 1, \ldots, m$;
        $u^{n+\frac{i}{m}} = u^{n+\frac{i-1}{m}} + (w_i - P_i u^{n+\frac{i-1}{m}})$.
```

Remark 1. For linear problems, we have

$$u^{n+1} = Eu^n + g,$$

where

$$E = (I - P_m) \cdots (I - P_1),$$

and

$$g = (I - P_m) \cdots (I - P_2)w_1 + (I - P_m) \cdots (I - P_3)w_2 + \cdots + w_m.$$

The convergence rate of the multiplicative Schwarz algorithm is thus determined by the spectral radius of the operator E.

Remark 2. Let w be a fixed element in $S_E^{p,H}$. The Schwarz algorithm can be extended to the more general case where space W is replaced by $w + W$ and the mappings P_i are defined as $P_i : w + W \to W_i$. We will use the extended version of it.

The parabolic system is solved by dividing the time interval into several smaller ones. To apply the multiplicative Schwarz algorithm on interval $[t_j, t_{j+1}]$, we take W

to be $g_D + V^h$, W_i to be the subspace V_i of V^h, and $w_i \in W_i$ to be the solution of the ODE system

$$(10) \qquad (u_t, v) + A(u, v) + (f, v) = \int_{\partial\Omega_N \bigcap \tilde{\Omega_i}} v^T g_N ds \quad \forall v \in V_i, \quad t_j < t \le t_{j+1},$$

with the initial condition $u = u(t_j)$ for $t = t_j$. When $t_j = 0$, $u(0)$ is the solution from (8). The mapping $P_i : g_D + V^h \to V_i$ is defined by

$$(11) \qquad A(g_D + u, v) = A(P_i(g_D + u), v) \quad \forall v \in V_i, \quad \forall u \in V^h.$$

The ODE systems are solved independently in the interior of the local grids. While the projections P_i involve the information from the local grid boundaries, the multiplicative Schwarz algorithm merges the two sets of systems and passes information from grid to grid throughout the entire domain.

With a chosen base time step, $\Delta t_j = t_{j+1} - t_j$, on the entire domain, we solve the ODE systems (10) in the interior of each small local grid using the extrapolation method [12]. A variable time step and a variable integration order are used based on an a posteriori error estimate. Fine temporal meshes match fine spatial meshes in a consistent manner as shown in section 5. The examples in section 5 also show the commanding advantage, in terms of efficiency, of our adaptive space-time meshes over the adaptive space meshes with uniform time steps. For a detailed description of the time integration procedure, please refer to [22].

In applying the multiplicative Schwarz algorithm, we can use either the one-level or two-level decomposition. In general, the two-level decomposition requires fewer iterations than one-level decomposition, since the solution on the coarse grid improves the initial guess for the solutions on the fine grid and smooths the solution from the overlapping subdomains. Table 1 in section 5 compares the number of iterations via the two different ways of decomposition.

4. Local adaptive mesh refinement (AMR) procedure.

4.1. Local AMR procedure.

We start with a uniform grid in space and a base time step. The spatial grid is divided into some overlapping subgrids and the systems (10)–(11) are solved on each subgrid. Time steps are adaptively refined or derefined based on an a posteriori temporal error estimate on each subgrid independent of one another. The maximum of the local time steps is used for the next base time step.

At the end of each base time step, an a posteriori spatial error estimate is performed elementwise. A dichotomy principle of Babuska, Theorems 3.1 and 4.1 in [2], is used for the error estimate, which says that errors in finite element solutions in two space dimensions computed with odd-degree polynomials arise mainly near element edges, while those computed with even-degree approximations arise mainly in element interiors. Elements having "high" error are grouped into different error sets, called photo-clusters, using Berger and Oliger's nearest neighbor algorithm [5] with a predefined distance λ (we take $\lambda = 2h$). We use a linked list to store these photo-clusters. A list iterator is used to build up such a list. The first photo-cluster is created by adding the first high-error element to it. Then we check the next high-error element; if it is within the predefined distance from the first cluster, we will add it to the cluster. Otherwise, we will create the second photo-cluster with this element in it. Follow such a procedure; if an error element is "far" from any of the constructed photo-clusters, a new photo-cluster is created and added to the list. If the error element is within

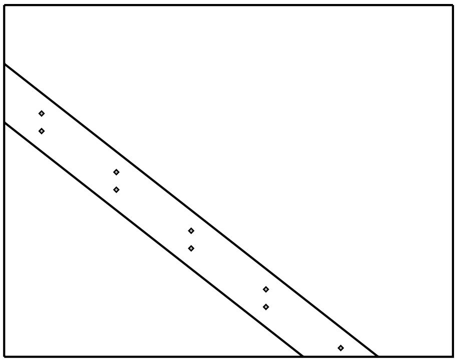


FIG. 3. *Cluster that intersects domain boundaries.*

a distance λ from two different photo-clusters, the two photo-clusters will be combined into one. By dividing the error elements into different photo-clusters, spatially distinct phenomena are separated.

For each photo-cluster, we form a local rectangle, called a cluster, using Berger and Oliger's procedure [5]. Each cluster is enlarged by increasing the length of its two sides so that the region between the enlarged and original clusters provides a buffer (we take the buffer-size to be $0.9h$) and artificial internal boundary conditions will be prescribed at low-error degrees of freedom, and thus fine-grid errors will not significantly propagate through the buffer. If we choose an appropriate λ and buffer zone (say, $\lambda > 2*$buffer-size), the different clusters will not overlap. Hence the solution procedure over different clusters will be independent.

Clusters may be skewed with respect to the coordinate axes. Thus, for each cluster, a second rectangle is formed, which we call a megagrid. The megagrid contains the cluster and is aligned with the coordinate axes. Its sides also overlap the grid-lines of the parent megagrid (the base grid is considered as a megagrid too). If the cluster intersects the parent megagrid boundaries, we take only the part inside the parent megagrid, as shown in Figure 3. The megagrids contain the information from the parent megagrid and pass the information to their clusters.

To keep the resolution region to a minimum we check the efficiency of each cluster which is measured by the ratio of the number of high-error elements to the total number of elements in the cluster. The cluster is considered as inefficient if the ratio is less than 0.3. Those clusters with six elements or less are assumed to be efficient. If the cluster is inefficient, we bisect it along the major side, then apply Berger and Oliger's procedure on each of the two parts separately to form two new clusters. This procedure is performed recursively until all the clusters are efficient. Figure 4a shows one example of such a case.

We refine each megagrid by a factor of 3, interpolate the data field on the refined megagrid, and decompose the megagrid into small rectangular grids. Those local grids that don't intersect any of the clusters in the megagrid are discarded to maximize the efficiency. This results in a piecewise polygonal region, as shown in Figure 4b, which is called a polygrid. The multiplicative Schwarz algorithm is then applied on the polygrid.

In some cases, different megagrids may overlap. But the polygrids within these intersecting megagrids are independent of each other, as shown in Figure 5. Since the re-solution process is on the polygrids, such procedures on different polygrids are

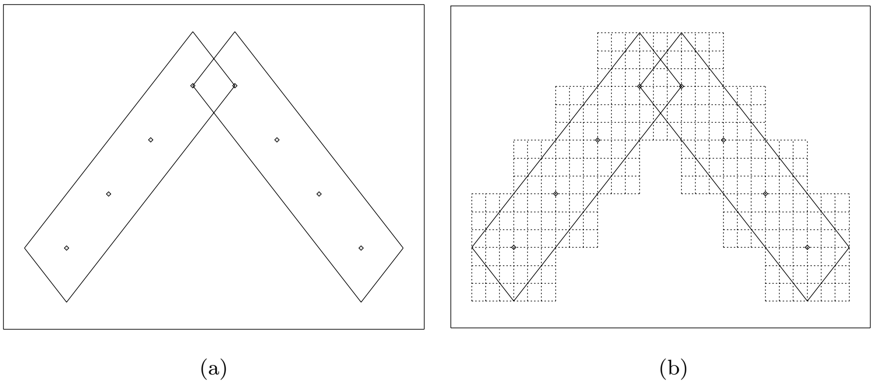


FIG. 4. *Cluster bisectioning and fine grids formation.*

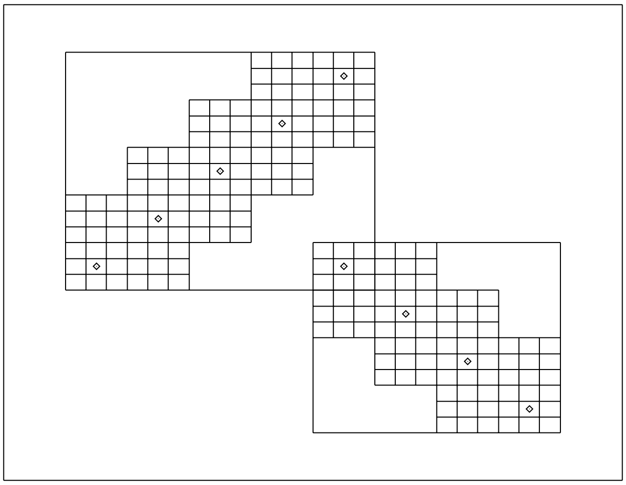


FIG. 5. *Grids formation for overlapping megagrids.*

independent of each other.

Each local grid is small with a maximum size of 4 × 4 and a minimum size of 3 × 3. If the mesh is fine enough, the refined polygrids will cover and possibly be slightly larger than the regions flagged with high error. Thus, a nearly minimal overrefinement and resolution is achieved. So our method has the advantage of both structured noncellular mesh and unstructured mesh methods. Since the megagrids are aligned with the problem domain and properly nested, the data structure is much simpler and data transition between megagrids is much easier. Compared to structured cellular grid methods, where solutions are computed over either the entire problem domain as in [3] or overlapping rectangular patches with large overlaps as in [16], we solve the problem on a smaller refined region with the same simplicity of data storage and data transition. Furthermore, the methods presented here enable us to solve the entire problem by solving many very small problems independently. It is therefore best suited for parallel computation. My next paper [23] will focus on issues of parallel computation using the methods provided here.

We use a recursive local AMR and derefinement procedure. A top level description

```
Procedure. Solve($t_{start}$, $t_{end}$, $\epsilon_{sp}$, $\epsilon_t$).
    begin
        $t = t_{start}$
        choose a base time step $\Delta t$
        generate the base grid $G$ (also a polygrid)
        decompose $G$ to form local grids
        initialize the solution field $u = u(t)$ on $G$
        do while($t < t_{end}$)
            begin
                $t \leftarrow t + \Delta t$
                LocalAMR($G$, $t$, $\Delta t$, $\epsilon_{sp}$, $\epsilon_t$)
            end
    end
```

FIG. 6. *Description of the solution procedure.*

```
Procedure. LocalAMR($G$, $t$, $\Delta t$, $\epsilon_{sp}$, $\epsilon_t$).
    begin
        Schwarz($G$, $t$, $\Delta t$, $\epsilon_t$)
        spatial error estimate
        if any error indicator $> \epsilon_{sp}$ then
            begin
                group error points into different clusters
                form and refine rectangular megagrids
                do data transition and interpolation
                form polygrids for resolution
                update solution field outside the polygrids
                for $j := 1$ to number of polygrids do
                    begin
                        LocalAMR(polygrid($j$), $t$, $\Delta t$, $\epsilon_{sp}$, $\epsilon_t$)
                    end
            end
    end
```

FIG. 7. *Recursive local AMR procedure on polygrid G.*

of these procedures is presented in the following three algorithms, Figures 6–8. In the first algorithm, *Solve*, an initial base time step is chosen which will be modified when the program starts. A root grid which is also a polygrid is generated and will be refined adaptively. Initial solution is projected to the root grid with a L_2 projection. A domain decomposition procedure is performed to generate a finite number of subdomains (4 or 5 for two-dimensional problems) as well as a number of small local grids which are mostly in the size of 4×4 with a few smaller sizes near the megagrid boundaries. The solution proceeds from initial time, t_{start}, to the final time, t_{end}.

The second algorithm, *LocalAMR*, receives a solution from the algorithm *Schwarz* for one base time step and then performs an elementwise spatial error estimate. The elements with high error are grouped into different clusters. A polygrid is formed for each cluster and the solution procedure is repeated on each polygrid. The solution is updated before the resolution so that the solution outside the polygrids will not be

```
Procedure. Schwarz(polyG, $t$, $\Delta t$, $\epsilon_t$).
    begin
        for $i := 1$ to number of local grids of polyG do
            begin
                solve the ODE systems (10), local time steps are
                modified based on the tolerance $\epsilon_t$
                store the solutions to $w_i$
            end
        do start the iteration
            begin
                for $i := 1$ to number of local grids of polyG do
                    begin
                        solve the projection equations (11) to get $P_i u$
                        update the solution field using $w_i$, $P_i u$
                    end
                estimate the error from the Schwarz algorithm
                while(error > $\epsilon_t$) continue the iteration
            end
        choose the new $\Delta t$ for the next base time step
    end
```

FIG. 8. *Multiplicative Schwarz algorithm.*

lost.

The algorithm *Schwarz* is called on each polygrid. It solves the ODE systems (10) and stores the solutions on each local grid. A temporal error estimator controls the local time steps that are refined or derefined automatically. This procedure is described in a previous paper [22] in more detail. Then the projection equations (11) are solved on each local grid and the multiplicative Schwarz procedure is performed on the polygrid. We prescribe a temporal error tolerance that is a factor of at least 100 less than the prescribed spatial error tolerance. Thus temporal error will not affect the spatial refinement.

Alternatively, we may form the polygrids without creating clusters. To do this, we form a megagrid for each photo-cluster. The megagrid contains the photo-cluster with a chosen buffer zone and is aligned with the coordinate axes. We then refine the megagrid by a factor of three and interpolate the data field on the refined megagrid. We decompose the megagrid to get a list of small local grids. For each local grid on the list, we check if there is an error element inside or with a distance less than γ from it, where γ is the buffer size chosen as the mesh size of the parent megagrid. If not, we remove this local grid from the list. By going over all the local grids on the list, we end up with a polygrid. Example 5.3 uses this procedure for the refinement. The pseudo-Pascal algorithm, Figure 9, describes this process.

Arney and Flaherty [3], Moore [15], and Moore and Flaherty [16] used a scheme that performed spatial refinement for each fine time step. This significantly increased the overhead of their methods, since spatial refinement is one of the most expensive procedures. We have improved this by performing spatial refinement once per base time step. This increases efficiency significantly. Grid changes is generally insignificant within one base time step. Therefore, such a strategy is very effective.

Spatial derefinement is performed if the spatial error is too small. For a given

```
Procedure. PolygridMaker(Megagrid MG, List<errNode> &Eset)
    begin
        remove := 1
        for i := 1 to number of local grids of MG do
            begin
                for j := 1 to number of error points in Eset do
                    begin
                        if distance( err_point(j), grid(i) ) < $\gamma$) do
                            begin
                                remove := 0
                                break
                            end
                    end
                if(remove) remove grid(i)
            end
    end
```

FIG. 9. *An alternative procedure of generating polygrid.*

local grid, if the estimated error on each of the m ($9 \leq m \leq 16$) elements inside the grid is less than $\beta\epsilon_{sp}$ (we choose $\beta = 0.3$), we delete the lower left 3×3 fine elements (see Figure 2), and a coarse element is returned. The remaining three or four fine elements on the right-most column of this local grid are deleted only if the errors on the elements of the neighbor local grid to the right are all less than $\beta\epsilon_{sp}$. The same is true for the three or four elements on the top-most row of the local grid and the neighbor local grid on the top. By going through all the local grids, we have formed some polygrids which are used for the next level of integration.

4.2. Data structures. We keep track of the data fields and grid structures by maintaining an extended tree structure of the megagrids with sibling pointers. We choose to store the megagrids because of their simple structures. Data transition between different levels of megagrids are simple since they are all aligned with the coordinate axes and properly nested. A polygrid within a megagrid is stored as a list of local grids. A local grid is composed of m ($9 \leq m \leq 16$) elements and lists of nodes and edges. Finite element procedure and time integration are performed on local grids. Intergrid information is propagated through the multiplicative Schwarz algorithm. The base grid Δ^h (which is also a megagrid and a polygrid) is the root of the tree. The megagrids created by the first level of refinement process are considered the offspring of the base grid and are said to be at the first level of the tree. Since LocalAMR is recursive, the tree structure continues, with a megagrid at level i having a parent coarse megagrid at level $i - 1$ and possible finer offspring megagrids at level $i + 1$ for each megagrid. We define pointers to its parent, to its first son, and to its next sibling megagrid. Information at a parent level megagrid is passed to its sons through interpolation. Sibling megagrids at the same level have polygrids which are independent of each other. The tree structure grows if further refinement is needed and shrinks if derefinement is performed.

The data structure as well as the major objects such as grid, megagrid, and polygrid are easily manipulated by using the object-oriented programming language C++. Iterators are used for the many linked lists defined in the program. The use of iterators makes adding or deleting elements easier when constructing clusters and polygrids.

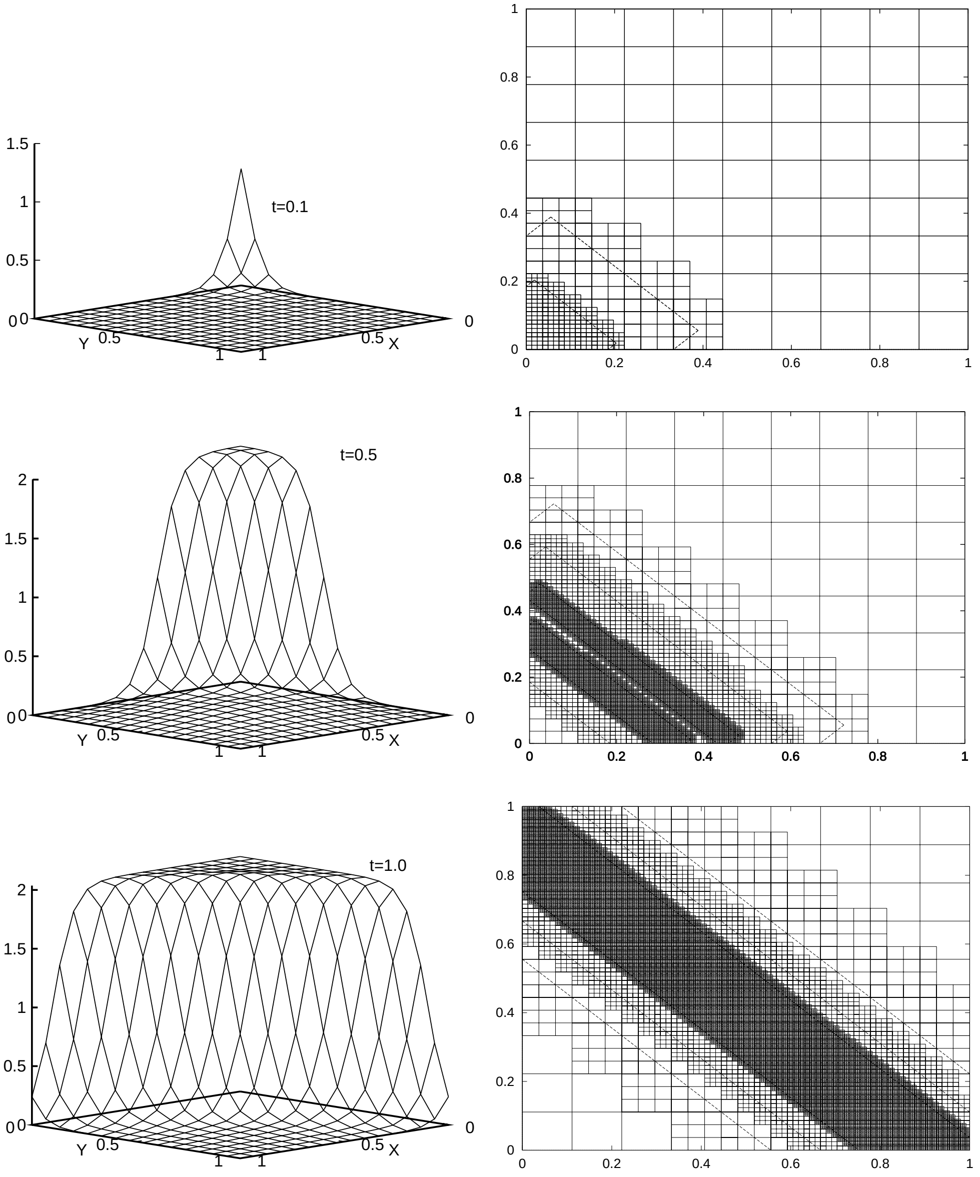


FIG. 10. *Surfaces and corresponding spatial meshes for Example* 5.1 *at* $t = 0.1, 0.5,$ *and* 1.0.

5. Numerical examples. We demonstrate the performance of our adaptive local space-time refinement procedure by applying our methods to three examples. The code is written in C++ and was run on a family of IBM RS6000 machines.

Example 5.1. Consider the forced heat conduction equation

$$u_t + f(x, y) = u_{xx} + u_{yy}, \qquad 0 \le x, y \le 1,\ 0 \le t \le t_f, \tag{12}$$

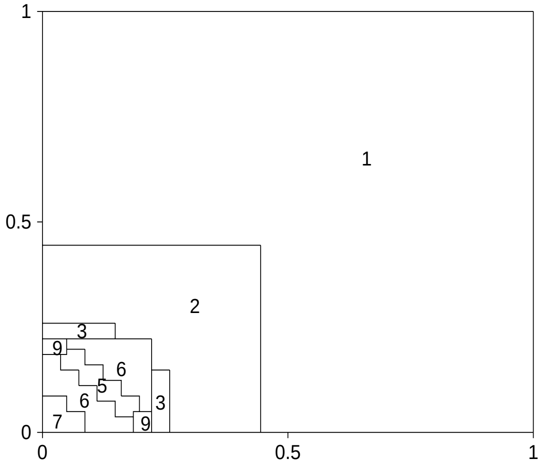


FIG. 11. *Number of time steps used in each region for Example* 5.1 *on* $0 \le t \le 0.1$.

with $f(t,x,y)$, the initial and boundary conditions specified so that the exact solution is

$$u(t,x,y) = 1.0 - \tanh[10(x+y-t+0.1)]. \tag{13}$$

We solved this problem for $0 \le t \le 1$. A 9×9 base spatial mesh with a uniform polynomial order 1 and an initial base time step of $\Delta t = 0.1$ are used. For the automatic error control, we use a spatial tolerance of 0.1 and a temporal error tolerance of 0.001. Surface plots and the adaptive meshes at $t = 0.1, 0.5$, and 1.0 are given in Figure 10. To get a better view of the surface plots, we rotated the coordinate plane 135^{o} clockwise. Figure 11 shows the number of time steps used in each spatial region for $0 \le t \le 0.1$ and a base time step $\Delta t = 0.1$. As can be seen from these figures, one large time step 0.1 and a very coarse spatial mesh are used in most of the domain. A fine mesh and a large number of time steps (9 in this case) are used only in a small fraction of the domain where changes are most rapid. Note that the time steps are not evenly divided. They are chosen automatically by the integration routine. The CPU times for the solution on $0 \le t \le 0.1$, with the same error tolerances, are 157.52 seconds with our space-time adaptive procedure and 1087.17 seconds using a spatially local adaptive procedure with uniform time stepping, a significant improvement in terms of computational efficiency.

Example 5.2. Consider the forced heat conduction equation (12) with $f(t,x,y)$ and the initial and boundary condition so that the exact solution is

$$u(t,x,y) = 0.8e^{-80[(x-r(t))^2+(y-s(t))^2]},$$

where

$$r(t) = 0.5 + 0.25 \sin \pi t$$

and

$$s(t) = 0.5 + 0.25 \cos \pi t.$$

The solution is a cone initially centered at $(0.5, 0.75)$ that rotates in a circle centered at $(0.5, 0.5)$ with a radius of 0.25 in a clockwise direction. The problem is solved on

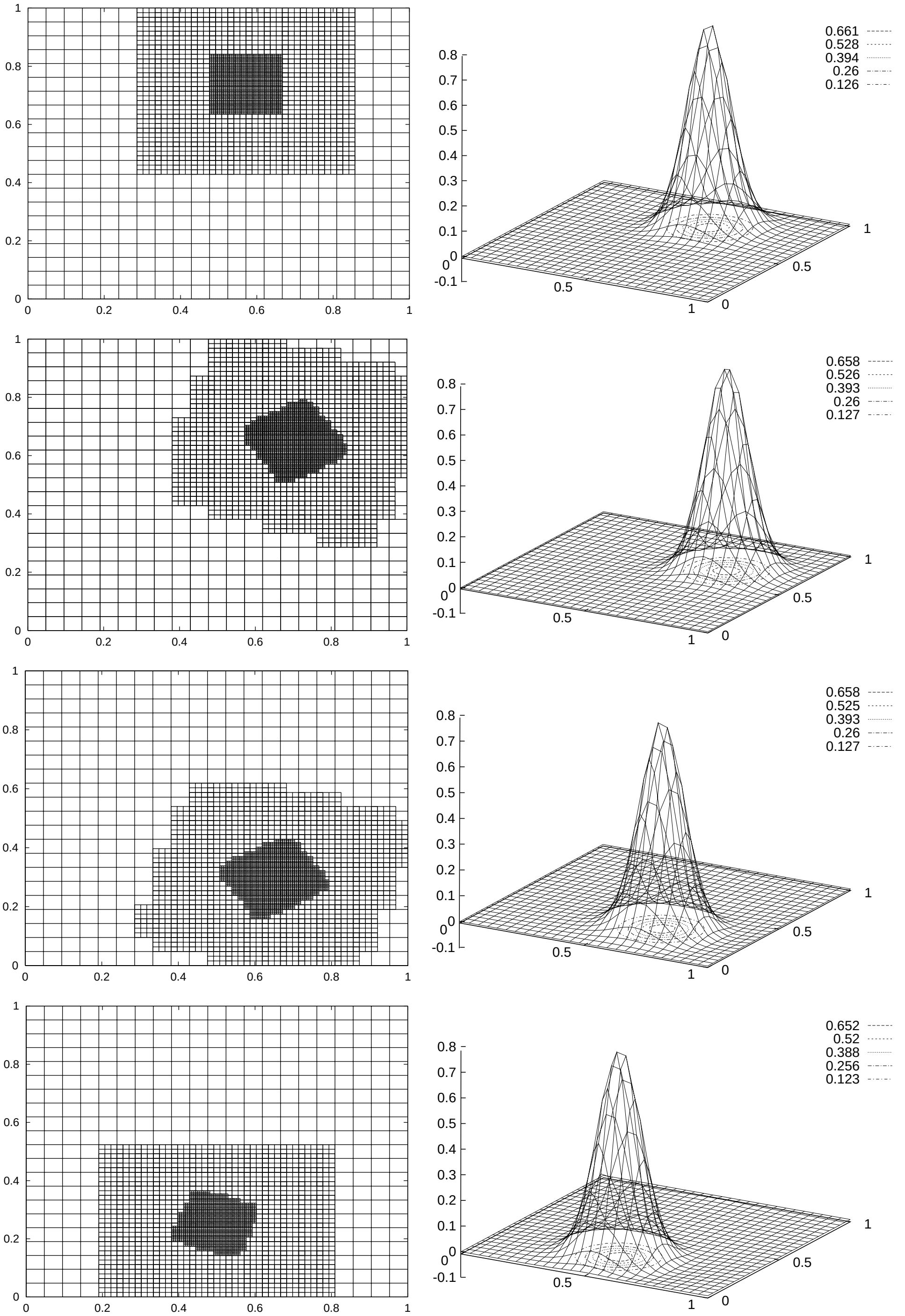


FIG. 12. *Spatial meshes and surface plots in Example* 5.2 *for* $t = 0.1, 0.3, 0.8,$ *and* 1.0.

Table 1
Convergence of multiplicative Schwarz algorithm.

# of iteration	H^1 error	# of iteration	H^1 error	# of iteration	H^1 error
1	0.829172	5	0.13051	9	0.0457369
2	0.420594	6	0.0936745	10	0.0412026
3	0.271475	7	0.0695346	11	0.039111
4	0.185898	8	0.0544481	12	0.0382874
				13	0.0380543

(a) On one-level decomposition.

# of iteration	H^1 error
1	0.148836
2	0.0396165
3	0.034211

(b) On two-level decomposition.

$0 \le t \le 1$ with an initial spatial grid of 21×21 and base time step of 0.1. The spatial error tolerance is 0.1 and the temporal error tolerance is 0.001. The grids and the surface plots at $t = 0.1, 0.3, 0.8$, and 1.0 are shown in Figure 12. The H^1 errors at 0.1, 0.3, 0.8, and 1.0 are 0.0204886, 0.0206745, 0.02065328, and 0.0204843, respectively.

To demonstrate the convergence of the multiplicative Schwarz algorithms, we solve the problem for one base time step of 0.1 starting at $t = 0.7$ using both one- and two-level decompositions. A 15×15 spatial grid and uniform elements of order 3 are used. Table 1 shows that the two-level algorithm needs many fewer iterations than the one-level method. The one-level algorithm takes 20m 0.26s CPU time, while the two-level algorithm takes 18m 20.09s, which is 90% of that needed for the one-level algorithm. Thus, the two-level algorithm is also slightly faster.

Example 5.3. Consider the combustion problem in Kapila [14]:

$$u_t - D(1+\alpha-u)e^{-\frac{\delta}{u}} = u_{xx} + u_{yy}, \ (x,y) \in \Omega, \qquad t > 0, \tag{14}$$

$$u(0,x,y) = 1, \qquad (x,y) \in \overline{\Omega}, \tag{15}$$

$$u(t,x,1) = 1, \ 0 \le x \le 1, \ t > 0, \ u(t,1,y) = 1, \qquad 0 \le y \le 1, \ t > 0, \tag{16}$$

$$u_x(t,x,0) = 0, \ 0 \le x \le 1, \ t > 0 \ u_y(t,0,y) = 0, \qquad 0 \le y \le 1, \ t > 0, \tag{17}$$

where

$$D = \frac{Re^{\delta}}{\alpha\delta}$$

is the Damkohler number, δ is the activation energy, α is the heat release, and R is the reaction rate. The temperature increases slowly with a hot spot forming at the origin. After some time, ignition occurs and the temperature at $(0,0)$ jumps rapidly. A sharp reaction front forms and propagates toward the boundaries $x = 1$ and $y = 1$ where boundary layers form.

We solve (14)–(17) with $\alpha = 1$, $\delta = 20$, and $R = 5$. An initial spatial grid of 9×9 and base time step of 0.1 with uniform element of order 1 is used. Spatial and temporal tolerances are chosen to be 0.1 and 0.001. Surface plots at $t = 0.25, 0.28, 0.29, 0.30, 0.32, 0.35$ are shown in Figure 13. Figure 14 shows the refined spatial grids at $t = 0.28, 0.30, 0.32$, and 0.35. The number of time steps used for one base time step are shown in Figure 15.

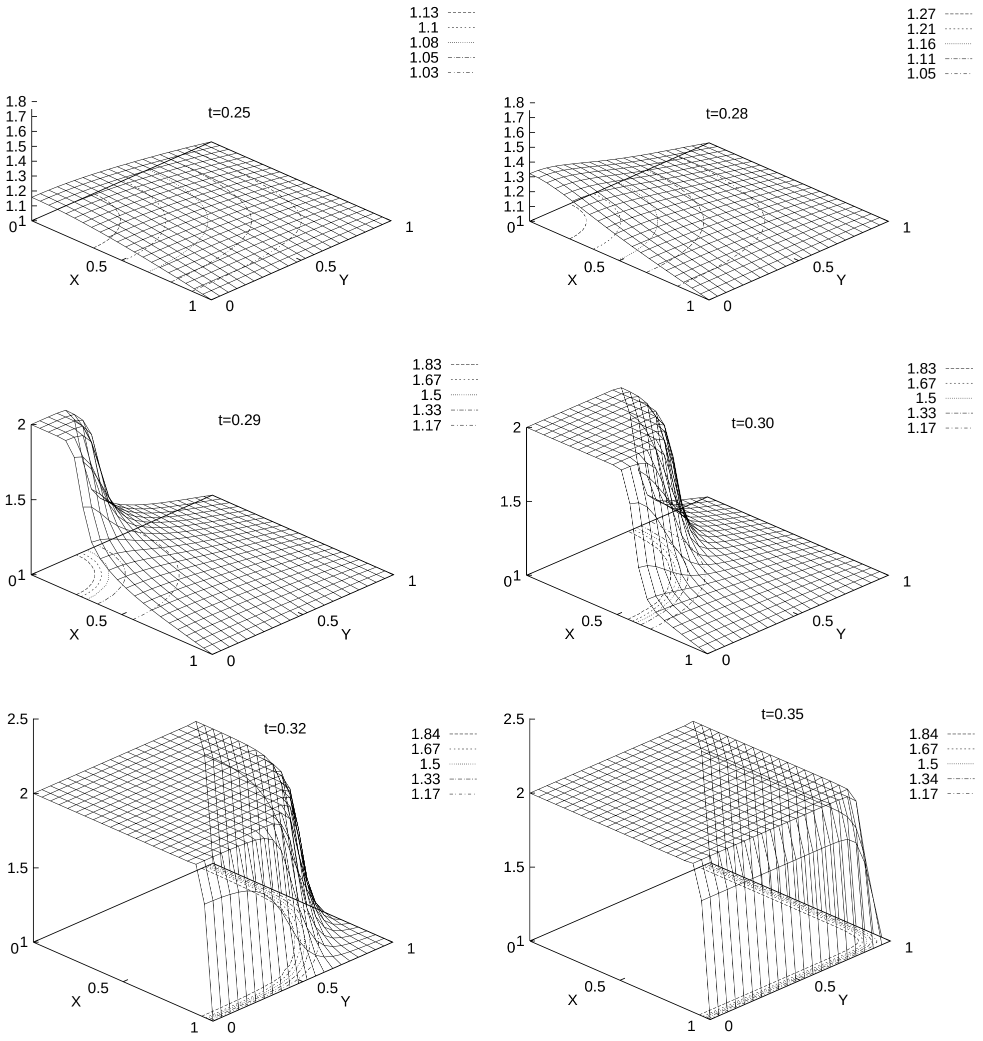


FIG. 13. *Surface and contour plots of Example* 5.3 *at* $t = 0.25, .28, .29, .30, .32, and .35$.

From these plots, we conclude that the temperature increases slowly from $t = 0$ to $t = 0.28$, where very coarse spatial mesh and time steps are used. At about $t = 0.28$, ignition occurs, the temperature at the origin increases rapidly to 2, and a sharp reaction front forms and propagates toward the Dirichlet boundaries $x = 1$ and $y = 1$. The space-time meshes follow the evolution process closely. We also note that, when the reaction front is close to the Dirichlet boundaries $x = 1$ and $y = 1$, the spatial mesh is fine near these boundaries since the spatial gradient is large there. But the changes with time become slow except at the corner $(1, 1)$. Thus, large time steps are used near these boundaries. Smaller time steps are used only near the corner as indicated in the plots for $0.34 \leq t \leq 0.35$.

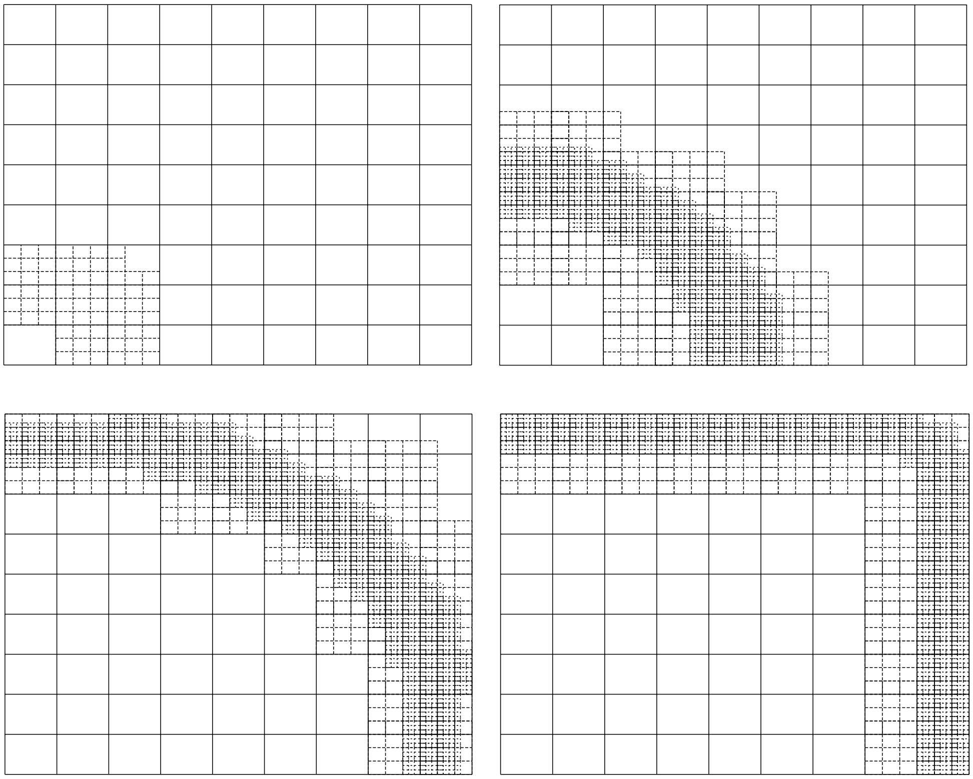


FIG. 14. *Spatial mesh used for Example* 5.3 *at* $t = .29, .30, .32,$ *and* .35.

We run this example from $t = 0.0$ to $t = 0.35$ with local space-time adaptive meshes and get a CPU time of 5 hours and 1763 seconds. We also run the example on the same interval with the same spatial and temporal error tolerances but with local adaptive spatial mesh and uniform time stepping; the CPU time is 51 hours and 3354 seconds, which is almost ten times longer. In most part of the domain, one step is enough to advance one base time step if our algorithm is used, while dozens of steps are needed if uniform time stepping is used.

6. Concluding remarks. An adaptive local space-time mesh refinement procedure was described for solving parabolic partial differential equations in two space dimensions. The test problems showed that fine time steps are used in regions with fine spatial meshes. The fine meshes follow the dynamics of the physical problems. The test problems given are all on rectangular domains, but the methods can be generalized to any domain with piecewise linear boundaries. Thus, with an appropriate mapping, we can generalize the method to solving any two-dimensional parabolic problem on an arbitrary domain. We will also generalize the methods to three-dimensional problems with some trivial changes. Since one of the most important benefits of domain decomposition methods is the ease with which the code can be run on parallel machines, an effort is being made in making the code running parallel.

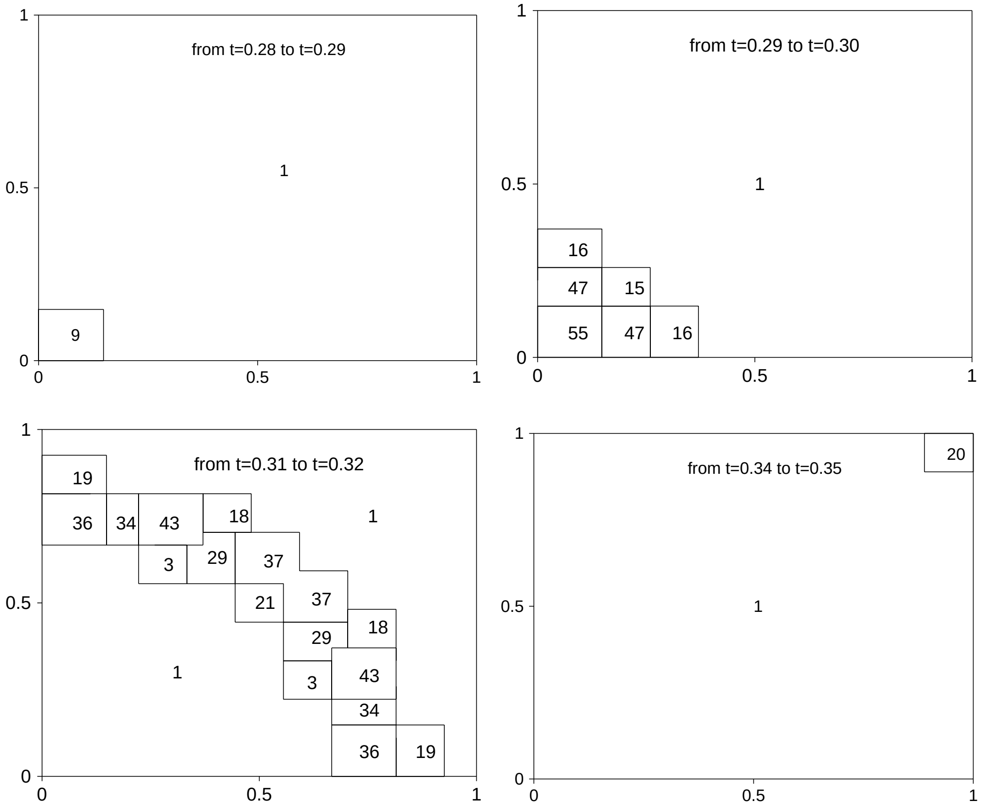


FIG. 15. *Number of time steps used for Example* 5.3 *start at* $t = .28, .29, .31,$ *and* .34 *with a base step* 0.01.

Acknowledgment. I wish to thank my thesis advisor Dr. Peter Moore for his inspiring guidance and constant encouragement.

REFERENCES

[1] S. ADJERID, J. E. FLAHERTY, AND Y. J. WANG, *Adaptive Method-of-Lines Techniques for Parabolic Systems*, SCOREC report 17-1993, Rensselaer Polytechnic Institute, Troy, NY.

[2] S. ADJERID, I. BABUSKA, AND J. E. FLAHERTY, *A posteriori error estimation for the finite element method of lines solution of parabolic systems*, Math. Models Methods Appl. Sci., to appear.

[3] D. C. ARNEY AND J. E. FLAHERTY, *An adaptive local mesh refinement method for time-dependent partial differential equations*, Appl. Numer. Math., 5 (1989), pp. 257–274.

[4] I. BABUSKA AND W. GUI, *Basic Principles of Feedback and Adaptive Approaches in the Finite Element Method*, Tech. note BN-1042, Institute for Physical Science and Technology, University of Maryland, College Park, MD, 1985.

[5] M. J. BERGER AND J. OLIGER, *Adaptive mesh refinement for hyperbolic partial differential equations*, J. Comput. Phys., 53 (1984), pp. 484–512.

[6] M. J. BERGER AND J. S. SALTZMAN, *AMR on the CM-*2, Appl. Numer. Math., 14 (1994), pp. 239–253.

[7] M. BIETERMAN AND I. BABUSKA, *An adaptive method of lines with error control for parabolic equations of the reaction-diffusion type*, J. Comput. Phys., 63 (1986), pp. 33–66.

[8] J. H. BRAMBLE, J. E. PASCIAK, J. WANG, AND J. XU, *Convergence estimates for product iterative methods with applications to domain decomposition and multigrid*, Math. Comp., 57 (1991), pp. 1–22.

[9] X.-C. Cai, *Multiplicative Schwarz Methods for parabolic problems*, SIAM J. Sci. Comput., 15 (1994), pp. 587–603.

[10] X.-C. Cai and O. B. Widlund, *Multiplicative Schwarz algorithms for nonsymmetric and indefinite problems*, SIAM J. Numer. Anal., 30 (1993), pp. 936–952.

[11] T. F. Chan and T. P. Mathew, *Domain decomposition algorithms*, Acta Numer., Cambridge University Press, Cambridge, UK, 1994, pp. 61–143.

[12] P. Deuflhard, *Order and stepsize control in extrapolation methods*, Numer. Math., 41 (1983), pp. 399–422.

[13] M. Dryia and O. B. Widlund, *Towards a unified theory of domain decomposition algorithms for elliptic problems*, in Proceedings of the Third International Symposium on Domain Decomposition Methods for Partial Differential Equations, Proceedings in Appl. Math. 43, T. F. Chan, R. Glowinski, J. Periaux, and O. B. Widlund, eds., SIAM, Philadelphia, 1990, pp. 3–21.

[14] A. K. Kapila, *Asymptotic Treatment of Chemically Reacting Systems*, Pitman, Boston, 1983.

[15] P. K. Moore, *Comparison of adaptive methods for one-dimensional parabolic systems*, Appl. Numer. Math., 16 (1995), pp. 471–488.

[16] P. Moore and J. E. Flaherty, *Adaptive local overlapping grid methods for parabolic systems in two space dimensions*, J. Comput. Physics, 98 (1992), pp. 54–63.

[17] P. K. Moore, *A Local Adaptive Finite Element Method for Solving One- and Two-Dimensional Systems of Parabolic Partial Differential Equations*, Ph.D. Thesis, Rensselaer Polytechnic Institute, Troy, NY, 1988.

[18] P. K. Moore and J. E. Flaherty, *High-order adaptive finite element—singly implicit Runge-Kutta methods for parabolic differential equations*, BIT, 33 (1993), pp. 309–331.

[19] J. J. Quirk, *A parallel adaptive grid algorithm for computational shock hydrodynamics*, Appl. Numer. Math., 20 (1996), pp. 427–453.

[20] B. A. Szabo and I. Babuska, *Introduction to Finite Element Analysis*, John Wiley, New York, 1991.

[21] J. Xu, *Iterative methods by space decomposition and subspace correction*, SIAM Rev., 34 (1992), pp. 581–613.

[22] H. Yu, *Solving parabolic problems with different time steps in different regions in space based on domain decomposition methods*, Appl. Numer. Math., 30 (1999), pp. 475–491.

[23] H. Yu, *A Local Space-Time AMR Procedure on Parallel Machines*, in preparation.

SIAM J. SCI. COMPUT. © 2001 Society for Industrial and Applied Mathematics
Vol. 23, No. 1, pp. 323–338

ENUMERATION OF ALL EXTREME EQUILIBRIA OF BIMATRIX GAMES*

CHARLES AUDET†, PIERRE HANSEN‡, BRIGITTE JAUMARD§, AND GILLES SAVARD§

Abstract. The set of equilibrium points of a bimatrix game is the union of polytopes that are not necessarily disjoint. Knowledge of the vertices of these polytopes (extreme equilibria) is sufficient to identify all equilibria. We present an algorithm that enumerates all extreme equilibria by exploiting complementary slackness optimality conditions of two pairs of parameterized linear programming problems. The algorithm is applied to randomly generated problems of size up to 29×29 when both dimensions are equal, and up to 700×5 when the second dimension is fixed. The number of extreme equilibria grows exponentially with problem size but remains moderate for the instances considered. Therefore, the results could be useful for further study of refinements of Nash equilibria.

Key words. bimatrix game, extreme equilibrium, implicit enumeration, two-person nonzero-sum game

AMS subject classifications. 90D05, 90C27

PII. S1064827598339086

1. Introduction.

A bimatrix game (or two-person nonzero-sum game in strategic form) may be stated as follows: given a pair of $m \times n$ payoff matrices A and B, Player I attempts to maximize his payoff x^tAy by selecting the probability vector x in $\mathbb{R}^m$, and simultaneously, Player II attempts to maximize his payoff x^tBy by choosing the probability vector y in $\mathbb{R}^n$.

It is well known that there is always at least one *equilibrium point* (see Nash [18]) in such a game, i.e., a pair of strategies $(\hat{x}, \hat{y})$ that simultaneously solve the two parameterized linear programs

$$(\text{P1}) \quad \text{subject to} \quad \begin{array}{cl} \max\limits_{x} & x^tA\hat{y} \\ (\text{s.t.}) & x^t\mathbf{1} = 1, \\ & x \geq \mathbf{0}, \end{array} \qquad \text{and} \qquad (\text{P2}) \quad \begin{array}{cl} \max\limits_{y} & \hat{x}^tBy \\ \text{s.t.} & \mathbf{1}^ty = 1, \\ & y \geq \mathbf{0}, \end{array}$$

where $\mathbf{0}$ and $\mathbf{1}$, respectively, denote vectors whose components are all equal to zero and one. The strategy $\hat{x}$ is thus a best response to $\hat{y}$, as it is an optimal solution of (P1); and $\hat{y}$ is a best response to $\hat{x}$, as it is an optimal solution of (P2). Hence at

*Received by the editors May 19, 1998; accepted for publication (in revised form) November 3, 2000; published electronically June 26, 2001.

http://www.siam.org/journals/sisc/23-1/33908.html

†École Polytechnique de Montréal, Département de Mathématiques et de Génie Industriel, C.P. 6079, Succursale “Centre-Ville,” Montréal, Québec, H3C 3A7, Canada (charles@crt.umontreal.ca). The work of this author was supported by a Natural Sciences and Engineering Research Council (NSERC) fellowship. Part of the work of this author was done during a postdoctoral stay at Rice University.

‡École des Hautes Études Commerciales and GERAD, 3000 chemin de la Côte-Sainte-Catherine, Montréal, Québec, H3T 2A7, Canada (pierreh@crt.umontreal.ca). The work of this author was supported by Fonds pour la Formation des Chercheurs et l’Aide à la Recherche (FCAR) grant 95ER1048 and NSERC grant GP0105574.

§École Polytechnique de Montréal and GERAD, Département de Mathématiques et de Génie Industriel, C.P. 6079, Succursale “Centre-Ville,” Montréal, Québec, H3C 3A7, Canada (brigitt@crt.umontreal.ca, gilles@crt.umontreal.ca). The work of the third author was supported by FCAR grant 95ER1048 and NSERC grant GP0036426. The work of the fourth author was supported by NSERC grant OGP0046405 and FCAR grant 93ER0141.

an equilibrium point neither player has an incentive to change his strategy unless the other player does so.

Linear programming duality theory yields the equivalent dual programs

$$(\mathrm{D1})\qquad \begin{array}{ll} \min\limits_{\alpha} & \alpha \\ \text{s.t.} & \mathbf{1}\alpha \geq A\hat{y}, \end{array} \qquad \text{and} \qquad (\mathrm{D2})\qquad \begin{array}{ll} \min\limits_{\beta} & \beta \\ \text{s.t.} & \beta\mathbf{1}^t \geq \hat{x}^t B, \end{array}$$

where α and β are dual variables in $\mathbb{R}$.

There are two classical and equivalent necessary and sufficient optimality conditions in linear programming theory and thus two equilibrium conditions in our framework. Both of these conditions require primal and dual feasibility: the pair of strategies $(\hat{x}, \hat{y})$ together with scalars $\hat{\alpha}$ and $\hat{\beta}$ must satisfy

$$\begin{aligned} (\hat{x}, \hat{\beta}) &\in X = \{(x, \beta) \in \mathbb{R}^{m+1} :\ x^t B \leq \beta\mathbf{1}^t,\ x^t\mathbf{1} = 1,\ x \geq \mathbf{0}\}, \\ (\hat{y}, \hat{\alpha}) &\in Y = \{(y, \alpha) \in \mathbb{R}^{n+1} :\ Ay \leq \mathbf{1}\alpha,\ \mathbf{1}^t y = 1,\ y \geq \mathbf{0}\}. \end{aligned}$$

The first optimality condition consists in the equality of primal and dual objective function values

$$(1.1)\qquad \hat{x}^t A\hat{y} = \hat{\alpha} \qquad \text{and} \qquad \hat{x}^t B\hat{y} = \hat{\beta}.$$

This means that the values of the dual variables must be equal to the payoffs. The second optimality condition is the complementary slackness condition

$$(1.2)\qquad \hat{x}^t(\mathbf{1}\hat{\alpha} - A\hat{y}) = 0 \qquad \text{and} \qquad (\hat{\beta}\mathbf{1}^t - \hat{x}^t B)\hat{y} = 0.$$

Equivalence of optimality conditions (1.1) and (1.2) appears when substituting $\hat{x}^t\mathbf{1}$ and $\mathbf{1}^t\hat{y}$ by 1.

There may be one, many, or an infinite number of equilibrium points. Indeed, for a given strategy $\hat{y}$, the set of optimal responses of Player I, i.e., the set of optimal solutions of (P1), denoted $X(\hat{y})$, is either a singleton or a polytope. The symmetric observation holds for (P2), where the set of optimal solutions is denoted $Y(\hat{x})$. In fact, the set E of all equilibrium points is the union of a finite number of polytopes called *maximal Nash subsets* [14]. Different equilibria possess different characteristics which make them unequally attractive. Identification of all equilibria of a bimatrix game would ease further refinement and classification of them. Discussion of equilibrium refinements can be found in Kohlberg [7] and in Van Damme [19], [20].

The contribution of this paper is to present a new algorithm that enumerates all vertices of every maximal Nash subset of a bimatrix game. Knowledge of these *extreme equilibria* allows identification of all equilibrium points.

The key feature of the algorithm is that it enumerates only a subset of the vertices of the polyhedra X and Y. To this effect, the complementary slackness conditions (1.2) of the primal and dual problems are exploited while the enumeration proceeds, and this allows early pruning of many branches of the search tree. Nondegeneracy of the bimatrix game is not required in order to show that the proposed algorithm enumerates in finite time all extreme equilibria.

The paper is organized as follows. The next section briefly reviews previous work on enumeration of equilibrium points of bimatrix games. Section 3 presents a detailed description of an algorithm that enumerates all extreme equilibria. Numerical results for much larger problems than those previously solved are presented in section 4.

2. Previous work on enumeration of equilibria. The classical algorithm to find a single equilibrium point of a bimatrix game is that of Lemke and Howson [9]. It is a path-following method designed primarily for nondegenerate games, but it can also be applied to degenerate games after a slight perturbation. Aggarwal [1] shows that this algorithm cannot be directly modified to enumerate all extreme equilibria, as some of them may be inaccessible to the path.

A complete listing of equilibrium points is impossible, as their number may be infinite, since Nash subsets are polytopes. However, given the vertices of all maximal Nash subsets (the extreme equilibria), one can fully describe the nonconvex set E of all equilibria. Let S_x be a set containing any number of extreme strategies of Player I, and let $\hat{x}$ be a convex combination of these strategies. Vorob'ev [23] shows that if $\hat{y}$ solves (P2) for each extreme strategy of S_x, then $(\hat{x}, \hat{y})$ is an equilibrium point. Let $Y(x)$ denote the set of optimal solutions of (P2) for $\hat{x} = x$. The set E is obtained by considering the union of such sets S_x over all possible subsets of extreme strategies of Player I:

$$E = \bigcup_{S_x \subseteq \{x:(x,y) \in V\}} \left\{ \operatorname{conv}(S_x) \times \bigcap_{x \in S_x} Y(x) \right\},$$

where V is the set of all extreme equilibria, and $\operatorname{conv}(S_x)$ denotes the convex envelope of S_x.

From symmetry we can also obtain the following characterization of E:

$$E = \bigcup_{S_x \subseteq vert(X),\, S_y \subseteq vert(Y),\, S_x \times S_y \subseteq E} \operatorname{conv}(S_x) \times \operatorname{conv}(S_y),$$

which shows that the equilibrium condition need only be checked for extreme points of X and Y, expressed by the conditions under the $\bigcup$ sign. See Winkels [24] and Jansen [6] for additional information.

In order to obtain all equilibrium points, it is therefore sufficient to find all extreme equilibria. The problem addressed in this paper is to enumerate all of them. Only a few algorithms are designed to do so. We briefly present their main ideas. The first one is due to Vorob'ev [23] and was later simplified by Kuhn [8], in which extreme equilibria are characterized in terms of square submatrices of the payoff matrices.

Mangasarian [11] suggests enumerating all vertices of X and Y (using Balinski's algorithm [2]) and then checking for each pair of vertices x of X and y of Y if the bilinear optimality condition $x^t(A+B)y = \alpha + \beta$ is satisfied. Mills [15] and Mangasarian and Stone [12] show that this condition is equivalent to (1.1). Winkels [24] proposes an algorithm which differs from the previous one only in the optimality test: it consists in checking the complementary slackness conditions (1.2). Mukhamediev [17] enumerates all vertices of a linear maxmin problem derived from the above mentioned bilinear optimality condition.

Another algorithm consists in enumerating each of the $(2^n - 1)(2^m - 1)$ possible supports (the sets of pure strategies that are assigned a positive probability) and then checking for equilibria. For a given support, the set of equilibria can either be empty, a singleton, or a polytope. In this last case, all vertices of that polytope must be enumerated. Different implementations are discussed in Dickhaut and Kaplan [3] and in McKelvey and McLennan [13].

Von Stengel [22] is currently exploring an algorithm that enumerates the vertices of X only while directly computing the complementary vertices of Y (if they exist). This is advantageous if one player has much less pure strategies than the other.

All these methods rely more or less directly on enumeration of all vertices of polyhedra. The number of such vertices typically increases exponentially with problem size and can be high even for low dimensional polyhedra. Moreover, important computational difficulties arise when degeneracy is encountered, i.e., when there are multiple basic representations of the same vertex. See Von Stengel [21] for equivalent definitions of degeneracy. For more information on computation of equilibria in finite games, the reader can refer to the recent surveys of McKelvey and McLennan [13], and Von Stengel [21].

Note that if X and Y are nondegenerate, it follows from the facts that (i) Dyer's [4] vertex enumeration algorithm for polyhedra is polynomial in input size and number of vertices, and (ii) the equilibrium conditions can be checked in polynomial time, that a similar property holds for Mangasarian's algorithm [11], after substitution of the vertex enumeration routine. We do not know if such a property holds for the algorithm proposed in this paper.

3. Enumeration method. Enumeration of all extreme equilibria is done through exploration of a search tree. To each node of the tree correspond two linear subproblems whose feasible regions are derived from X and Y. They differ through conversion of inequalities into equalities. Two types of branching rules govern these conversions.

The first type of rule relies on an equivalent formulation of the complementary slackness condition (1.2). At an equilibrium point, the feasibility conditions insure that the vectors $\hat{x}$ and $\hat{y}$ are nonnegative and that $\hat{x}^t B \leq \hat{\beta}\mathbf{1}^t$ and $A\hat{y} \leq \mathbf{1}\hat{\alpha}$. Therefore, condition (1.2) can be written through the $m+n$ complementary slackness conditions

$$\hat{x}_i = 0 \ \text{ or } \ A_{i\cdot}\hat{y} = \hat{\alpha} \quad \text{and} \quad \hat{y}_j = 0 \ \text{ or } \ \hat{x}^t B_{\cdot j} = \hat{\beta}, \tag{3.1}$$

where $A_{i\cdot}$ denotes the ith row of A and $B_{\cdot j}$ denotes the jth column of B for $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$. The first type of branching rule splits the current node of the tree through two branches. In one branch, a pure strategy is forced to be played with probability zero (a variable is fixed to zero), and in the second, a pure strategy is forced to have maximum payoff, so it is a best response (a slack variable is fixed to zero). This rule is invoked by the algorithm until all complementary slackness conditions (3.1) are forced to be satisfied and thus for all nodes whose depth in the search tree is less than or equal to $m + n$.

When the depth of the current node exceeds $m+n$, the second type of branching rule is used. At such a node, all complementary slackness conditions (3.1) are guaranteed to be satisfied by the first type of branching rule, and thus any solution of the current converted equalities is necessarily an equilibrium point. This branching rule consists in splitting the current node through as many branches as there are strictly positive variables and slack variables of X and Y corresponding to the equilibrium point. In each branch the variable or slack variable is fixed to zero. All degenerate solutions are reached, possibly several times, and thus elimination of duplicates is required.

Backtracking occurs when conversion of inequalities into equalities reduces the feasible region to the empty set. Let us now introduce more formal definitions in order to fully describe the algorithm.

3.1. Notation. As discussed above, every new branch is obtained by converting a single inequality of X or Y to an equality. In order to efficiently select which inequality to take, we introduce the following linear programming problems, parameterized in the objective functions

$$P(y) \equiv \max_{(x,\beta)\in X} x^tAy - \beta,$$
$$Q(x) \equiv \max_{(y,\alpha)\in Y} x^tBy - \alpha.$$

Problem $P(y)$ is an aggregation of the primal and dual problems (P1) and (D2), and $Q(x)$ is an aggregation of (P2) and (D1). Indeed, the constraints appearing in the definition of X and Y are those of the primal and dual problems. Moreover, the objective functions $x^tAy - \beta$ and $x^tBy - \alpha$ are the sum of the primal and dual objective functions. Therefore, it is likely that an optimal solution of $P(y)$ solves (P1) and (D2), and one of $Q(x)$ solves (P2) and (D1). The search for equilibria is done through the feasible regions X and Y, and thus any objective functions could be used. The intuitive motivation behind this choice of objective functions is to guide the algorithm toward equilibrium points.

At each node of the search tree is associated a pair of current subproblems $\tilde{P}$ and $\tilde{Q}$ that are identical to P and Q, except that some of the inequality constraints or nonnegativity constraints are converted to equalities. The depth of the root node is as follows.

1. At each node, one of the three following cases occurs as follows:
   (i) Either $\tilde{P}$ or $\tilde{Q}$ is infeasible.
   (ii) Both $\tilde{P}$ and $\tilde{Q}$ are feasible, but there is not enough information to obtain an equilibrium point. (The depth of the current node is less than or equal to $m + n$.)
   (iii) Both $\tilde{P}$ and $\tilde{Q}$ are feasible and there is sufficient information to deduce an extreme equilibrium point. (The depth of the current node is greater than $m + n$.)

In the first case, the current node is discarded. In the two other cases, the current node is split into others through new branches. The subproblems of the new nodes are identical to the current ones except for one additional constraint converted into an equality in either $\tilde{P}$ or $\tilde{Q}$, chosen according to the branching rules. Thus for each new branch, only one of the two subproblems needs to be checked for feasibility.

In the first type of branching rule, the selected pair of complementary inequalities to be converted into equalities corresponds to the largest complementary product $x_i(\alpha - A_{i\cdot}y)$ or $(\beta - x^tB_{\cdot j})y_j$, where $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$. Of course, the pair of inequalities must be chosen among those that were not already converted in $\tilde{P}$ or $\tilde{Q}$.

A node is not only characterized by the pair of current subproblems $\tilde{P}$ and $\tilde{Q}$. One of the two vectors x or y is also associated to each node. This vector describes a feasible strategy for the current subproblem that does not differ from its predecessor in the tree. We now introduce the notation that specifies which constraint is converted. This notation will be used when formally defining both types of branching rule. Given the current subproblems $\tilde{P}$ and $\tilde{Q}$, we define

P^i to be $\tilde{P}$ in which the constraint $x_i \geq 0$ is converted to $x_i = 0$,

P_j to be $\tilde{P}$ in which the constraint $x^tB_{\cdot j} \leq \beta$ is converted to $x^tB_{\cdot j} = \beta$,

Q^j to be $\tilde{Q}$ in which the constraint $y_j \geq 0$ is converted to $y_j = 0$, and

Q_i to be $\tilde{Q}$ in which the constraint $A_{i\cdot}y \leq \alpha$ is converted to $A_{i\cdot}y = \alpha$,

where $i \in \{1, 2, \ldots, m\}$ and $j \in \{1, 2, \ldots, n\}$.

3.2. Algorithm EEE (enumeration of extreme equilibria). The algorithm is now briefly stated, followed by a more intuitive discussion of several important steps and a proof of its validity.

Step a. Initialization.
Initialize the set of nodes to be explored to $T = \{(x, P, Q)\}$, where x is set to an arbitrary feasible value (typically, $x = (\frac{1}{m}, \frac{1}{m}, \dots, \frac{1}{m})$). Go to Step b.

Step b. Selection of a node.
If the set T of nodes to be explored is empty, then stop. Otherwise, choose and remove a node N from T. Either $N = (x, \tilde{P}, \tilde{Q})$ or $N = (y, \tilde{P}, \tilde{Q})$. In the first case go to Step c; in the second case, go to Step d.

Step c. Direct feasibility test ($\tilde{\boldsymbol{Q}}$).
If $\tilde{Q}(x)$ is infeasible, then go to Step b; otherwise, choose $(\tilde{y}, \tilde{\alpha})$ that solves $\tilde{Q}(x)$ and $(\tilde{x}, \tilde{\beta})$ that solves $\tilde{P}(\tilde{y})$. If the depth of the current node is not greater than $m + n$, go to Step e; otherwise, go to Step f.

Step d. Direct feasibility test ($\tilde{\boldsymbol{P}}$).
If $\tilde{P}(y)$ is infeasible, then go to Step b; otherwise, choose $(\tilde{x}, \tilde{\beta})$ that solves $\tilde{P}(y)$ and $(\tilde{y}, \tilde{\alpha})$ that solves $\tilde{Q}(\tilde{x})$. If the depth of the current node is not greater than $m + n$, go to Step e; otherwise, go to Step f.

Step e. Dichotomous branching.
Let

$$\tau_i = \begin{cases} \tilde{x}_i(\alpha - A_{i\cdot}\tilde{y}) & \text{if the variable } x_i \text{ of } \tilde{P} \text{ is not forced to be null, and} \\ & \text{the constraint } A_{i\cdot}y \le \alpha \text{ of } \tilde{Q} \text{ is not fixed at equality,} \\ -1 & \text{otherwise,} \end{cases}$$

$$\pi_j = \begin{cases} (\beta - \tilde{x}^t B_{\cdot j})\tilde{y}_j & \text{if the variable } y_j \text{ of } \tilde{Q} \text{ is not forced to be null,} \\ & \text{the constraint } x^t B_{\cdot j} \le \beta \text{ of } \tilde{P} \text{ is not fixed at equality,} \\ -1 & \text{otherwise,} \end{cases}$$

and then select the indices ĩ and j̃ that maximize the values τ_i and π_j (ties are broken arbitrarily) for $i = 1, 2, \dots, m$ and $j = 1, 2, \dots, n$. In the case where $\tau_{\tilde{\imath}} \ge \pi_{\tilde{\jmath}}$, add

$$(\tilde{y}, P^{\tilde{\imath}}, \tilde{Q}) \text{ and } (\tilde{x}, \tilde{P}, Q_{\tilde{\imath}})$$

to T, and in the case where $\tau_{\tilde{\imath}} < \pi_{\tilde{\jmath}}$, add

$$(\tilde{x}, \tilde{P}, Q^{\tilde{\jmath}}) \text{ and } (\tilde{y}, P_{\tilde{\jmath}}, \tilde{Q})$$

to T. Go to Step b.

Step f. Polychotomous branching.
The solution $(\tilde{x}, \tilde{y})$ is an equilibrium strategy since the complementary slackness conditions are satisfied. Record it on the list of equilibria if not already there, and add to T all nodes corresponding to a strictly positive variable or slack variable. These nodes are in the four sets

$$\{(\tilde{y}, P^i, \tilde{Q}) : \tilde{x}_i > 0\}, \qquad \{(\tilde{x}, \tilde{P}, Q_i) : A_{i\cdot}\tilde{y} < \alpha\},$$
$$\{(\tilde{x}, \tilde{P}, Q^j) : \tilde{y}_j > 0\}, \text{ and } \{(\tilde{y}, P_j, \tilde{Q}) : \tilde{x}^t B_{\cdot j} < \beta\}.$$

Go to Step b.

We now discuss each step of the enumeration method. The initialization step places the original bimatrix game into the set of nodes to be explored, together with a feasible strategy for Player I, and sets the depth of this node to one.

The order in which the nodes are chosen from the set T (in Step b) is not important since all of them must eventually be selected to ensure that no extreme equilibria are missed. Our implementation uses a depth-first strategy.

Steps c and d are the backtracking steps. In each of them, only one subproblem among $\tilde{P}$ and $\tilde{Q}$ is checked for feasibility, as only one problem differs in its feasible region from its predecessor (identification of the modified problem is done through the vector x or y associated to the node). When the feasible region is nonempty, these steps end with a feasible strategy $(\tilde{x}, \tilde{y})$. The parameterized linear programs are solved by a Simplex algorithm, and so the solutions are vertices of the polyhedra X and Y. (This might not be so with an interior point algorithm.) In order to reduce the number of pivots, the optimal bases found when solving $\tilde{P}$ and $\tilde{Q}$ are kept in memory and reused at the next node.

The last two steps define the branching structure of the algorithm. New nodes are added to the search tree. Step e is invoked when the depth of the current node in the search tree is less than or equal to $m+n$ (i.e., when at least one of the complementary conditions is not forced to be satisfied). It follows that $\tau_{\bar{\imath}}$ and $\pi_{\bar{\jmath}}$ will not both be equal to -1. Based on the complementary slackness conditions, two new branches are created: one in which either a variable $x_{\bar{\imath}}$ is fixed at 0 or a constraint $x^t B_{\cdot\bar{\jmath}} \leq \beta$ is replaced by $x^t B_{\cdot\bar{\jmath}} = \beta$, and the other in which either a variable $y_{\bar{\jmath}}$ is fixed at 0 or a constraint $A_{\bar{\imath}\cdot} y \leq \alpha$ is replaced by $A_{\bar{\imath}\cdot} y = \alpha$. Figure 3.1 summarizes this dichotomous branching rule using the above notation for the subproblems.

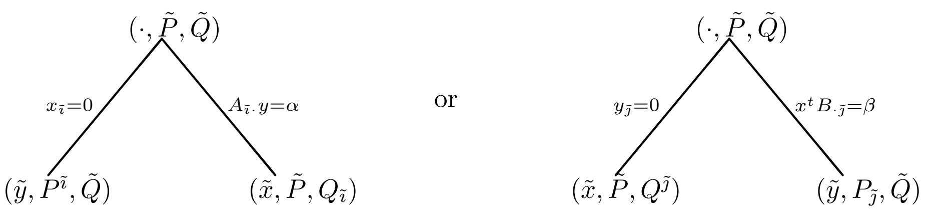

$(\cdot, \tilde{P}, \tilde{Q})$
$x_{\bar{\imath}}=0$
$A_{\bar{\imath}\cdot}y=\alpha$
$(\tilde{y}, P^{\bar{\imath}}, \tilde{Q})$
$(\tilde{x}, \tilde{P}, Q_{\bar{\imath}})$
or
$(\cdot, \tilde{P}, \tilde{Q})$
$y_{\bar{\jmath}}=0$
$x^tB_{\cdot\bar{\jmath}}=\beta$
$(\tilde{x}, \tilde{P}, Q^{\bar{\jmath}})$
$(\tilde{y}, P_{\bar{\jmath}}, \tilde{Q})$

FIG. 3.1. *Dichotomous branching rule.*

Step f is reached only when the depth of the current node in the search tree is greater than $m + n$. The corresponding current solution $(\tilde{x}, \tilde{y})$ is necessarily an extreme equilibrium strategy since all the complementary conditions are satisfied, and feasibility is ensured by Steps c and d. A new branch is created for each strictly positive variable or slack variable of the current solution of $\tilde{P}$ and $\tilde{Q}$. This implies that any equilibria found in the subtree rooted at the current node will differ from $(\tilde{x}, \tilde{y})$.

An important aspect of either of these two branching rules is that every new branch obtained from the current node has one additional inequality constraint converted into an equality, thus reducing the size of one subproblem of each branch. This reduction ensures that every branch of the tree will eventually terminate with an infeasible subproblem, even when degeneracy is encountered. In the degenerate case, it is possible that the same equilibrium is obtained at more than one node of the enumeration tree, but it is recorded only once.

THEOREM 3.1. *Algorithm EEE enumerates in finite time all extreme equilibrium points of a bimatrix game.*

Proof. Let $(\hat{x}, \hat{y})$ be an extreme equilibrium point, and let $\hat{\alpha}$ and $\hat{\beta}$ be the payoffs $\hat{x}^t A \hat{y}$ and $\hat{x}^t B \hat{y}$, respectively. The point $\hat{s} = (\hat{x}, \hat{y}, \hat{\alpha}, \hat{\beta})$ is a vertex of $X \times Y$. We define the following sets of indices indicating binding constraints at that point:

$$I^x = \{i : \hat{x}_i = 0\}, \quad J^y = \{j : \hat{y}_j = 0\}, \quad I^y = \{i : A_{i\cdot}\hat{y} = \alpha\}, \quad J^x = \{j : \hat{x}^t B_{\cdot j} = \beta\}.$$

Clearly, the point $\hat{s}$ belongs to the set

$$S = \left\{ \begin{array}{lrr} (x, \beta, y, \alpha) \in X \times Y : & x_i = 0 \ \forall i \in I^x; & y_j = 0 \ \forall j \in J^y; \\ & A_{i\cdot} y = \alpha \ \forall i \in I^y; & x^t B_{\cdot j} = \beta \ \forall j \in J^x; \end{array} \right\}.$$

Suppose that the set S is not a singleton, i.e., that there is an s' in S such that $s' \neq \hat{s}$. For a given ϵ, consider the point $s_\epsilon = s' + \epsilon(\hat{s} - s')$. If $\epsilon \in [0, 1]$, then convexity of S implies that s_ϵ belongs to S. Recall that $\hat{s}$ is a vertex of $X \times Y$; therefore, for $\epsilon > 1$, s_ϵ does not belong to $X \times Y$ nor to S. It follows that there must be a binding constraint at the point $\hat{s}$ that is free at s'. Since S is defined through all binding constraints at $\hat{s}$, s' cannot belong to S. Therefore, the set S consists of the singleton $\{(\hat{x}, \hat{\beta}, \hat{y}, \hat{\alpha})\}$.

Furthermore, the complementary slackness conditions (3.1) ensure that these sets of indices satisfy $I^x \cup I^y = \{1, 2, \ldots, m\}$ and $J^x \cup J^y = \{1, 2, \ldots, n\}$. If the equilibrium point is nondegenerate, then these pairs of sets are disjoint, i.e., $I^x \cap I^y = \emptyset$ and $J^x \cap J^y = \emptyset$. In the degenerate case, multiple basic representations imply nonempty intersections.

Consider a current node of the enumeration tree where the extreme equilibrium point $(\hat{x}, \hat{y})$ has not yet been detected by the algorithm. If the depth is less than or equal to $|I^x| + |I^y| + |J^x| + |J^y|$ (this sum is equal to $m + n$ in the nondegenerate case), the branching rules select a variable $\tilde{\imath}$ or $\tilde{\jmath}$ in such a way that a new branch of this node will necessarily impose one of the following constraints:

$$\begin{array}{llll} x_{\tilde{\imath}} = 0, \text{ where } \tilde{\imath} \in I^x, & \text{or} & y_{\tilde{\jmath}} = 0, \text{ where } \tilde{\jmath} \in J^y, & \text{or} \\ A_{\tilde{\imath}\cdot} y = \alpha, \text{ where } \tilde{\imath} \in I^y, & \text{or} & x^t B_{\cdot\tilde{\jmath}} = \beta, \text{ where } \tilde{\jmath} \in J^x. & \end{array}$$

If $(\hat{x}, \hat{y})$ is an extreme equilibrium of the problem of the predecessor's node, then it is also one for the current problem as addition of this constraint does not render the current problem infeasible.

There will therefore be a node of depth $|I^x| + |I^y| + |J^x| + |J^y| + 1$ containing the constraints appearing in the set S. At that node, Step e of the algorithm discovers the extreme equilibrium $(\hat{x}, \hat{y})$.

The number of constraints and variables in the sets X and Y are finite. In the enumeration tree, each node contains a problem that has one more variables or slack variables fixed at 0 than its predecessor, and thus, in view of the constraints appearing in (P1), every branch of the tree is bound to end up with an infeasible problem. Therefore, the number of nodes is finite.

Moreover, processing each node is done in finite time as it basically reduces to the resolution of two parameterized linear programs bounded in size by P and Q. It follows that the algorithm is finite. □

We now illustrate the algorithm EEE on a small example and then on a slightly larger but degenerate example.

3.3. A small example. Consider Nash's [18] 2 × 2 example

$$A = \begin{bmatrix} 1 & -10 \\ 10 & -1 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 10 \\ -10 & -1 \end{bmatrix}.$$

Even though this example contains dominated strategies (see e.g., Luce and Raiffa [10]) and could be solved directly by inspection, it is analyzed here to illustrate the steps of the algorithm.

At the initialization step, the problems

$$\begin{aligned} P(y) = \max_{x\geq 0,\beta}\ & (y_1 - 10y_2)x_1 + (10y_1 - y_2)x_2 - \beta \\ \text{s.t.}\quad & x_1 + x_2 = 1, \\ & x_1 - 10x_2 \leq \beta, \\ & 10x_1 - x_2 \leq \beta, \\ Q(x) = \max_{y\geq 0,\alpha}\ & (x_1 - 10x_2)y_1 + (10x_1 - x_2)y_2 - \alpha \\ \text{s.t.}\quad & y_1 + y_2 = 1, \\ & y_1 - 10y_2 \leq \alpha, \\ & 10y_1 - y_2 \leq \alpha, \end{aligned}$$

together with $x^t = (\frac{1}{2}, \frac{1}{2})$, constitute the first node placed into the stack T. This node is selected in Step b and removed from T. Step c solves $Q(\frac{1}{2}, \frac{1}{2})$, obtaining $\tilde{y}_1 = 0, \tilde{y}_2 = 1$, and then solves $P(0,1)$, getting $\tilde{x}_1 = 0, \tilde{x}_2 = 1$. At the end of this step, $\tilde{x}^t = \tilde{y}^t = (0,1)$, $\tilde{\alpha} = \tilde{\beta} = -1$, and the slack variables associated with the constraints $A\tilde{y} \leq \mathbf{1}\tilde{\alpha}$ and $\tilde{x}^t B \leq \tilde{\beta}\mathbf{1}^t$ are both $(9,0)$.

The branching step selects the index $\tilde{\imath} = 1$. The stack T now contains two nodes, one in which the constraint $x_1 = 0$ is added to $\tilde{P}$ (with feasible starting point $\tilde{y}^t = (0,1)$ for $\tilde{Q}$), and the other in which the constraint $A_{1\cdot}y = \alpha$ is added to $\tilde{Q}$ (with feasible starting point $\tilde{x}^t = (0,1)$ for $\tilde{P}$).

Figure 3.2 displays the complete search tree generated by the algorithm. Selection of the nodes in Step b is done following a depth-first strategy. The circled numbers indicate the order in which the nodes are selected.

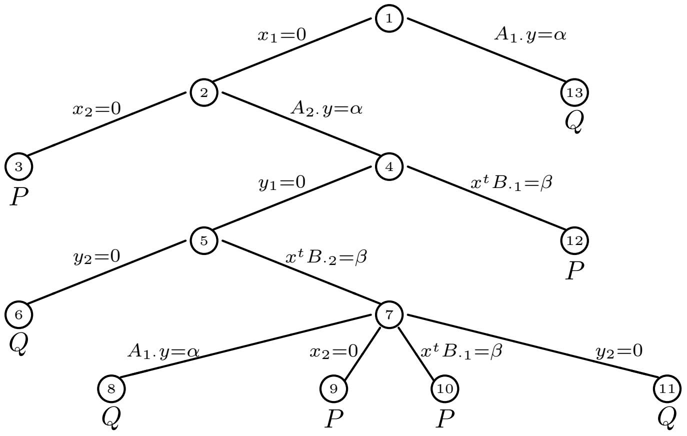


FIG. 3.2. *Search tree.*

We discuss a few important nodes of the search tree. At the third node, subproblem $\tilde{P}$ is found to be infeasible as the constraint $x_1 + x_2 = 1$ is violated by the added constraints $x_1 = x_2 = 0$. Step d detects this infeasibility and backtracks to Step b. The stack T now only contains the problems corresponding to the nodes 4 and 13.

At the seventh node, an equilibrium point is found since all complementary conditions are satisfied, and the current subproblems $\tilde{P}$ and $\tilde{Q}$ are both feasible. Although this point $\tilde{x}^t = \tilde{y}^t = (0,1)$ was obtained at the first node of the search tree, it is recorded only as an equilibrium point at this deeper node. This reduces the time spent in checking if a point is an equilibrium or not. Therefore, the time required to find a first equilibrium could be reduced, but this is not our goal. In Step e, a total of four new nodes are added to the stack T. One corresponds to the strictly positive slack variable of the constraint $A_{1\cdot}\tilde{y} = y_1 - 10\tilde{y}_2 \leq \tilde{\alpha}$, a second to $\tilde{x}_2 \geq 0$, another to the slack variable of the constraint $\tilde{x}^tB_{\cdot 1} = \tilde{x}_1 - 10\tilde{x}_2 \leq \beta$, and a last one to $\tilde{y}_2 \geq 0$. Each of these nodes yields infeasible problems which are detected in either Step c or d. The stack T now contains the problems corresponding to the nodes 12 and 13; all other nodes are completely processed.

At the thirteenth node, subproblem $\tilde{Q}$ is found to be infeasible. Indeed the addition of $y_1 - 10y_2 = \alpha$ to nine times $y_1 + y_2 = 1$ violates the constraint $10y_1 - y_2 \leq \alpha$. Step c detects this infeasibility and backtracks to Step b. The stack T is now empty, thus ending execution of the algorithm.

The use of the complementary slackness conditions is the key to the efficiency of the algorithm. For example, consider node 13. As mentioned above, the complementary condition $A_{1\cdot}y = \alpha$ obtained from the branching rule allows immediate pruning of the branch. If that rule was replaced by $x_1 = 0$ versus $x_1 > 0$, then pruning of the branch is not immediately possible since there is a vertex $(x^t, \beta) = (1, 0, 10)$ of the set X that satisfies the constraint $x_1 > 0$. That point is therefore a candidate for an extreme equilibrium strategy. Under this weakened branching rule, node 13 should be further investigated. For larger problems, this significantly reduces the number of nodes visited by the algorithm.

3.4. A degenerate example. Winkels's [24] 6×2 example is next discussed to illustrate how algorithm EEE behaves when degeneracy occurs:

$$A^t = \begin{bmatrix} 1 & 1 & 3 & 3 & \frac{5}{2} & \frac{5}{2} \\ 3 & 3 & 1 & 1 & \frac{5}{2} & \frac{5}{2} \end{bmatrix}, \quad B^t = \begin{bmatrix} 1 & 0 & -2 & 4 & -1 & 6 \\ 2 & -1 & 2 & -1 & 6 & -1 \end{bmatrix}.$$

The first node of depth greater than $m + n = 8$ where both subproblems are feasible is the nineteenth. To reach that node, the first branching rule converted eight complementary slackness conditions into

$$x_1 = x_2 = x_3 = x_4 = 0, \; A_{5\cdot}y = A_{6\cdot}y = \alpha, \quad \text{and} \quad x^tB_{\cdot 1} = x^tB_{\cdot 2} = \beta.$$

The extreme equilibrium point obtained by solving the current subproblems is

$$\begin{aligned} \tilde{x}^t &= (0,0,0,0,\tfrac{1}{2},\tfrac{1}{2}), & \tilde{y}^t &= (\tfrac{3}{4},\tfrac{1}{4}), & \tilde{\alpha} &= \tfrac{5}{2}, \\ (A\tilde{y} - \mathbf{1}\tilde{\alpha})^t &= (1,1,0,0,0,0), & (\tilde{x}^tB - \tilde{\beta}\mathbf{1}^t) &= (0,0), & \tilde{\beta} &= \tfrac{5}{2}. \end{aligned}$$

Both alternatives of the complementary slackness condition $\tilde{x}_3 = 0$ or $A_{3\cdot}\tilde{y} = \tilde{\alpha}$ are satisfied. (The same is true for $\tilde{x}_4 = 0$ or $A_{4\cdot}\tilde{y} = \tilde{\alpha}$.) The solution is therefore degenerate. Figure 3.3 displays the subtree rooted at node 19, obtained with the second branching rule.

A second extreme equilibrium point is obtained at node 20:

$$\begin{aligned} \tilde{x}^t &= (0,0,0,0,\tfrac{1}{2},\tfrac{1}{2}), & \tilde{y}^t &= (\tfrac{1}{4},\tfrac{3}{4}), & \tilde{\alpha} &= \tfrac{5}{2}, \\ (A\tilde{y} - \mathbf{1}\tilde{\alpha})^t &= (0,0,1,1,0,0), & (\tilde{x}^tB - \tilde{\beta}\mathbf{1}^t) &= (0,0), & \tilde{\beta} &= \tfrac{5}{2}. \end{aligned}$$

The same extreme equilibrium is found at node 27 but is not recorded again. Nodes 21 to 26 and 28 to 37 are discarded through the infeasibility criterion of Steps c or d. Processing of these nodes is rapid since infeasibility is detected in the first phase of the Simplex algorithm. All twelve equilibria were obtained with 417 nodes.

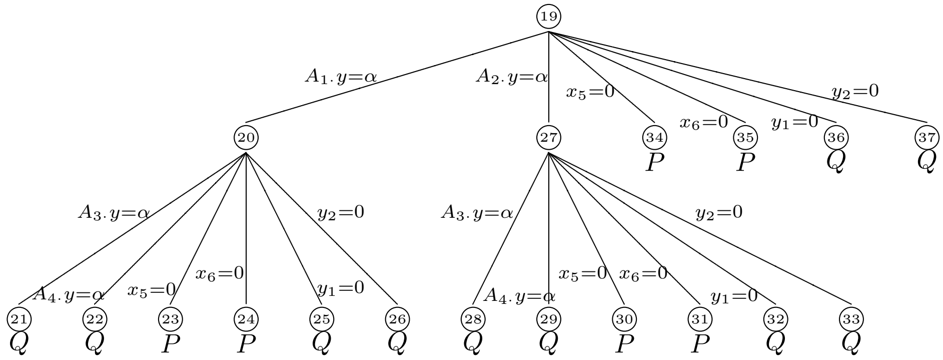


FIG. 3.3. *Search tree.*

4. Numerical results. We have shown that the algorithm EEE enumerates all extreme equilibria of bimatrix games. However, the size of the instances that can be solved and the precision to which computations are made depend on the implementation and on the computers on which the experiments are made. The algorithm is coded in C, and the CPLEX library is used to solve the parameterized linear programs $\tilde{P}$ and $\tilde{Q}$. Computational experiments are made on a SPARC station SS20/514MP under Solaris 2.4-27 using double precision floating point arithmetic.

The entries in Table 4.1 are mean values (μ) and standard deviations (σ) for ten randomly generated problems where the coefficients of the matrices A and B are drawn from a uniform distribution over the real interval $[0, 10]$. This process is unlikely to produce degenerate bimatrix games. Note that algorithm EEE may be inefficient for degenerate games (as illustrated by the fact that it requires 417 nodes for Winkels's small example discussed above). Indeed, the same equilibrium will be found many times at a level greater than $n + m$. The columns *time* and *nodes* are the average computing times in seconds and average total number of nodes explored in the search tree. The *equilibrium* columns display the average *total* number of extreme equilibria and the average number of the node at which the *first* extreme equilibrium point is detected.

The total number of extreme equilibrium points does not increase very rapidly. It remains almost stable for large problems with m fixed at 5. This is due in part to the large number of dominated strategies inherent to such randomly generated games. That number goes down when both dimensions are close. When $m = n$, there appears to be an exponential growth in the number of extreme equilibria, but it remains, however, moderate. The average node at which the first extreme equilibria is detected is fairly large as the method only checks for equilibria at nodes of depth greater than $m + n$. The logarithm of the execution times is plotted on the graphs of Figure 4.1.

It appears that the time required when $m = n$ grows exponentially with n. For large values of n, some of the problems were difficult to solve, yielding large standard deviations. One of the ten problems generated for $n = 21$ was very difficult and required 2140 seconds and 272034 nodes. When m is fixed at $5, 10$, or 15, the execution time seems to grow asymptotically towards an exponential since the logarithm of the computational times approaches linearity for large values of n. It also appears that the time spent by the algorithm at each node is roughly proportional to the product of m and n.

Table 4.1
Results of algorithm EEE on randomly generated problems.

$m = 5$						$m = 10$					
		Time	Nodes	Equilibrium				Time	Nodes	Equilibrium	
n		(sec)		total	first	n		(sec)		total	first
100	μ	81.3	11189	19.6	484.8	30	μ	59.0	11138	45.2	205.4
	σ	33.0	4808	10.6	307.8		σ	26.7	4781	18.7	131.6
200	μ	400.1	34864	27.2	1436.0	50	μ	143.2	23599	63.2	219.0
	σ	186.0	16202	12.9	1187.9		σ	109.0	15561	32.1	288.4
300	μ	1190.0	71865	34.2	2224.0	70	μ	696.0	86059	105.4	549.2
	σ	429.8	26301	14.1	1909.5		σ	275.2	34126	43.5	515.6
400	μ	2303.6	105576	38.6	1659.2	90	μ	1641.5	166143	155.8	944.0
	σ	861.5	40074	15.0	528.9		σ	659.7	66673	66.3	687.9
500	μ	3992.6	147541	41.6	2948.6	110	μ	3108.8	278523	182.0	739.2
	σ	1421.3	53457	17.7	2466.4		σ	1290.0	115584	83.2	869.6
600	μ	6523.2	199437	47.6	4989.8	130	μ	5822.0	452405	222.6	3483.2
	σ	2631.5	79950	15.2	3733.7		σ	1717.3	130085	75.3	4606.8
700	μ	10060.4	263500	47.8	4246.6	150	μ	9583.3	677202	266.0	1122.6
	σ	2656.8	71379	9.3	2955.1		σ	2817.4	192742	69.5	1439.8

$m = 15$						$m = n$					
		Time	Nodes	Equilibrium				Time	Nodes	Equilibrium	
n		(sec)		total	first	n		(sec)		total	first
10	μ	10.7	2768	20.2	89.0	5	μ	0.2	96	3.2	16.6
	σ	8.1	2015	12.9	41.9		σ	0.0	20	1.1	5.1
20	μ	73.7	12970	58.6	173.0	9	μ	1.6	541	8.4	38.2
	σ	41.9	6889	28.5	125.2		σ	1.2	403	6.3	20.4
30	μ	330.1	50118	126.0	198.8	13	μ	9.5	2323	18.0	80.4
	σ	195.1	28996	77.9	141.5		σ	6.2	1471	11.0	55.2
40	μ	1150.3	143245	222.8	508.4	17	μ	83.4	15421	61.4	238.8
	σ	683.7	83440	122.6	689.1		σ	78.6	12648	33.9	190.6
50	μ	3252.4	345908	355.8	666.4	21	μ	705.7	97725	204.4	475.8
	σ	1745.6	174208	127.8	591.7		σ	671.9	85925	144.1	470.4
60	μ	6898.2	674661	491.6	1194.2	25	μ	2427.6	280202	441.3	452.4
	σ	2961.0	274986	171.1	1254.7		σ	1190.5	133640	226.8	586.6
70	μ	13671.3	1196031	666.5	1028.0	29	μ	16216.0	1444924	1254.5	1597.6
	σ	5803.0	504102	227.6	1165.2		σ	11214.7	921033	679.6	3731.4

Figure 4.2 suggests that the total execution time of algorithm EEE is polynomial in the total number of extreme equilibria. The figure displays the logarithm of the execution times on the y-axis and the logarithm of the total number of extreme equilibria on the x-axis for the randomly generated games of Table 4.1 where $m = n$.

When m and n differ significantly, analysis of dominated strategies prior to the execution of algorithm EEE can improve computational time for problems generated as above. We used McKelvey and McLennan's [13] package GAMBIT v0.94 to solve various instances. Computational results are summarized in Table 4.2, where a single instance corresponds to each line. Problems are randomly generated as above and solved by EEE, then dominated strategies are identified and removed, and, finally, EEE is executed again on these smaller games.

The first five columns display characteristics of the instances considered and of the execution of EEE, as in the other tables. The columns ElimDom are derived from the strongly dominated strategy removal routine of the GAMBIT package. The columns m' and n' indicate the size of the resulting reduced bimatrix game. The last columns detail execution of the algorithm EEE on these smaller instances, where the bracketed entries are the sum of the execution times of EEE and ElimDom.

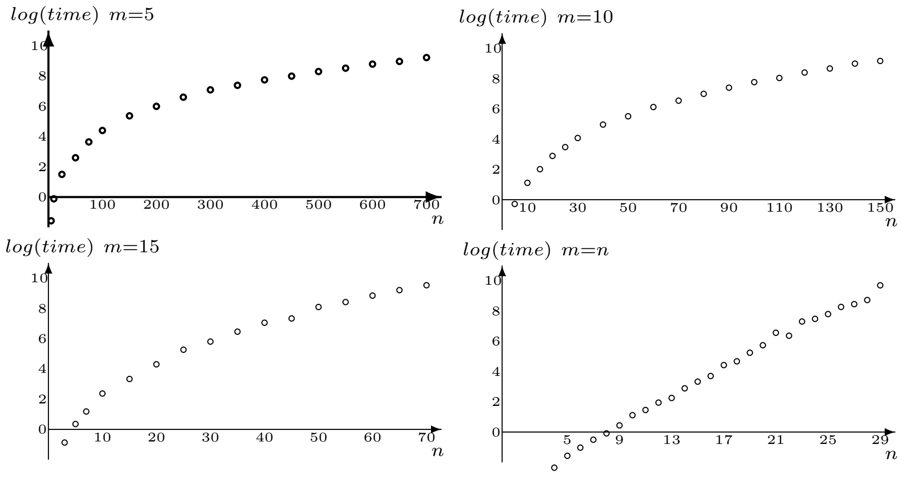


FIG. 4.1. *Logarithm of the execution times of algorithm EEE.*

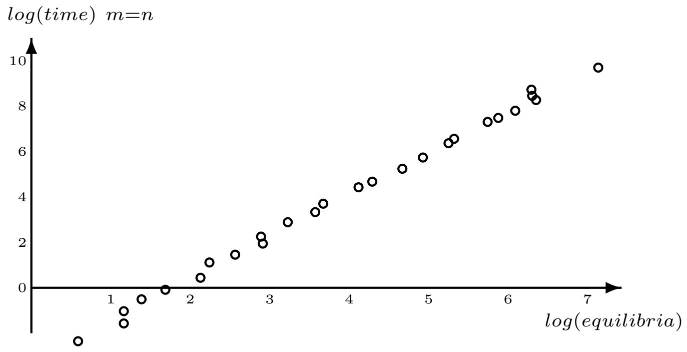


FIG. 4.2. *Logarithm of the execution times of algorithm EEE versus logarithm of the number of equilibria.*

Enumeration of all extreme equilibria using GAMBIT's *EnumMixed* procedure is possible in reasonable time (less than three hours) only for the smallest instances considered. For example, the time required for that procedure (excluding that of ElimDom), for the problem where the dimensions are reduced from 100×5 to 17×5, is 70.6 seconds, and the time required for the one reduced from 200×5 to 29×5 is 1803.5 seconds. However, for the next instance, the one where the size is reduced from 300×5 to 47×5, the time required is more than 17.5 hours. This computational time is more than 1700 times longer than the 36.1 seconds required by algorithm EEE on the same reduced game.

A significant gain in computing time occurs when there is an important difference between the dimensions and when the largest dimension is not too high. For such randomly generated games that contain many dominated strategies, it appears that their identification and removal prior to the execution of the algorithm EEE reduces the total computational time.

The alternative straightforward approach consisting in enumerating in a first step all vertices of the polyhedra X and Y and then checking for equilibrium may be

Table 4.2
Games with dominated strategies.

		Equil	Algorithm EEE		ElimDom			Algorithm EEE		
m	n	total	time(sec)	nodes	time(sec)	m'	n'	time[tot](sec)		nodes
5	100	11	42.7	6824	54.7	5	17	2.5	[57.2]	1117
5	200	13	281.4	21584	335.1	5	29	7.6	[343.0]	2487
5	300	35	1182.2	71286	1132.2	5	47	36.1	[1168.3]	9461
5	400	45	2652.5	127514	2498.9	5	53	53.9	[2552.8]	13101
5	500	55	4291.7	169048	4612.7	5	57	71.2	[4683.9]	17861
5	600	45	5857.9	190022	7923.3	5	55	62.4	[7985.7]	15247
5	700	47	9823.4	257690	12601.4	5	61	81.8	[12683.2]	18003
10	30	37	53.1	12405	12.8	10	23	34.6	[47.4]	9258
10	50	95	331.7	53945	49.6	10	34	140.2	[189.8]	28503
10	70	115	678.4	100443	117.3	10	48	298.9	[416.2]	57073
10	90	239	2245.9	256877	195.7	10	63	857.8	[1053.5]	129368
10	110	313	4400.8	450717	316.0	10	71	1907.3	[2223.3]	263828
10	130	307	6595.2	569323	496.8	10	76	2112.5	[2609.3]	262707
10	150	209	6131.0	477901	733.6	10	84	1637.2	[2370.8]	203327
15	40	111	814.1	117412	80.5	15	39	875.6	[956.1]	110597
15	50	255	3001.9	360150	124.5	15	45	2109.9	[2234.4]	264109
15	60	399	8853.0	946564	193.5	15	53	7047.5	[7241.0]	767941
15	70	553	18828.4	1775200	306.2	15	62	15370.0	[15676.2]	1447174

efficient for small problems. However, the number of vertices grows much more rapidly than the number of nodes visited by algorithm EEE. Table 4.3 displays the mean and the standard deviation of the number of nodes required by the search tree as well as the number of vertices of the polyhedron Y for ten problems randomly generated as above. Computational times, including those of the GAMBIT package, are also reported.

Table 4.3
Comparison of algorithms in terms of times, nodes and vertices.

	Algorithm EEE		Fukuda		GAMBIT
	Time	Nodes	Time	Vertices	Time
$m = n$	(sec)		(sec)	of Y	(sec)
9 μ	1.6	541	.9	499	57.4
σ	1.2	403	.6	204	82.1
10 μ	3.0	856	3.9	1084	590.9
σ	1.6	444	1.0	175	464.4
11 μ	4.3	1248	15.2	2411	4848.3
σ	1.6	442	5.3	811	3401.9
12 μ	7.0	1899	56.2	4803	
σ	3.0	763	22.6	1517	
13 μ	9.5	2323	260.5	8737	
σ	6.2	1471	154.6	4852	
14 μ	18.0	4197	1620.9	22731	
σ	7.9	1626	895.4	8852	
15 μ	28.0	6313	9501.0	50874	
σ	15.2	3369	6896.2	24298	

Fukuda's [5] state-of-the-art implementation in C of the double description method of [16] is used for vertex enumeration. The number of vertices of Y grows much more rapidly than the number of nodes visited by algorithm EEE. Computational time required for enumerating all vertices follows their number. Even if another algorithm or a different implementation were used, the time would quickly exceed that required by algorithm EEE. Moreover, vertex enumeration of Y is only the first step of the straightforward approach. Thus it does not appear that this approach is competitive.

In a second step (as long as the first one), all vertices of the polyhedron X should be enumerated, and a final step would verify complementary conditions for each pair of vertices. This approach does not seem reasonable even for midsized problems, e.g., the average number of vertices for the 15×15 problems is over 5×10^4 for both polyhedra, and thus approximately 2.5×10^9 equilibrium conditions should be checked. (These conditions require the knowledge of slack variables.) Moreover, additional difficulties arise in vertex enumeration procedures when degeneracy is encountered. Table 4.3 shows that algorithm EEE is more than 1000 times faster than GAMBIT for a problem of size 11×11, and that factor grows with problem size. For those problems, GAMBIT's procedure for eliminating dominated strategies always took less than 2.1 seconds for each instance considered; thus the most computational time is needed for the extreme equilibria enumeration procedure.

In conclusion, the proposed algorithm allows enumeration of all extreme equilibrium points of substantially larger bimatrix games than done by previous ones. The key to the efficiency of the algorithm is the systematic use of complementary slackness conditions in the branching rule. This allows joint use of information concerning the polyhedra X and Y, unlike vertex enumeration based procedures that treat them as two distinct entities.

REFERENCES

[1] V. Aggarwal, *On the generation of all equilibrium points for bimatrix games through the Lemke-Howson algorithm*, Math. Program., 4 (1973), pp. 233–234.

[2] M. L. Balinski, *An algorithm for finding all vertices of convex polyhedral sets*, J. Soc. Indust. Appl. Math., 9 (1961), pp. 72–88.

[3] J. Dickhaut and T. Kaplan, *A program for finding Nash equilibria*, Mathematica J., 1 (1991), pp. 87–93.

[4] M. E. Dyer, *The complexity of vertex enumeration methods*, Math. Oper. Res., 8 (1983), pp. 381–402.

[5] K. Fukuda, *CDD + Reference Manual*, Institute for Operations Research, Swiss Federal Institute of Technology, Zurich, Switzerland, 1995.

[6] M. J. M. Jansen, *Maximal Nash subsets for bimatrix games*, Naval Res. Log., 18 (1981), pp. 147–152.

[7] E. Kohlberg, *Refinements of the Nash equilibrium: The main ideas*, in Game Theory and Applications, T. Ichiishi, A. Neyman, and Y. Tauman, eds., San Diego Academic Press, San Diego, CA, 1990, pp. 3–45.

[8] H. W. Kuhn, *An algorithm for equilibrium points in bimatrix games*, Proc. of the Nat. Acad. Sci. USA, 47 (1961), pp. 1657–1662.

[9] C. E. Lemke and T. T. Howson, *Equilibrium points of bimatrix games*, J. Soc. Indust. Appl. Math., 12 (1964), pp. 413–423.

[10] R. D. Luce and H. Raiffa, *Games and Decisions: Introduction and Critical Survey*, John Wiley and Sons, New York, 1957.

[11] O. L. Mangasarian, *Equilibrium points of bimatrix games*, J. Soc. Indust. Appl. Math., 12 (1964), pp. 778–780.

[12] O. L. Mangasarian and H. Stone, *Two-person nonzero-sum games and quadratic programming*, J. Math. Anal. Appl., 9 (1964), pp. 348–355.

[13] R. D. McKelvey and A. McLennan, *Computation of equilibria in finite games*, in Handbook of Computational Economics Vol. I, H. M. Amman, D. A. Kendrick, and J. Rust, eds., Elsevier, Amsterdam, 1996, pp. 87–142.

[14] C. B. Millham, *On Nash subsets of bimatrix games*, Naval Res. Logist., 74 (1974), pp. 307–317.

[15] H. Mills, *Equilibrium points in finite games*, J. Soc. Indust. Appl. Math., 8 (1960), pp. 397–402.

[16] T. S. Motzkin, H. Raiffa, G. L. Thompson, and R. M. Thrall, *The double description method*, in Contributions to the Theory of Games II, H. W. Kuhn and A. W. Tucker, eds., Princeton University Press, Princeton, 1953.

[17] B. M. Mukhamediev, *The solution of bilinear programming problems and finding the equilibrium situations in bimatrix games*, Zh. Vychisl. Mat. Mat. Fiz., 18 (1978), pp. 351–359 (in Russian); Comput. Math. Math. Phys. 18 (1978), pp. 60–66 (in English).

[18] J. F. Nash, *Equilibrium points in n-person games*, Proc. Natl. Acad. Sci. USA, 36 (1950), pp. 48–49.

[19] E. Van Damme, *Refinements of Nash equilibrium*, in Advances in Economic Theory, J. J. Laffont, ed., Cambridge University Press, Cambridge, UK 1992, pp. 32–75.

[20] E. Van Damme, *Stability and Perfection of Nash Equilibria*, 2nd ed., Springer-Verlag, Berlin, New York, 1996.

[21] B. Von Stengel, *Computing equilibria for two-person games*, in Handbook of Game Theory, Vol. 3, R. J. Aumann and S. Hart, eds., Elsevier, Amsterdam, 2001, to appear.

[22] B. Von Stengel, *Improved Equilibrium Enumeration for Bimatrix Games*, Extended abstract, Institute for Theoretical Computer Science, ETH, Zürich, Switzerland, 1998.

[23] N. N. Vorob'ev, *Equilibrium points in bimatrix games*, Teor. Veroyatnost. i Primenen., 3 (1958), pp. 318–331 (in Russian); Theory Probab. Appl., 3 (1958), pp. 297–309 (in English).

[24] H. M. Winkels, *An algorithm to determine all equilibrium points of a bimatrix game*, in Game Theory and Related Topics, O. Moeschlin and D. Pallaschke, eds., North-Holland, Amsterdam, 1979, pp. 137–148.

SIAM J. SCI. COMPUT.
Vol. 23, No. 1, pp. 339–361

MAXIMUM PRINCIPLE PRESERVING SCHEMES FOR INTERFACE PROBLEMS WITH DISCONTINUOUS COEFFICIENTS[*]

ZHILIN LI[†] AND KAZUFUMI ITO[†]

Abstract. New finite difference methods using Cartesian grids are developed for elliptic interface problems with variable discontinuous coefficients, singular sources, and nonsmooth or even discontinuous solutions. The new finite difference schemes are constructed to satisfy the sign property of the discrete maximum principle using quadratic optimization techniques. The methods are shown to converge under certain conditions using comparison functions. The coefficient matrix of the resulting linear system of equations is an M-matrix and is coupled with a multigrid solver. Numerical examples are also provided to show the efficiency of the proposed methods.

Key words. elliptic interface problems, finite difference methods, discontinuous coefficients, singular source term, discrete maximum principle, quadratic optimization, multigrid methods

AMS subject classifications. 65N06, 65N50

PII. S1064827500370160

1. Introduction.

Many important practical problems lead to partial differential equations (PDEs) whose solutions or derivatives have discontinuities across some interfaces within the solution domains. In this paper, we propose a class of numerical methods that preserve the discrete maximum principle for the interface problems defined below:

$$(\beta u_x)_x + (\beta u_y)_y - \kappa(x,y)\,u = f(x,y), \qquad (x,y) \in \Omega = \Omega^+ \cap \Omega^-, \tag{1.1}$$

with a boundary condition on $\partial\Omega$, where $\beta \geq \beta_{min} > 0$, $\kappa \geq 0$, and f are *piecewise* continuous but may have a jump discontinuity across some *smooth* curve Γ in the domain Ω; see Figure 1(a) for an illustration. The source term f can also contain singular sources as reflected in the following jump conditions:

$$[u]\Big|_{\mathbf{X}\in\Gamma} = w(s), \qquad [\beta u_n]\Big|_{\mathbf{X}\in\Gamma} = v(s), \tag{1.2}$$

where $\mathbf{X} = (X, Y)$ is a point on the interface Γ, s is the arc-length parameterization of the interface Γ, and the jump is defined as the difference of the limiting values of two different sides of the interface, for example,

$$[u]\Big|_{\mathbf{X}\in\Gamma} = \lim_{\mathbf{x}\to\mathbf{X},\mathbf{X}\in\Omega^+} u(\mathbf{x}) \;-\; \lim_{\mathbf{x}\to\mathbf{X},\mathbf{X}\in\Omega^-} u(\mathbf{x}),$$

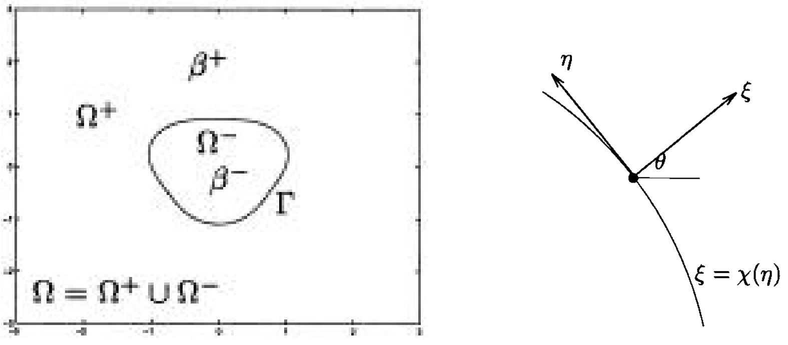
where $\mathbf{x} = (x, y) \in \Omega$. The interface Γ can be arbitrary, but it is assumed to be a smooth closed curve[1] lying in Ω. In the case that κ is continuous and $w(s) \equiv 0$, the

[*] Received by the editors April 4, 2000; accepted for publication (in revised form) March 7, 2001; published electronically June 26, 2001. This work was supported by ARO grant 39676-MA, NSF grant DMS-96-26703, and the North Carolina State FR&PD Fund.

http://www.siam.org/journals/sisc/23-1/37016.html

[†] Center for Research in Scientific Computation, Department of Mathematics, North Carolina State University, Raleigh, NC 27695 (zhilin@math.ncsu.edu, kito@math.ncsu.edu).

[1] We impose some conditions on β, κ, and Γ for simplicity, especially for theoretical discussions. Nevertheless, most of these conditions can be relaxed when we apply the algorithms discussed in this paper.



(a) (b)

FIG. 1.1. (a) *A diagram of a rectangular domain* $\Omega = \Omega^+ \cup \Omega^-$ *with an immersed interface* Γ. *The coefficient* $\beta(\mathbf{x})$ *may have a jump across the interface.* (b) *A diagram of the local coordinates in the normal and tangential directions, where* θ *is the angle between the* x*-axis and the normal direction.*

interface problem can be written as the boundary value problem below:

$$\begin{aligned} &\nabla \cdot (\beta \nabla u(\mathbf{x})) - \kappa\, u(\mathbf{x}) = f(\mathbf{x}) + \int_\Gamma v(s)\delta_2(\mathbf{x} - \mathbf{X}(s))ds, \\ &\text{with a given boundary condition on} \quad \partial\Omega, \end{aligned}$$

where δ_2 is the two-dimensional Dirac delta function; the second term at the right-hand side is a distribution which satisfies

$$\iint_\Omega \int_\Gamma v(s)\delta_2(\mathbf{x} - \mathbf{X}(s))\Psi(\mathbf{x})ds\, d\mathbf{x} = \int_\Gamma v(s)\Psi(\mathbf{X}(s))ds \tag{1.3}$$

for any arbitrary smooth function $\Psi(x, y)$. The discussion of the existence and the regularity of the solution can be found, for example, in [2, 3]. In general, if β, κ, and f are piecewise smooth in Ω, $w = 0$, and v is continuous along Γ, then the solution to the interface problem exists and is in $H^1(\Omega)$.

There are several numerical methods in the literature designed for the interface problems discussed in this paper. We just mention a few here: the finite element methods using body fitting grids [3]; the fast solvers based on integral equations for piecewise constant coefficients including the fast multipole method; the first order ghost fluid method [11]. In this paper, our new methods are based on the immersed interface method (IIM) [7, 8].

The IIM is a second order finite difference method based on Cartesian grids for interface problems in which the jump conditions across the interface are known. The IIM gives sharp solutions (no smear-out) across the interfaces since the jump conditions are enforced. Generally the IIM uses only local information, specifically, the PDEs, the jump conditions, the interface, and the underlying grid. The IIM has been successfully coupled with evolution schemes such as the particle approach and the level set method for moving interface and free boundary problems.

For an interface problem with a variable coefficient that has discontinuity across the interface, the resulting linear system of equations from the original IIM is not symmetric positive definite. While it is stable for one-dimensional and certain two-dimensional problems [6], the stability of the algorithm depends on the choice of one or more extra grid points in addition to the standard five-point stencil [5]. In other words, it is not always guaranteed that the original IIM will converge and satisfy the maximum principle. Many efficient linear solvers may not work or may work poorly for the linear system of equations derived from the original IIM.

In this paper, we will develop new methods for *arbitrary $\beta(x,y)$ using direct finite difference discretization*. The new methods satisfy the sign property that guarantees the discrete maximum principle. The sign property is enforced through a constrained quadratic optimization problem. The coefficient matrix of the resulting linear system of equations is diagonally dominant and its symmetric part is negative definite. Such a matrix is an M-matrix that guarantees some iterative methods such as the SOR method converge. Two specific schemes using the optimization approach are discussed. One is a first order method that uses the standard five-point stencil. The other one is a second order method that uses a standard nine-point stencil. The convergence of the methods is warranted if the solution to the optimization problem exists and certain bounds are satisfied. The existence of the solution to the optimization problem is proved for the first order method and verified numerically for the second order method.

The idea in this paper actually was proposed and tested by Z. Li (1993) and S. Moskow and F. Santosa (1996–1997). However, due to various reasons, it has never been written up and published.

The paper is organized as follows. In section 2, we lay down some theoretical fundamentals for interface problems that are needed in deriving the new methods. We derive the new methods using optimization techniques in section 3. Convergence analysis for the first order and second order methods is given in section 4 and 5, respectively. The numerical results are presented in section 6.

2. Preliminaries.

2.1. The local coordinates. Given a point (x_i^*, y_j^*) on the interface, it is convenient to use the local coordinates in the normal and the tangential directions:

$$\xi = (x - x_i^*)\cos\theta + (y - y_j^*)\sin\theta, \qquad \eta = -(x - x_i^*)\sin\theta + (y - y_j^*)\cos\theta, \tag{2.1}$$

where θ is the angle between the x-axis and the normal direction, pointing to the direction of a specified side, say, the $+$ side in Figure 1(b). We use the superscripts $-$ or $+$ to denote the limiting values of a function from one side or the other. Under the local coordinates, the limiting differential equation approaching the interface from a particular side, for example, from the $-$ side, can be written

$$\beta^-\left(u_{\xi\xi}^- + u_{\eta\eta}^-\right) + \beta_\xi^- u_\xi^- + \beta_\eta^- u_\eta^- - \kappa^- u^- = f^-. \tag{2.2}$$

In a neighborhood of the point (x_i^*, y_j^*), the interface Γ can be parameterized as

$$\xi = \chi(\eta), \quad \text{with} \quad \chi(0) = 0, \quad \chi'(0) = 0. \tag{2.3}$$

The curvature of the interface at (x_i^*, y_j^*) is $\chi''(0)$.

2.2. The interface relations. Let $u(x,y)$ be the solution to (1.1) and (1.2). Let (x_i^*, y_j^*) be a point on the interface. We have derived the following interface relations in [7, 8] that represent the limiting quantities of the solution of (1.1)–(1.2) and its derivatives from one side of the interface in terms of the other.

$$\begin{aligned}
& u^+ = u^- + w, \qquad u_\xi^+ = \rho\, u_\xi^- + \frac{v}{\beta^+}, \qquad u_\eta^+ = u_\eta^- + w', \\
& u_{\xi\,\xi}^+ = \left(\frac{{\beta_\xi}^-}{\beta^+} - \chi'' \right) u_\xi^- + \left(\chi'' - \frac{{\beta_\xi}^+}{\beta^+} \right) u_\xi^+ + \frac{{\beta_\eta}^-}{\beta^+}\, u_\eta^- - \frac{{\beta_\eta}^+}{\beta^+}\, u_\eta^+ \\
& \qquad\qquad + (\rho - 1)\, u_{\eta\,\eta}^- + \rho u_{\xi\,\xi}^- - w'' + \frac{[f]}{\beta^+} + \frac{[\kappa]\, u^- + \kappa^+\, [u]}{\beta^+}, \\
& u_{\eta\,\eta}^+ = u_{\eta\,\eta}^- + (u_\xi^- - u_\xi^+)\, \chi'' + w'', \\
& u_{\xi\,\eta}^+ = \frac{{\beta_\eta}^-}{\beta^+}\, u_\xi^- - \frac{{\beta_\eta}^+}{\beta^+} u_\xi^+ + \left(u_\eta^+ - \rho u_\eta^- \right) \chi'' + \rho\, u_{\xi\,\eta}^- + \frac{v'}{\beta^+},
\end{aligned} \tag{2.4}$$

where $\rho = \beta^-/\beta^+$. These interface relations are used in deriving the new finite difference methods in the next section.

3. The algorithm description. We assume the domain Ω is a rectangle, say, $[a, b] \times [c, d]$. We take a uniform grid with

$$x_i = a + ih, \quad y_j = a + jh, \qquad i = 0, 1, \ldots, m, \quad j = 0, 1, \ldots, n,$$

where $h = (b - a)/m$ and $h = (d - c)/n$. The discussions are easily generalized to the cases in which the spatial step sizes are different in each direction. Our goal is to develop a finite difference equation of the form

$$\sum_{k=1}^{n_s} \gamma_k\, U_{i+i_k, j+j_k} - \kappa_{ij}\, U_{ij} = f_{ij} + C_{ij} \tag{3.1}$$

at any grid point (x_i, y_j), where n_s is the number of grid points in the finite difference stencil, and U_{ij} is the solution to the linear system of equations (3.1) and an approximation to the solution $u(x, y)$ of (1.1) and (1.2) at (x_i, y_j). The sum over k involves a finite number of points neighboring (x_i, y_j). So each i_k, j_k will take values in the set $\{0, \pm 1, \pm 2, \ldots\}$. The coefficients γ_k and the indices i_k, j_k, and n_s will depend on (i, j), so these should really be labeled γ_{ijk}, etc., but for simplicity of notation we will concentrate on a single grid point (x_i, y_j) and drop these indices.

The local truncation error at a grid point (x_i, y_j) is defined as

$$T_{ij} = \sum_{k=1}^{n_s} \gamma_k\, u\left(x_{i+i_k}, y_{j+j_k}\right) - \kappa_{ij}\, u(x_i, y_j) - f(x_i, y_j) - C_{ij}. \tag{3.2}$$

We say (x_i, y_j) is a *regular grid point* if all the grid points in the standard five-point stencil are in the same side of the interface. At regular grid points, we use the standard five-point ($n_s = 5$) central finite difference scheme with $C_{ij} = 0$. The local truncation error at regular grid points is $O(h^2)$.

We need to determine formulas of the form (3.1) for *irregular grid points* around which the standard five-point stencil contains grid points from both sides of the interface. First we choose a point (x_i^*, y_j^*) on the interface Γ near the grid point (x_i, y_j). Usually, we take (x_i^*, y_j^*) either as the orthogonal projection of (x_i, y_j) on the interface or the intersection of the interface and one of the axes. We then expand each $u(x_{i+i_k}, y_{j+j_k})$ about (x_i^*, y_j^*) under the local coordinates,

$$u(x_{i+i_k}, y_{j+j_k}) = u(\xi_k, \eta_k) = u^\pm + \xi_k u_\xi^\pm + \eta_k u_\eta^\pm + \frac{1}{2}\xi_k^2 u_{\xi\xi}^\pm + \xi_k \eta_k u_{\xi\eta}^\pm + \frac{1}{2}\eta_k^2 u_{\eta\eta}^\pm + O(h^3),$$

where the + or − sign is chosen depending on whether (ξ_k, η_k) lies on the + or − side of Γ. If we do this expansion at each point involved in the finite difference equation (3.1), then the local truncation error T_{ij} can be expressed as a linear combination of the values $u^{\pm}, u_\xi^{\pm}, u_\eta^{\pm}, u_{\xi\xi}^{\pm}, u_{\xi\eta}^{\pm}, u_{\eta\eta}^{\pm}$ as the following:

$$\begin{aligned} T_{ij} &= a_1\, u^- + a_2\, u^+ + a_3\, u_\xi^- + a_4\, u_\xi^+ + a_5\, u_\eta^- + a_6 u_\eta^+ + a_7\, u_{\xi\,\xi}^- + a_8\, u_{\xi\,\xi}^+ \\ &\quad + a_9\, u_{\eta\,\eta}^- + a_{10}\, u_{\eta\,\eta}^+ + a_{11}\, u_{\xi\,\eta}^- + a_{12}\, u_{\xi\,\eta}^+ - \kappa^- u^- - f^- - C_{ij} + O(h). \end{aligned}$$

The quantities $f^{\pm}$, $\kappa^{\pm}$, $\beta^{\pm}$ are the limiting values of the functions at (x_i^*, y_j^*) from the + side or − side of the interface. The coefficients a_j depend only on the position of the stencil relative to the interface. They are independent of the functions u, κ, and f. If we define the index sets K^+ and K^- by

$$K^{\pm} = \{k : \ (\xi_k, \eta_k) \text{ is on the } \pm \text{ side of } \Gamma\},$$

then the a_j are given by

$$\begin{aligned} &a_1 = \sum_{k\in K^-} \gamma_k, && a_2 = \sum_{k\in K^+} \gamma_k, && a_3 = \sum_{k\in K^-} \xi_k\gamma_k, \\ &a_4 = \sum_{k\in K^+} \xi_k\gamma_k, && a_5 = \sum_{k\in K^-} \eta_k\gamma_k, && a_6 = \sum_{k\in K^+} \eta_k\gamma_k, \\ &a_7 = \frac{1}{2}\sum_{k\in K^-} \xi_k^2\gamma_k, && a_8 = \frac{1}{2}\sum_{k\in K^+} \xi_k^2\gamma_k, && a_9 = \frac{1}{2}\sum_{k\in K^-} \eta_k^2\gamma_k, \\ &a_{10} = \frac{1}{2}\sum_{k\in K^+} \eta_k^2\gamma_k, && a_{11} = \sum_{k\in K^-} \xi_k\eta_k\gamma_k, && a_{12} = \sum_{k\in K^+} \xi_k\eta_k\gamma_k. \end{aligned} \tag{3.3}$$

Using the interface relations (2.4), we eliminate the quantities from one side, say, the + side, using the quantities from the other side, say, the − side, and collect terms to get an expression of the form

(3.4)
$$\begin{aligned} T_{ij} \;=\; & \left(a_1 + \frac{a_8\,[\kappa]}{\beta^+} + a_2\right) u^- + \left\{ a_3 + a_8\left(\frac{\beta_\xi{}^-}{\beta^+} - \chi''\right) + a_{10}\chi'' + a_{12}\frac{\beta_\eta{}^-}{\beta^+} \right. \\ & \left. +\ \rho\left(a_4 + a_8\left(\chi'' - \frac{\beta_\xi{}^+}{\beta^+}\right) - a_{10}\chi'' - a_{12}\frac{\beta_\eta{}^+}{\beta^+}\right) - \beta_\xi{}^- \right\} u_\xi^- \\ & + \left\{ a_5 + a_6 + a_8\left(\frac{\beta_\eta{}^-}{\beta^+} - \frac{\beta_\eta{}^+}{\beta^+}\right) + a_{12}(1-\rho)\,\chi'' - \beta_\eta{}^- \right\} u_\eta^- \\ & + \left\{a_7 + a_8\rho - \beta^-\right\} u_{\xi\,\xi}^- + \left\{a_9 + a_{10} + a_8\,(\rho - 1) - \beta^-\right\} u_{\eta\,\eta}^- \\ & + \{a_{11} + a_{12}\rho\}\, u_{\xi\,\eta}^- - \kappa^- u^- - f^- + (\hat{T}_{ij} - C_{ij}) + O(h), \end{aligned}$$

where

$$\begin{aligned} \hat{T}_{ij} &= a_2\, w + a_{12}\,\frac{v'}{\beta^+} + \left(a_6 - \frac{a_8\,\beta_\xi{}^+}{\beta^+} + a_{12}\chi''\right) w' \\ &\quad + a_{10}\, w'' + \frac{1}{\beta^+}\left(a_4 + a_8(\chi'' - \frac{\beta_\xi{}^+}{\beta^+}) - a_{10}\chi'' - a_{12}\frac{\beta_\eta{}^+}{\beta^+}\right) v \\ &\quad + a_8\left\{\frac{[f]}{\beta^+} + \frac{\kappa^+\, w}{\beta^+} - w''\right\}. \end{aligned} \tag{3.5}$$

We want to make the magnitude of the truncation error as small as possible by choosing γ_k's so that some of coefficients of u^-, u_ξ^-, u_η^-, $\ldots$ vanish:

$$\begin{aligned}
a_1 + a_2 + a_8 \frac{[\kappa]}{\beta^+} &= 0, \\
a_3 + \rho a_4 + a_8 \frac{{\beta_\xi}^- - \rho {\beta_\xi}^+ - [\beta]\chi''}{\beta^+} + a_{10} \frac{[\beta]\chi''}{\beta^+} + a_{12} \frac{{\beta_\eta}^- - \rho {\beta_\eta}^+}{\beta^+} &= {\beta_\xi}^-, \\
a_5 + a_6 - a_8 \frac{[\beta_\eta]}{\beta^+} + a_{12} (1-\rho)\, \chi'' &= {\beta_\eta}^-, \\
a_7 + a_8\, \rho &= \beta^-, \\
a_9 + a_{10} + a_8\, (\rho - 1) &= \beta^-, \\
a_{11} + a_{12}\, \rho &= 0.
\end{aligned} \tag{3.6}$$

Once the γ_k's are chosen, we set $C_{ij} = \hat{T}_{ij}$, where $\hat{T}_{ij}$ is given by (3.5). If we use a six-point stencil and equations (3.6) hold, then this leads to the original IIM [7].

3.1. An optimization approach. In order to obtain finite difference schemes that satisfy the discrete maximum principle, see section 6.5 of Morton and Mayers [12] for the definition. We need to impose the sign restriction on the coefficients γ_k's in (3.1)

$$\gamma_k \geq 0 \quad \text{if} \quad (i_k, j_k) \neq (0,0), \qquad \gamma_k < 0 \quad \text{if} \quad (i_k, j_k) = (0,0), \tag{3.7}$$

along with several equations in (3.6). At regular grid points, the standard central finite difference scheme satisfies the sign restriction and the equations in (3.6). So we will concentrate our discussion only on an irregular grid point (x_i, y_j). We form the following constrained quadratic optimization problem to determine the coefficients of the finite difference scheme

$$\min_{\gamma} \left\{ \frac{1}{2} \|\gamma - g\|_2^2 \right\} \qquad \text{subject to} \tag{3.8}$$

$$A\gamma = b, \quad \gamma_k \geq 0 \quad \text{if} \quad (i_k, j_k) \neq (0,0), \quad \gamma_k < 0 \quad \text{if} \quad (i_k, j_k) = (0,0), \tag{3.9}$$

where $\gamma = [\,\gamma_1, \gamma_2, \ldots, \gamma_{n_s}\,]^T$ is the vector composed of the coefficients of the finite difference scheme, $g \in R^{n_s}$, and $A\gamma = b$ is the system of linear equations that contains several or all equations in (3.6). Naturally we want to choose $\{\gamma_k\}$ in such a way that they become the coefficients of the standard five-point central difference scheme if $\beta^+ = \beta^-$ is a constant. This can be done by selecting the vector g as

$$\begin{aligned}
g_k &= \frac{\beta_{i+i_k, j+j_k}}{h^2} \quad \text{if} \quad (i_k, j_k) \in \{(-1,0), (1,0), (0,-1), (0,1)\,\}; \\
g_k &= -\frac{4\beta_{i,j}}{h^2} \quad \text{if} \quad (i_k, j_k) = (0,0); \qquad g_k = 0 \quad \text{otherwise}.
\end{aligned} \tag{3.10}$$

There are two parameters in the optimization algorithm (3.8)–(3.9) that are to be determined. The first one is the set of grid points (x_{i_k}, y_{j_k}), whose total number is n_s, involved in the finite difference scheme. The second one is the set of equations from (3.6), which is used as the equality constraint in (3.9). The solution $\{\gamma_k\}$ to the constrained optimization problem is the coefficients of the finite difference scheme at the particular irregular grid point. We will discuss a first, as well as a second, order method in this paper.

3.2. The first order method. In this section we describe the first order method. Note that $A_i\gamma = b_i$, $1 \le i \le 3$, the first three equations in (3.6) represent the leading terms in (3.4), and the last three equations $A_l\gamma = b_l$, $4 \le l \le 6$ in (3.6) are one order higher in terms of h than the first three equations. To obtain a first order method, it is sufficient to select the first three equations as the equality constraints. Moreover, since the $O(h^2)$ terms have little effect on the convergence order, we neglect them to get simplified equality constraints

$$a_1 + a_2 + a_8\frac{[\kappa]}{\beta^+} = 0, \qquad a_3 + \rho a_4 = {\beta_\xi}^-, \qquad a_5 + a_6 = {\beta_\eta}^- \tag{3.11}$$

for the first three equations. We take $n_s = 5$, and the standard five-point stencil

$$\{(i_1,j_1),(i_2,j_2),(i_3,j_3),(i_4,j_4),(i_5,j_5)\} = \{(-1,0),(0,0),(1,0),(0,-1),(0,1)\}\,.$$

Thus, the optimization problem (3.8) is written as

$$\min_{\gamma} \left\{ \frac{1}{2}\|\gamma - g\|_2^2 + \sum_{l=4}^{6} w_l \left(A_l\gamma - b_l\right)^2 \right\} \qquad \text{subject to (3.11) and} \tag{3.12}$$
$$\gamma_k \ge 0 \quad \text{if} \quad (i_k, j_k) \ne (0,0), \qquad \gamma_k < 0 \quad \text{if} \quad (i_k,j_k) = (0,0),$$

where w_l are predefined weights for $l = 4, 5, 6$ (we often simply take $w_l = 0$). It will be shown in section 4 that the resulting finite difference scheme is of first order.

3.3. The second order method. In order to get a second order method, we require all six equations in (3.6) to be satisfied plus the sign restriction (3.7) for the optimization problem. Therefore $n_s \ge 6$. In the conforming finite element method for interface problems with a uniform triangulation [10], a nonstandard nine-point stencil is used for the second order method and the stiffness matrix is symmetric positive definite. Since we do not require symmetry in the linear system of the finite difference equations described in this paper, we expect that for the standard nine-point stencil ($n_s = 9$) the optimization problem has solutions. We will see this through numerical verifications in section 5. Moreover, we prefer to use the standard nine-point stencil so that the resulting linear system of equations is block tridiagonal and a multigrid solver DMGD9V (developed for the standard nine point stencil) can be used.

3.4. Solving the optimization problems. There are several commercial and educational software packages that are designed to solve constrained quadratic optimization problems, for example, the *QP* function in Matlab; the *QL* program using Fortran computer language developed by Schittkowski [13]; and the *IQP* Fortran code from *Port* managed by Lucent Technologies. The information about these software packages can be found through the Internet.

Most of the quadratic optimization solvers require users to provide an initial guess, lower and upper bounds, and other information. We take the initial guess as the vector g, and the lower and upper bounds as

$$0 < \gamma_k < \frac{\beta_{\max}}{h^2} \quad \text{if} \quad (i_k,j_k) \ne (0,0); \qquad -\frac{4\beta_{\max}}{h^2} < \gamma_k < 0 \quad \text{if} \quad (i_k,j_k) = (0,0),$$

where $\beta_{\max}$ is an estimation of the upper bound of the coefficient $\beta(x,y)$.

In case that the optimization solver fails to give a solution, we can either increase the number of grid points or switch to a first order scheme that satisfies the discrete maximum principle, such as the smoothing method, without affecting global accuracy. If the optimization solver fails to return a feasible solution, it may mean either the *curvature* or the *jump* in β is too large relative to the underlying grid.

4. Convergence analysis for the first order method.

4.1. Existence of the solution to the optimization problem. In this subsection, we assume that $\beta(x, y)$ is piecewise constant in the solution domain although the method itself does not have such a restriction; see Example 6.1. We first show that the constrained optimization problem (3.12) has a solution and then show that the resulting finite difference scheme is of first order.

Since we use the standard five-point stencil ($n_s = 5$), the stencil at an irregular grid point (x_i, y_j) is composed of (x_{i+i_k}, y_{j+j_k}) with

$$(i_k, j_k) = \{\, (i-1, j), (i, j), (i+1, j), (i, j-1), (i, j+1)\, \} \tag{4.1}$$

corresponding to $k = 1, 2, 3, 4, 5$. Since the linear system of equations is obtained from the Taylor expansion at (x_i^*, y_j^*), a point on the interface, we need only use the limiting value of $\beta(x, y)$ at (x_i^*, y_j^*), which we denote as β^+ and β^-, respectively. Assuming $\kappa = 0$ in (1.1), we can eliminate γ_2 from the first equation in (3.11) to get

$$\gamma_2 = -\sum_{k=1,k\neq 2}^{5} \gamma_k. \tag{4.2}$$

The equality constraints from $a_3 + \rho a_4 = 0$ and $a_1 + a_2 = 0$ in (3.11) with high order terms of h being neglected and the inequality constraints of the optimization problem are the following:

$$\sum_{k=1,k\neq 2}^{5} \left(\frac{\xi_k}{\beta^\pm} - \frac{\xi_2}{\beta^\pm} \right) \gamma_k = 0, \qquad \sum_{k=1,k\neq 2}^{5} (\eta_k - \eta_2)\, \gamma_k = 0, \tag{4.3}$$

$$\gamma_k \geq 0, \quad k = 1, 3, 4, 5, \tag{4.4}$$

where $\beta^\pm$ takes either β^+ or β^- depending on which side of the interface the grid point (x_{i+i_k}, y_{j+j_k}) is. We will prove the following theorem.

THEOREM 4.1. *Let (x_i^*, y_j^*) be a point on a smooth interface Γ that satisfies*

$$|x_i^* - x_i| + |y_j^* - y_j| < \sqrt{2}\, h. \tag{4.5}$$

Assume that the interface Γ cuts through the axes at no more than two points within $[x_i - h,\, x_i + h] \times [y_j - h,\, y_j + h]$, and the tangent line at (x_i^*, y_j^*) separates the grid points in the five-point stencil in the same way as the interface does. Then there is at least one set of solutions $\gamma_k > 0$, $k = 1, 3, 4, 5$, that satisfies (4.3).

Proof. Without loss of generality, and for simplicity for indexing, we assume that $x_i^* - x_i \geq 0$, $y_j^* - y_j \geq 0$, the angle between the x-axis and the normal direction θ is in the interval $[\,0, \pi/2)$. There are two different cases depending on which side the grid point (x_i, y_{j+1}) is. We will prove one of the cases as shown in Figure 4.1 in which the condition (4.5) is assumed. The proof of the other case is technically similar and will be omitted. From Figure 4.1, it is easy to see that

$$\begin{aligned} &\xi_1 < 0, \quad \xi_3 > 0, \quad \xi_4 \leq 0, \quad \xi_5 \geq 0, \\ &\eta_1 \geq 0, \quad \eta_3 \leq 0, \quad \eta_4 < 0, \quad \eta_5 > 0. \end{aligned}$$

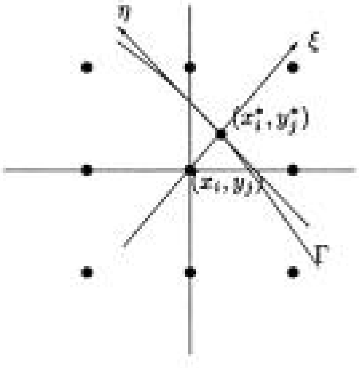


FIG. 4.1. *A diagram of the geometry at an irregular grid point if h is sufficiently small.*

With such a geometry, the two equations (4.3) are

$$\left(\frac{\xi_1}{\beta^-}-\frac{\xi_2}{\beta^-}\right)\gamma_1+\left(\frac{\xi_3}{\beta^+}-\frac{\xi_2}{\beta^-}\right)\gamma_3+\left(\frac{\xi_4}{\beta^-}-\frac{\xi_2}{\beta^-}\right)\gamma_4+\left(\frac{\xi_5}{\beta^+}-\frac{\xi_2}{\beta^-}\right)\gamma_5=0,$$

$$(\eta_1-\eta_2)\,\gamma_1+(\eta_3-\eta_2)\,\gamma_3+(\eta_4-\eta_2)\,\gamma_4+(\eta_5-\eta_2)\,\gamma_5=0. \tag{4.6}$$

Introduce the parameters α_{ij} as in the following:

$$\alpha_{11}=-\left(\frac{\xi_1}{\beta^-}-\frac{\xi_2}{\beta^-}\right)>0,\quad \alpha_{13}=\frac{\xi_3}{\beta^+}-\frac{\xi_2}{\beta^-}>0,$$
$$\alpha_{14}=-\left(\frac{\xi_4}{\beta^-}-\frac{\xi_2}{\beta^-}\right)\geq 0,\quad \alpha_{15}=\frac{\xi_5}{\beta^+}-\frac{\xi_2}{\beta^-}\geq 0,$$
$$\alpha_{23}=-(\eta_3-\eta_2)\geq 0,\quad \alpha_{21}=(\eta_1-\eta_2)\geq 0,$$
$$\alpha_{24}=-(\eta_4-\eta_2)>0,\quad \alpha_{25}=(\eta_5-\eta_2)>0.$$

We can rewrite the equations (4.6) as[2]

$$\alpha_{13}\gamma_3-\alpha_{11}\gamma_1=\alpha_{14}\gamma_4-\alpha_{15}\gamma_5,\qquad \alpha_{23}\gamma_3-\alpha_{21}\gamma_1=-\alpha_{24}\gamma_4+\alpha_{25}\gamma_5. \tag{4.7}$$

We are ready to prove the theorem by distinguishing the following cases.

- $\alpha_{23}=0$. In this case, we also have $\alpha_{21}=0$. We choose

$$\gamma_5=\frac{\beta^+}{h^2}>0,\qquad \gamma_4=\frac{\alpha_{25}}{\alpha_{24}}\gamma_5>0,$$
$$\gamma_1=\frac{2\beta^-}{h^2}+\frac{|\,\alpha_{14}\gamma_4-\alpha_{15}\gamma_5\,|}{\alpha_{11}}>0,\qquad \gamma_3=\frac{1}{\alpha_{13}}\left(\alpha_{11}\gamma_1+\alpha_{14}\gamma_4-\alpha_{15}\gamma_5\right)>0.$$

- $\alpha_{14}=0$. In this case, we also have $\alpha_{15}=0$. We choose

$$\gamma_1=\frac{\beta^-}{h^2}>0,\qquad \gamma_3=\frac{\alpha_{11}}{\alpha_{13}}\gamma_1>0,$$
$$\gamma_4=\frac{2\beta^-}{h^2}+\frac{|\,\alpha_{23}\gamma_3-\alpha_{21}\gamma_1\,|}{\alpha_{24}}>0,\qquad \gamma_5=\frac{1}{\alpha_{25}}\left(\alpha_{24}\gamma_4+\alpha_{23}\gamma_3-\alpha_{21}\gamma_1\right)>0.$$

[2] It is also easy to show that $\alpha_{23}=\alpha_{21}$ and $\alpha_{24}=\alpha_{25}$.

- $\dfrac{\alpha_{11}}{\alpha_{21}} = \dfrac{\alpha_{13}}{\alpha_{23}} = \tau > 0$. We choose

$$\gamma_4 = \frac{\beta^-}{h^2} > 0, \qquad \gamma_5 = \frac{\alpha_{14} + \tau\alpha_{24}}{\alpha_{15} + \tau\alpha_{25}}\,\gamma_4.$$

With such choice of γ_4 and γ_5, we have

$$\alpha_{14}\gamma_4 - \alpha_{15}\gamma_5 = -\tau\alpha_{24}\gamma_4 + \tau\alpha_{25}\gamma_5.$$

Then we choose

$$\gamma_1 = \frac{2\beta^-}{h^2} + \frac{|\,\alpha_{14}\gamma_4 - \alpha_{15}\gamma_5|}{\alpha_{11}}, \qquad \gamma_3 = \frac{1}{\alpha_{13}}\left(\alpha_{11}\gamma_1 + \alpha_{14}\gamma_4 - \alpha_{15}\gamma_5\right).$$

- $\dfrac{\alpha_{11}}{\alpha_{13}} > \dfrac{\alpha_{21}}{\alpha_{23}}$. This is one of the general cases. Consider the following two functions:

$$g_1(\gamma_4, \gamma_5) = \alpha_{14}\gamma_4 - \alpha_{15}\gamma_5, \qquad g_2(\gamma_4, \gamma_5) = -\alpha_{24}\gamma_4 + \alpha_{25}\gamma_5.$$

In the first quadrant of the γ_4-γ_5 plane, there are three different regions where $g_1 > g_2$, $g_1 = g_2$, and $g_1 < g_2$, respectively. Choose a point $\gamma_4^* > 0$, $\gamma_5^* > 0$ in the first quadrant of the γ_4-γ_5 plane in such a way that $g_1 < g_2$. Let γ_1^*, γ_2^* be the solution to the system of equations (4.7) with γ_4 and γ_5 being substituted by γ_4^* and γ_5^*; since $\frac{\alpha_{11}}{\alpha_{13}} > \frac{\alpha_{21}}{\alpha_{23}}$, we can conclude that $\gamma_1^* > 0$ and $\gamma_2^* > 0$.

- $\dfrac{\alpha_{11}}{\alpha_{13}} < \dfrac{\alpha_{21}}{\alpha_{23}}$. The proof is almost exactly the same as the previous case except we choose $\gamma_4^* > 0$, $\gamma_5^* > 0$ such that $g_1 > g_2$. This completes the proof. □

COROLLARY 4.2. *From the proof of the existence of the optimization problem, we can conclude that*

$$|\gamma_k| \le \frac{C}{h^2}, \quad k = 1, 2, \dots, 5,$$

and there is at least one of γ_k from each side of the interface such that

$$|\gamma_k| \ge \frac{\bar{C}}{h^2},$$

where the constants are $O(1)$ and independent of the grid points involved but depend on the coefficient β.

These results of the corollary imply the condition (4.12) in Theorem 4.4 in the next subsection.

4.2. Convergence proof of the first order method. We need the following lemma which is a generalization of Theorem 6.1 and Theorem 6.2 of Morton and Mayers [12] for multiple subregions J_i.

LEMMA 4.3. *Given a finite difference scheme L_h defined on a discrete set of interior points J_Ω for a Dirichlet elliptic PDE, assume the following conditions hold:*

1. *J_Ω can be partitioned into a number of disjoint regions*

$$J_\Omega = J_1 \cup J_2 \cup J_3 \cup \cdots \cup J_{N_\Omega}, \quad J_i \cap J_k = \emptyset \quad \text{if } i \ne k. \tag{4.8}$$

2. *The truncation error of the finite difference scheme at a grid point p satisfies*

$$|T_p| \le T_i \quad \forall p \in J_i, \quad i = 1,\, 2,\, \dots,\, N_\Omega. \tag{4.9}$$

3. *There exists a nonnegative mesh function ϕ defined on $\cup_{i=1}^{N_\Omega} J_i$ satisfying*

$$L_h\phi_p \geq C_i > 0 \quad \forall p \in J_i, \qquad i = 1, 2, \ldots, N_\Omega. \tag{4.10}$$

Then the global error of the approximate solution U_{ij} from the finite difference scheme at mesh points is bounded by

$$||E_h||_\infty \leq \left(\max_{A\in J_{\partial\Omega}} \phi_A \right) \max_{1\leq i\leq N_\Omega} \left\{ \frac{T_i}{C_i} \right\}, \tag{4.11}$$

where E_h is the difference of the exact solution of the differential equation and the approximate solution of the finite difference equations at the mesh points, and $J_{\partial\Omega}$ is the set that contains the boundary points.

The proof of this lemma is trivial and is omitted.

Theorem 4.4. *Let $u(x, y)$ be the exact solution to* (1.1) *and* (1.2) *with $\kappa = 0$ and a Dirichlet boundary condition. Assume that* (1) *$u(x, y)$ has* piecewise *continuous second order derivatives;* (2) *h is sufficiently small;* (3) *at all irregular grid points, the following is true:*

$$|\gamma_k| \leq \frac{C_1}{h^2}, \quad k = 1, 2, \ldots, 5, \quad \text{and} \sum_{\xi_k\geq 0} \gamma_k\xi_k \geq \frac{C_2}{h}. \tag{4.12}$$

Then we have the following error estimate for U_{ij}, the solution of the finite difference equations obtained from the first order optimization method

$$\|u(x_i, y_j) - U_{ij}\|_\infty \leq Ch, \tag{4.13}$$

where $C = O(1)$ depends on the underlying grid and the interface, f, u, and the coefficient β.

Proof. If $h > 0$ is sufficiently small, we have proved that the optimization problem has solutions. Consider the solution to the following interface problem:

$$\begin{aligned} &\nabla \cdot \beta\nabla\phi = 1, \\ &\qquad [\phi] = 0, \qquad [\beta\phi_n] = 1, \qquad \phi_{\partial\Omega} = 1. \end{aligned} \tag{4.14}$$

From the results in [2, 3], we know that the solution ϕ exists, and it is unique and piecewise continuous. Therefore the solution is also bounded. Let

$$\bar\phi(x, y) = \phi(x, y) + \left| \min_{(x,y)\in\Omega} \phi(x, y) \right|. \tag{4.15}$$

Then $\bar\phi(x, y) \geq 0$ and still satisfies the differential equation and the jump conditions in (4.14). We define the discrete finite difference operator as

$$L_h u(x_i, y_j) = \sum_k^{n_s} \gamma_k\, u(x_{i+i_k}, y_{j+j_k}). \tag{4.16}$$

Let (x_i, y_j) be an irregular grid point. If h is small enough, then we know that γ_k is bounded by C/h^2. Therefore we have

$$\begin{aligned}
L_h\bar{\phi}(x_i,y_j) &= \sum_k^{n_s} \gamma_k\,\bar{\phi}(x_{i+i_k}, y_{j+j_k}) \\
&= (a_1+a_2)\bar{\phi}^- + \left(a_3+\rho a_4 - a_8\frac{[\beta]\chi''}{\beta^+}\right)\bar{\phi}_\xi^- \\
&\quad + \left(a_5+a_6-a_8\frac{[\beta_\eta]}{\beta^+}+a_{12}(1-\rho)\,\chi''\right)\bar{\phi}_\eta^- + \sum_{\xi_k\ge 0}\gamma_k\xi_k + O(1) \\
&= \sum_{\xi_k\ge 0}\gamma_k\xi_k + O(1),
\end{aligned}$$

where a_k are defined in (3.3). Note that $a_8\beta\chi''/\beta$, $a_8\beta_\eta/\beta$, and $a_{12}(1-\rho)\,\chi''$ are $O(1)$ provided that $\gamma_k \le C_1/h^2$. Thus from (4.12), the comparison function $\bar{\phi}$ satisfies

$$L_h\bar{\phi}(x_i,y_j) = \begin{cases} 1+O(h^2) & \text{if } (x_i,y_j) \text{ is a regular grid point,} \\ \displaystyle\sum_{\xi_k\ge 0}\gamma_k\xi_k \ge \frac{C_1}{h}+O(1) & \text{if } (x_i,y_j) \text{ is an irregular grid point.} \end{cases}$$

At a *regular* grid point we have

$$\frac{|T_{ij}|}{L_h\bar{\phi}(x_i,y_j)} \le \frac{C_2h^2}{1} = C_2h^2. \tag{4.17}$$

At an irregular grid point we have

$$\frac{|T_{ij}|}{L_h\bar{\phi}(x_i,y_j)} \le \frac{C_3}{C_1/h} = \frac{C_3}{C_1}h. \tag{4.18}$$

From Lemma 4.3, we conclude the global error then satisfies

$$\|u(x_i,y_j)-U_{ij}\|_\infty \le \left(\max_{(x,y)\in\partial\Omega}\phi(x,y)\right)\max\left\{\frac{C_3}{C_1}, C_2h\right\}h \le Ch.$$

Remarks. Although the theorem is proved for the case $\kappa = 0$, we should be able to claim that the condition (4.12) is also true when $\kappa \ge 0$ and so is the theorem, since $\kappa \ge 0$ increases the diagonal dominance of the resulting linear system of equations. When $\kappa < 0$, the original differential equation may not be well-posed, and the condition may not be satisfied.

5. Convergence of the second order method. In this section we first show the existence of the solution to the optimization problem numerically and then argue that the resulting finite difference scheme is of second order.

5.1. Numerical verification of the existence of the solution to the optimization problem. We again assume that $\kappa = 0$ in (1.1) (see the remarks in the previous section), and h is small enough so that the interface behaves like a straight line relative to the underlying grid. Under these conditions, those terms that contain χ'' in (3.6) are high order terms compared to those in a_3 and ρa_4 in terms of h and therefore can be neglected. For the second order method, if we scale the system of equations (3.6), then β_ξ^- and β_η^- in the second and third equations in (3.6) are high order terms compared with β^- in the fourth and fifth equations in (3.6) in terms of h, therefore we can simply set $\beta_\xi^- = \beta_\eta^- = 0$ as long as h is sufficiently small.

CONJECTURE 5.1. *Let (x_i, y_j) be an irregular grid point, and let (x_i^*, y_j^*) be its orthogonal projection on the interface. Then the optimization problem defined in* (3.8)–(3.9) *has solutions. The solution of the coefficients $\{\gamma_k\}$ also satisfies*

$$\left|\frac{\gamma_k}{\beta^-}\right| \le \frac{C}{h^2}, \quad k = 1, \dots, 9. \tag{5.1}$$

Furthermore there are γ_k's from each side of the interface such that

$$\left|\frac{\gamma_k}{\beta^-}\right| \ge \frac{C_1}{h^2}, \tag{5.2}$$

and thus

$$\sum_{\xi_k \ge 0} \gamma_k \xi_k \ge \frac{C_2}{h} \tag{5.3}$$

provided that $\max_{\xi_k \ge 0}\{\,\xi_k\} \ge C_3\, h$ *for some* $C_3 > 0$. *The constants are* $O(1)$ *which depend on the coefficient* β.

Numerical verification of Conjecture 5.1. First we shift and scale the problem in the following way:

$$\bar{x} = \frac{x - x_i}{h}, \qquad \bar{y} = \frac{y - y_j}{h}, \qquad \bar{\gamma}_k = \frac{\gamma_k}{h^2}.$$

For simplicity, we will use the same notation without bars. The nine-point stencil then corresponds to the square $-1 \le x, y \le 1$. With the local coordinates (2.1), we can just consider the case that the orthogonal projection is in the first quadrant.

Given any point (x^*, y^*) on the interface, and an angle θ

$$0 \le x^* \le 1, \qquad 0 \le y^* \le 1 - x^*, \qquad 0 \le \theta < \frac{\pi}{2}. \tag{5.4}$$

The tangent line of the interface at (x^*, y^*),

$$(x - x^*)\,\cos\theta + (y - y^*)\,\sin\theta = 0, \tag{5.5}$$

is a good approximation to the interface if $h > 0$ is sufficiently small so that $\chi''\,h^2$ is negligible.

The interface cuts the square $-1 \le x,\ y \le 1$ into two parts. We denote the side which contains the origin as $-$ side and the other half as $+$ side. We also scale the coefficient β either as $\beta^- = 1$, $\beta^+ = 1/\rho$, or $\beta^+ = 1$, $\beta^- = \rho$. The optimization problem is

$$\min_\gamma \frac{1}{2}\sum_{k=1}^{9} (\gamma_k - g_k)^2 \qquad \text{subject to}$$

$$A\gamma = b, \quad \gamma_k \ge 0 \quad \text{if} \quad (i_k, j_k) \ne (0,0); \quad \gamma_k < 0 \quad \text{if} \quad (i_k, j_k) = (0,0),$$

where

$$g_k = \beta^\pm \quad \text{if} \quad (i_k, j_k) \in \{(-1,0), (1,0), (0,-1), (0,1)\,\};$$

$$g_k = -4 \quad \text{if} \quad (i_k, j_k) = (0,0); \qquad g_k = 0 \quad \text{otherwise},$$

and $A\gamma = b$ is the following system of equations from (3.6):

$$\begin{aligned} &a_1 + a_2 = 0, && a_3 + \rho a_4 = 0, && a_5 + a_6 = 0,\\ &a_7 + a_8\,\rho = \beta^-, && a_9 + a_{10} + a_8\,(\rho - 1) = \beta^-, && a_{11} + a_{12}\,\rho = 0. \end{aligned}$$

To solve the optimization problem numerically, we use a uniform grid:

$$r_i = i\,\Delta r, \quad \Delta r = \frac{1}{M}, \quad i = 0, 1, \ldots, M, \quad \theta_j = j\,\Delta\theta, \quad \Delta\theta = \frac{\pi}{2N}, \quad j = 0, 1, \ldots, N,$$

$$\rho_k = 10^{-N_1 + k\,\Delta\rho}, \quad \Delta\rho = \frac{N_2 + N_1}{L}, \quad k = 0, 1, \ldots, L.$$

The projection then is $x_i^* = r_i\,\cos\theta_j$, $y_i^* = r_i\,\sin\theta_j$ excluding those $y_i^* > 1 - x_i^*$ that are outside of the five-point stencil. Define

$$\gamma_{max}(\rho) = \max_{r_i,\theta_j}\max_{1\le k\le 9}\frac{|\gamma_k|}{\beta^-} = \max_{r_i,\theta_j}\frac{|\gamma_5|}{\beta^-}, \qquad \gamma_{min}(\rho) = \min_{r_i,\theta_j}\frac{|\gamma_5|}{\beta^-},$$

$$S_{min}(\rho) = \min_{r_i,\theta_j,\ \max\limits_{\xi_k\ge 0}\{\xi_k\}\,\ge\, C_3 h}\ \sum_{\xi_k\ge 0}\xi_k\gamma_k.$$

Our numerical results show that the solution to the optimization problem always exits. Figure 5.1 summarizes the numerical verification results for Conjecture 5.1. In Figures 5.1(a) and (b), the dashed line is $\gamma_{max}(\rho)$ and it is bounded by $|\gamma_k|\,h^2 \le 10$. The solid line is $\gamma_{min}(\rho)$ and $|\gamma_5|\,h^2 \ge 1$. If $\beta^- = \beta^+$, we will have $\gamma_5 h^2 = 4$ exactly as we can see from Figure 5.1. Figures 5.1(a) and (b) confirm the inequalities (5.1) and (5.2).

In Figures 5.1(c) and (d), we plot $hS_{min}(\rho)/C_2$, where

$$C_2 = \frac{\max\{\beta^-/\beta^+, \beta^-/\beta^+\}}{\max\{\,1,\ \beta^-\}\ \max\{\,1,\ \beta^+\}}. \tag{5.6}$$

This constant was found by numerical experiments. The minimum of $\sum_{\xi_k\ge 0}\gamma_k\xi_k$ is taken in all cases except for the point $(1, 0)$ where the interface actually is $x = 1$. In this case, the grid point touches the interface and the finite difference scheme is the standard centered scheme using the five-point stencil with possible nonzero correction C_{ij} for the jump in the solution and the flux. In Figures 5.1(a) and (b), we have

$$\sum_{\xi_k\ge 0}\gamma_k\xi_k \ge \frac{0.01\,C_2}{h} \quad \text{if} \quad \max_{\xi_k\ge 0}\{\xi_k\} \ge C_3 h.$$

Thus the numerical verification confirms the inequalities (5.1)–(5.3). We have tried different grid sizes and all the results showed the same conclusions. The ratio ρ of the jump in β ranges from 10^{-9} to 10^{10}, which should cover most applications.

The complete theoretical proof of the theorem is still an open problem. Although we are able to prove the conjecture for special values of ρ, for example, $\rho \ge 1$, we omit those proofs due to the space limitation.

5.2. Convergence analysis of the second order method. Parallel to the convergence result for the first order method, we have the following theorem.

THEOREM 5.1. *Let $u(x, y)$ be the exact solution to* (1.1) *and* (1.2) *with $\kappa \ge 0$ and a Dirichlet boundary condition. Assume* (1) *the optimization problem* (3.8)–(3.9)

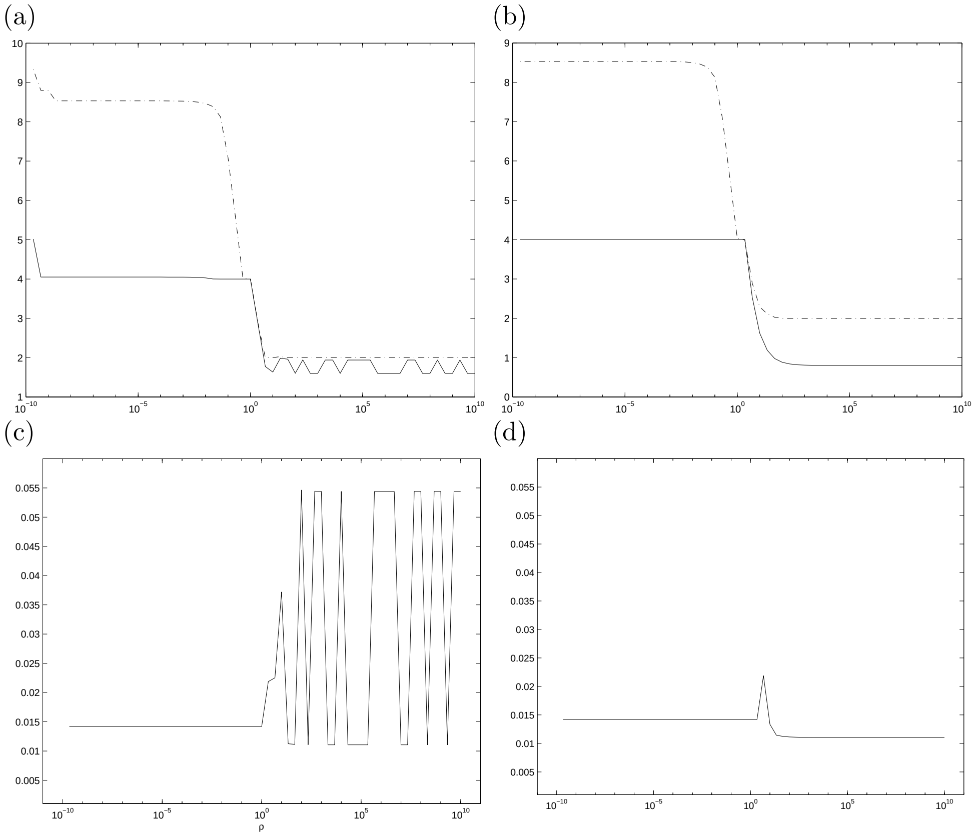


FIG. 5.1. *Numerical verification of Conjecture* 5.1 *with $m = n = L = 60$, $N_1 = 9$, and $N_2 = 10$.* (a) *Plots of $\gamma_{max}(\rho)$ (dash-dotted line) and $\gamma_{min}(\rho)$ (solid line) versus ρ in log scale with $\beta^+ = 1$, $\beta^- = \rho$.* (b) *The same plots as* (a) *with $\beta^- = 1$, $\beta^+ = 1/\rho$.* (c) *Plot of $S_{min}(\rho)$, whose minimum is* 0.0111*, with $\beta^+ = 1$, $\beta^- = \rho$.* (d) *Plot of $S_{min}(\rho)$, whose minimum again is* 0.0111*, with $\beta^- = 1$, $\beta^+ = 1/\rho$.*

with the six-equation constraint (3.6) *using the standard nine-point stencil has a set of solutions $\{\gamma_k\}$ at every irregular grid point;* (2) *$u(x, y)$ has* piecewise *continuous third order derivatives;* (3) *h is sufficiently small;* (4) *the following is true:*

$$|\gamma_k| \le \frac{C_1}{h^2} \quad \textit{and} \quad \sum_{\xi_k \ge 0} \gamma_k \xi_k \ge \frac{C_2}{h}. \tag{5.7}$$

Then we have the following error estimate for U_{ij}, the solution of the finite difference scheme obtained from the second order method

$$\|u(x_i, y_j) - U_{ij}\|_\infty \le Ch^2, \tag{5.8}$$

where the constant C depends on the underlined grid and interface, as well as u, f, and β.

Proof. If (5.7) is true, then we know that

$$L_h\bar{\phi}(x_i, y_j) \ge \left\{ \begin{array}{ll} 1 + O(h^2) & \text{if } (x_i, y_j) \text{ is a regular grid point,} \\ \displaystyle\sum_{\xi_k \ge 0} \gamma_k \xi_k \ge \frac{C_1}{h} + O(1) & \text{if } (x_i, y_j) \text{ is an irregular grid point,} \end{array} \right.$$

where $\bar{\phi}(x,y)$ is defined in (4.15). Note that at some regular grid points, $L_h\tilde{\phi}(x_i,y_j)$ can be very large, but it is nonnegative. Thus, the first inequality above still holds. Therefore at a regular point we have

$$\frac{|T_{ij}|}{L_h\bar{\phi}(x_i,y_j)} \le \frac{C_3h^2}{1}.$$

At an irregular grid point where (5.7) is satisfied, we have

$$\frac{|T_{ij}|}{L_h\bar{\phi}(x_i,y_j)} \le \frac{C_4h}{C_2/h} = \frac{C_4}{C_2}h^2$$

since the truncation error is bounded by

$$|T_{ij}| \le C_4h$$

for some constant C_4 that depends on the second derivatives of the solution on each side of the interface. Thus, from Lemma 4.3, we have proved the quadratic convergence.

Remarks. We have numerically verified that equation (5.7) holds as long as $\max_{\xi_k\ge 0}\{\xi_k\} > C_3h$. The second condition in (5.7) may be violated when the interface is very close to a grid point involved in the stencil other than (x_i,y_j) and almost parallel to a grid line in the neighborhood of (x_i,y_j). In this case, the finite difference scheme actually is very close to the standard finite difference scheme using the five-point stencil with a correction term to the right-hand side. This fact can be stated in the following theorem (the proof of the theorem is given in the appendix).

Theorem 5.2. *If the conditions* (1)–(3) *in Theorem* 5.1 *are satisfied and either* (5.7) *or*

$$\max_{\xi_k\ge 0}\{\xi_k\} \le C_5h^{1+\sigma} \quad \textit{with } C_5>0 \textit{ and } \sigma>0 \tag{5.9}$$

is true, then

$$\|u(x_i,y_j)-U_{ij}\|_\infty \le Ch^2, \tag{5.10}$$

where the constants depend on the underlying grid and the interface, u, f, and β.

6. Numerical results.

We have performed a number of numerical experiments for both the first order and the second order methods. The results agree with our analysis in sections 4 and 5. The computations are done using either Sun's Ultra-1 workstations or IBM SP2 machines. The code has not been parallelized. The linear system of equations is solved using the multigrid method DMGD9V developed by de Zeeu [4]. The interface is a closed curve in the solution domain and is expressed in terms of the periodic cubic spline interpolation as in [8]. The implementation of the methods is sequential and not optimized. In this section, the following notations are used: m and n with $m=n$ are the number of grid lines in the x- and y-directions; n_1 is the number of control points used in the cubic spline interpolation to represent the interface Γ; n_{coarse} and n_{finest} are the number of the coarsest and finest grid lines, respectively, when the multigrid solver DMGD9V is used; n_l is the number of levels used for the multigrid method.

Table 6.1

The comparison of the grid refinement analysis of the first and second order methods for Example 6.1 *with* $b = 10$, $C = 0.1$, *and* $n_{coarse} = 6$. *First and second order convergence are confirmed.*

			The first order method		The second order method	
n_{finest}	n_1	n_l	$\parallel E_n \parallel_\infty$	order	$\parallel E_n \parallel_\infty$	order
42	40	4	$5.0184\ 10^{-3}$		$4.8638e\ 10^{-4}$	
82	80	5	$1.7610\ 10^{-3}$	1.5652	$1.4476e\ 10^{-4}$	1.7484
162	160	6	$1.4726\ 10^{-3}$	0.2628	$3.0120\ 10^{-5}$	2.2649
322	320	7	$5.3827\ 10^{-4}$	1.4650	$8.2255\ 10^{-6}$	1.8726
642	640	8	$2.6156\ 10^{-4}$	1.0459	$2.0599\ 10^{-6}$	1.9975

Table 6.2

The grid refinement analysis of the second order method for Example 6.1 *with* $n_{coarse} = 9$. *Second order convergence is confirmed.*

			$b = 1000$, $C = 0.1$		$b = 0.001$, $C = 0.1$	
n_{finest}	n_1	n_l	$\parallel E_n \parallel_\infty$	order	$\parallel E_n \parallel_\infty$	order
34	40	3	$5.1361\ 10^{-4}$		9.3464	
66	80	4	$8.2345\ 10^{-5}$	2.7598	2.0055	2.3204
130	160	5	$1.8687\ 10^{-5}$	2.1878	$5.8084\ 10^{-1}$	1.8280
258	320	6	$4.0264\ 10^{-6}$	2.2394	$1.3741\ 10^{-1}$	2.1031
514	640	7	$9.430\ 10^{-7}$	2.1059	$3.5800\ 10^{-2}$	1.9514

6.1. The grid refinement analysis.

Example 6.1. This example is the same as the Example 2 in [7]. The interface is the circle $x^2 + y^2 = \frac{1}{4}$ within the computation domain $-1 \le x, y \le 1$. The equations are

$$(\beta u_x)_x + (\beta u_y)_y = f(x,y) + C \int_\Gamma \delta(\vec{x} - \vec{X}(s))\, ds, \tag{6.1}$$

with $\quad f(x,y) = 8\,(x^2+y^2) + 4 \quad$ and

$$\beta(x,y) = \begin{cases} x^2 + y^2 + 1 & \text{if } x^2 + y^2 \le \frac{1}{4}, \\ b & \text{if } x^2 + y^2 > \frac{1}{4}. \end{cases}$$

Dirichlet boundary conditions are determined from the exact solution

$$u(x,y) = \begin{cases} r^2 & \text{if } r \le \frac{1}{2}, \\ \left(1 - \frac{1}{8\,b} - \frac{1}{b}\right)/4 + \left(\frac{r^4}{2} + r^2\right)/b + C\,\log(2r)/b & \text{if } r > \frac{1}{2}. \end{cases} \tag{6.2}$$

In this example, we have *variable* and discontinuous coefficients. Tables 6.1 and 6.2 show the results of the grid refinement analysis using the first and second order methods for different choices of b and c. The maximum error over all grid points,

$$\parallel E_n \parallel_\infty = \max_{i,j} \mid u(x_i, y_j) - U_{ij} \mid,$$

is presented. The order of convergence is computed from

$$\text{order} = \left| \frac{\log\left(\parallel E_{n_1} \parallel_\infty / \parallel E_{n_2} \parallel_\infty\right)}{\log(n_1/n_2)} \right|,$$

TABLE 6.3

The grid refinement analysis of the second order method for Example 6.2 *with* $n_{coarse} = 9$*; second order convergence is confirmed.*

			$\beta^+ = 1000$, $\beta^- = 1$		$\beta^+ = 1$, $\beta^- = 1000$	
n_{finest}	n_1	n_l	$\parallel E_n \parallel_\infty$	order	$\parallel E_n \parallel_\infty$	order
34	40	3	$1.8322\ 10^{-1}$		$8.0733\ 10^{-3}$	
66	80	4	$3.5224\ 10^{-3}$	5.9574	$3.0371\ 10^{-3}$	1.4739
130	160	5	$4.5814\ 10^{-5}$	3.0090	$7.1981\ 10^{-4}$	2.1238
258	320	6	$1.4240\ 10^{-5}$	1.7049	$1.6876\ 10^{-4}$	2.1162
514	640	7	$3.1501\ 10^{-6}$	2.1887	$2.7407\ 10^{-5}$	2.6371

which is the solution of the equation

$$\parallel E_n \parallel_\infty = C\, h^{\text{order}}$$

with two different n's.

As explained in [9], for interface problems, the errors usually do not decline monotonously. Instead it depends on the relative location of the grid and the interface. However, the average of the orders approaches 1 and 2 for the first and second order methods in Tables 6.1 and 6.2. Compared with the result in [7], the new second order method gives a slightly better result. Notice that as the parameter b gets smaller, the solution in the outside of the interface becomes larger and the problem becomes harder to solve. But our second order method still converges quadratically.

Example 6.2. In this example, the coefficients of the differential equation are $\kappa = 0$ and β is piecewise constant, β^- and β^+. The jumps $[u]$ in the solution, $[\beta u_n]$ in the flux, and $[f]$ in the source term are chosen so that the following function is the exact solution:

$$u(x,y) = \begin{cases} x^2 - y^2 & \text{if } x^2 + 4y^2 \le 1, \\ \sin(x)\cos(y) & \text{if } x^2 + 4y^2 > 1. \end{cases} \tag{6.3}$$

Unlike Example 6.1, the solution in this example is *discontinuous* and independent of the coefficient β. Table 6.3 shows the results of the grid refinement analysis using the second order method. Again we see clearly second order convergence. Figure 6.1(a) plots the solution which is composed of two pieces.

Compared with the original IIM, the advantages of the new methods, especially the second order method, are the following: (a) The SOR method always converges for the new methods since the resulting linear system of equations is diagonally dominant. The system obtained from the original IIM, however, does not have this property. As the result, an iterative method including the SOR and the multigrid DMGD9V either does not converge or has significantly more iterations. (b) When both old and new methods converge, the errors of the solution obtained from the new methods are more evenly distributed; see, for example, Figure 6.1(b).

6.2. Algorithm efficiency analysis. A natural concern about the new methods proposed in this paper is how much extra cost is needed in solving the quadratic optimization problem at each irregular grid point. In Figure 6.2(a), we plot the percentage of the computational time used in the interface treatment versus the ratio of the jump in the coefficients $\log(\beta^+/\beta^-)$. The interface in polar coordinates is

$$r = r(\theta) = \frac{1}{2} + 0.1\,\sin(5\theta), \quad 0 \le \theta \le 2\pi. \tag{6.4}$$

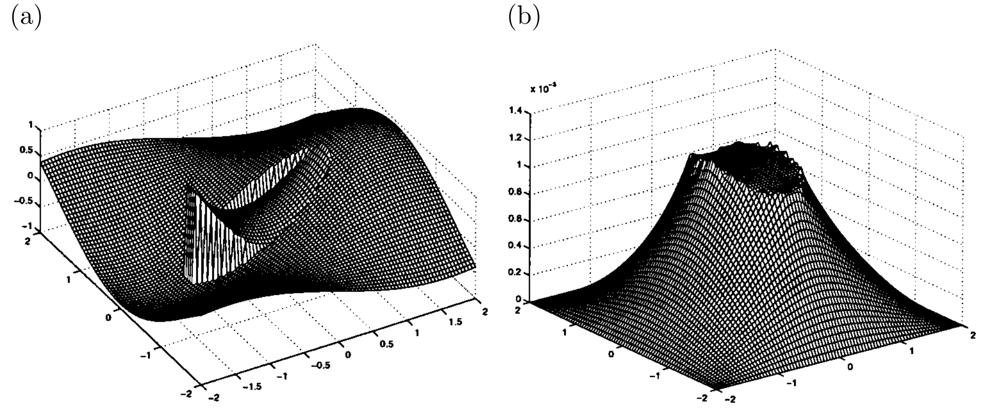


Fig. 6.1. (a) *The solution of Example* 6.2 *with jumps in the solution as well as in the normal derivatives. The parameters are* $\beta^+ = 1$*,* $\beta^- = 100$*, and* $n_{finest} = 82$*.* (b) *The error plot with the same parameters. The error distribution is better than that obtained from the original immersed interface method.*

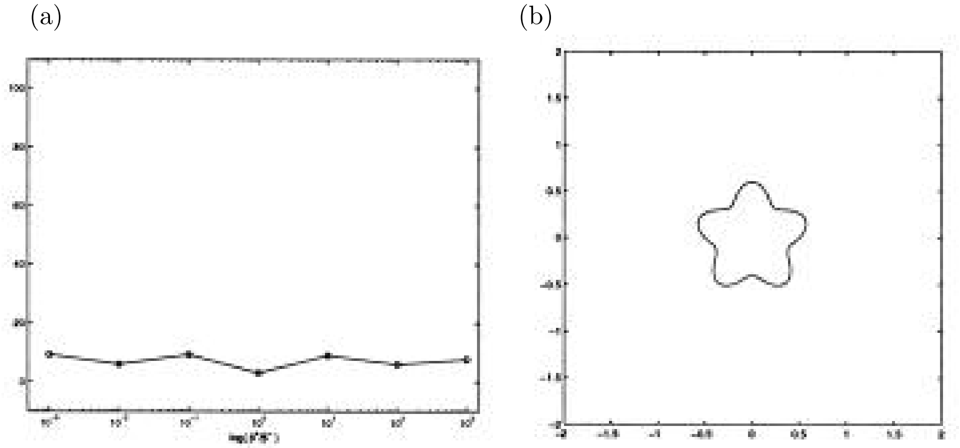


Fig. 6.2. (a) *Plot of percentage of the CPU time used for dealing with interfaces versus* $\log(\beta^+/\beta^-)$*.* (b) *The domain of the test example.*

For this interface, the curvature is quite large; see Figure 6.2(b). The cost for dealing with the irregular grid points includes solving the quadratic optimization problem, interpolating the cubic spline, indexing the irregular grid points, and finding (x_i^*, y_j^*). For regular problems, the multigrid solver is as fast as the fast Poisson solver using FFT.[3] Therefore the multigrid method that we used is indeed a fast solver but it does depend on the jump in β. In all our simulations, the cost for dealing with the irregular grid points near or on the interface is less than 10%. The ratio may increase when a better multigrid solver is used; see [1], for example. When $\beta^- = \beta^+$, the finite difference coefficients become the standard five-point stencil scheme and the cost for the interface treatment reaches its minimum.

The CPU time used in the entire solution process depends on the geometry and the jump in the coefficient β. Table 6.4 lists some statistics for Example 6.2 on an IBM

[3] Generally the fast Poisson solver using FFT can only be used for constant coefficients.

Table 6.4

CPU time for Example 6.2 *with different parameters using an IBM SP*2 *machine. The outputs vary with machines and time.*

n_{finest}	n_1	n_l	n_{coarse}	β^-	β^+	CPU time (s)
130×130	160	5	9	10	1	0.03
258×258	320	6	9	1	1	0.03
258×258	320	6	9	1	100	0.05
258×258	320	6	9	1	10000	0.06
258×258	320	6	9	100	1	0.06
258×258	320	6	9	10000	1	3.29
514×514	640	7	9	1	1000	0.15
514×514	640	7	9	1000	1	0.35

SP2 machine. In this example, when $\beta^- < \beta^+$, the CPU time is just a little more than that needed for one fast Poisson solver. When $\rho = \beta^-/\beta^+ > 1$ gets bigger, we see the CPU time grows slowly. The DMGD9V cannot return a solution in reasonable time when $\rho > 10^6$ with a 258×258 grid; see the remarks below. Generally we see that the second order method converges very fast if DMGD9V does work for the interface problem.

Remark 6.1. The linear system of equations using the finite difference methods proposed in this paper is irreducible and diagonally dominant. The multigrid solver DMGD9V is designed for systems of equations with standard nine-point stencil. The method requires the system of equations to have positive/negative symmetric part and works well for problems with large variation in the coefficients. So it is natural that we choose to use DMGD9V. The multigrid method converges very fast for the system of equations obtained from our algorithms, if it converges. However, we do observe occasionally that the multigrid stops before it returns a convergent result. In these cases, we still can make the multigrid method work by changing some built-in parameters such as the number of maximum iterations on the coarse grid, the number of smoothing operations, etc. Other DMGD9V users[4] have had similar experiences. We are currently working with L. Adams of the University of Washington to develop better multigrid methods for elliptic and parabolic interface problems.

7. Conclusions.

In this paper, we have proposed two finite difference methods that preserve the discrete maximum principles for elliptic interface problems. The methods have been shown to converge provided that the solution to the optimization problem exists and certain bounds are satisfied, which is proved for the first order method and numerically verified for the second order method. The methods can be easily generalized to parabolic interface problems. We would strongly recommend the second order method over the first order method. A Fortran package for a single closed interface and a Dirichlet boundary condition is available upon request (zhilin@math.ncsu.edu).

Appendix. Complete proof of Theorem 5.2.

At an irregular grid point, if (5.7) is true, then we know that

$$L_h\bar{\phi}(x_i, y_j) = \sum_{\xi_k \geq 0} \gamma_k \xi_k + \text{high order terms} \geq \frac{C_2}{h},$$

[4] Private communication with Dr. X. Wu from Caltech and Exxon Company.

where $\bar{\phi}(x,y)$ is defined in (4.15) in section 4. If (5.7) is violated but (5.9) is true, then all the grid points from the other side of the interface other than (x_i, y_j) are very close to the interface. For the nine-point stencil, this can happen only when the interface is very close to one of the grid lines, $x = x_{i-1}$, or $x = x_{i+1}$, or $y = y_{j-1}$, or $y = y_{j+1}$. Without loss of generality, we assume that the interface cuts the grid line $y = y_j$ at x_i^*, where $x_i < x_i^* \le x_{i+1}$, $x_{i+1} - x_i^* \le C_5 h^{1+\sigma}$. The normal direction is nearly parallel to the x-axis. Since $\max_{\xi_k \ge 0} \{\xi_k\} \le C_5 h^{1+\sigma}$, we know that (x_{i+1}, y_j), (x_{i+1}, y_{j-1}), and (x_{i+1}, y_{j+1}) are only three grid points in the nine-point stencil that are on the different side from (x_i, y_j). Since there is no interface involved in the y-direction, we can decompose the equations (3.6) in the x- and y-directions. In the y-direction, assuming (x_i, y_j) is on the $-$ side, we have

$$\frac{\beta_{i,j-\frac{1}{2}} u(x_i, y_{j-1}) - (\beta_{i,j-\frac{1}{2}} + \ \beta_{i,j+\frac{1}{2}}) u(x_i, y_j) + \beta_{i,j+\frac{1}{2}} u(x_i, y_{j+1})}{h^2} = (\beta u_y)_y + O(h^2),$$

which is the standard three-point difference scheme.

The equations in the x-direction are

$$\tilde{a}_1 + \tilde{a}_2 = \beta_x^-, \qquad \tilde{a}_3 + \rho \tilde{a}_4 = 0, \qquad \tilde{a}_7 + \tilde{a}_8\, \rho = \beta^-,$$

where we use $\tilde{a}_i$ to represent those terms in x-directions in a_i's in (3.6). We define a function

$$\text{(A.1)} \qquad \psi_{ij}^x(x) = \begin{cases} \dfrac{(x_i^* - x_i)(x_i - x)}{\gamma_{ij}^{(2)} h^3} & \text{if } x \le x_i^*, \\ \dfrac{(x_{i+1} - x_i^*)(x - x_{i+1})}{\gamma_{i+1,j}^{(8)} h^3} & \text{if } x > x_i^*, \end{cases}$$

where $\gamma_{ij}^{(2)}$ is the coefficient of the finite difference scheme centered at (x_i, y_j) corresponding to the grid point (x_{i-1}, y_j); $\gamma_{i+1,j}^{(8)}$ is the coefficient of the finite difference scheme centered at (x_{i+1}, y_j) corresponding to the grid point (x_{i+2}, y_j). Notice that $\psi_{ij}^x(x_i) = \psi_{ij}^x(x_{i+1}) = 0$ and $\psi_{ij}^x(x_j) > 0$ for $j \ne i,\ i+1$ and that $\psi_{ij}^x(x) \le M\ \forall\, (x,y) \in \Omega$. More important, we can easily derive

$$\text{(A.2)} \qquad L_h \psi_{ij}^x(x_i, y_j) = \frac{x_i^* - x_i}{h^2} (1 + O(C_6 h^\sigma)),$$

$$\text{(A.3)} \qquad L_h \psi_{ij}^x(x_{i+1}, y_j) = \frac{x_{i+1} - x_i^*}{h^2} (1 + O(C_7 h^\sigma)).$$

The high order terms are due to the fact that L_h may not be exactly the standard five-point finite difference scheme, although it is close. Therefore when all $\xi_k \ge 0$ is very small, $x_i^* - x_i$ is close to h and $L_h \psi_{ij}^x(x_i, y_j) \ge \frac{1 - C_8 h^\sigma}{h}$.

Similarly we can construct such $\psi_{ij}^x(x)$ or $\psi_{ij}^y(y)$ at the few irregular grid points where (5.9) holds. Define the comparison function as

$$\tilde{\phi}(x,y) = \bar{\phi}(x,y) + \sum \psi_{ij}^x(x) + \sum \psi_{ij}^y(y),$$

where $\bar{\phi}$ is the same as defined in (4.15). Then

$$L_h\tilde{\phi}(x_i,y_j) \geq \begin{cases} 1+O(h^2) & \text{if } (x_i,y_j) \text{ is a regular grid point,} \\ \displaystyle\sum_{\xi_k\geq 0} \gamma_k\xi_k \geq \frac{C_2}{h}+O(1) & \text{if (5.7) is true,} \\ \dfrac{1-C_8h^\sigma}{h} & \text{if } \displaystyle\max_{\xi_k\geq 0}\{\xi_k\} \leq C_5h^{1+\sigma}, \text{ with } \sigma>0. \end{cases}$$

Note that at some regular grid points, $L_h\tilde{\phi}(x_i,y_j)$ can be very large but it is nonnegative. Thus, the first inequality above still holds. Therefore at a regular point we have

$$\frac{|T_{ij}|}{L_h\tilde{\phi}(x_i,y_j)} \leq \frac{C_3h^2}{1}.$$

At an irregular grid point where (5.7) is satisfied, we have

$$\frac{|T_{ij}|}{L_h\tilde{\phi}(x_i,y_j)} \leq \frac{C_4h}{C_2/h} = \frac{C_4}{C_2}h^2$$

since the truncation error is bounded by

$$|T_{ij}| \leq C_4h$$

for some constant C_4 that depends on the third derivatives of the solution on each side of the interface. At an irregular grid point where (5.9) is true, we also have

$$\frac{|T_{ij}|}{L_h\tilde{\phi}(x_i,y_j)} \leq \frac{C_4h}{(1-C_8h^\sigma)/h} = \frac{C_4}{1-C_8h^\sigma}h^2.$$

Thus from Lemma 4.3, we have again proved the quadratic convergence.

Acknowledgments. We would like to thank K. Schittkowski for his quadratic optimization code and P. de Zeeuw for his multigrid solver. We are also grateful for the discussion with them and L. Adams and R. LeVeque.

REFERENCES

[1] L. M. Adams, *A multigrid algorithm for immersed interface problems*, in Proceedings of Copper Mountain Multigrid Conference, Copper Mountain, CO, NASA Conference Publication 3339, 1995, pp. 1–14.

[2] I. Babuška, *The finite element method for elliptic equations with discontinuous coefficients*, Computing, 5 (1970), pp. 207–213.

[3] Z. Chen and J. Zou, *Finite element methods and their convergence for elliptic and parabolic interface problems*, Numer. Math., 79 (1998), pp. 175–202.

[4] D. De Zeeuw, *Matrix-dependent prolongations and restrictions in a blackbox multigrid solver*, J. Comput. Appl. Math., 33 (1990), pp. 1–27.

[5] A. L. Fogelson and J. P. Keener, *Immersed interface methods for Neumann and related problems in two and three dimensions*, SIAM J. Sci. Comput, 22 (2000), pp. 1630–1654.

[6] H. Huang and Z. Li, *Convergence analysis of the immersed interface method*, IMA J. Numer. Anal., 19 (1999), pp. 583–608.

[7] R. J. LeVeque and Z. L. Li, *The immersed interface method for elliptic equations with discontinuous coefficients and singular sources*, SIAM J. Numer. Anal., 31 (1994), pp. 1019–1044.

[8] Z. Li, *The Immersed Interface Method—A Numerical Approach for Partial Differential Equations with Interfaces*, Ph.D. thesis, University of Washington, Seattle, WA, 1994.

[9] Z. Li, *A fast iterative algorithm for elliptic interface problems*, SIAM J. Numer. Anal., 35 (1998), pp. 230–254.

[10] Z. Li, T. Lin, and X. Wu, *New Cartesian Grid Methods for Interface Problem Using Finite Element Formulation*, Tech. report NCSU CRSC-TR99-5, North Carolina State University, Raleigh, NC, 1999.

[11] X. Liu, R. Fedkiw, and M. Kang, *A boundary condition capturing method for Poisson's equation on irregular domain*, J. Comput. Phys., 160 (2000), pp. 151–178.

[12] K. W. Morton and D. F. Mayers, *Numerical Solution of Partial Differential Equations*, Cambridge Press, Cambridge, UK, 1995.

[13] K. Schittkowski, *QL-quadratic Programming, Version* 1.5, 1991; available from http://www.uni-bayreuth.de/departments/math/~kschittkowski/ql.htm.

SPECIAL ISSUE: 2000 COPPER MOUNTAIN CONFERENCE

The Copper Mountain Conferences, organized annually by the University of Colorado in cooperation with SIAM, focus alternately on multigrid methods and iterative methods. The conferences have built a reputation for the relaxed atmosphere that makes it *the* place for young researchers to present their ideas and for the older celebrities to listen to so many exciting, often still unpolished, ideas. This development of young researchers is further stimulated by the Student Paper Competition, through which the best work of doctoral students is highlighted.

On April 3–7, 2000, we saw the sixth conference in the series on iterative methods, and the research area was as fresh, lively, and inspiring as ever. The Program Committee quickly agreed that it would be very attractive for the entire research community to have the best contributions collected as papers in a special issue of *SIAM Journal on Scientific Computing*. We are very grateful to SIAM for graciously accepting our proposal. SIAM handed the responsibility for this special issue to the Program Committee, and those present at the meeting accepted the invitation to act as guest associate editors. The next step was to invite all speakers to submit a paper, which led to 31 submissions. All papers underwent the usual SIAM review process, maintaining the high SIAM standards for acceptance. Eventually, 18 papers were accepted for publication, among them the top 2 in the Student Paper Competition, by Judith Vogel (with Daniel Szyld) and Michiel Hochstenbach.

A special thanks should go to all those anonymous referees who helped us meet the deadlines through their very constructive efforts. I consider this a strong sign that we are members of a friendly research family rather than individuals in an impersonal community.

The 18 accepted papers are distributed over the following categories:

- 6 papers related to preconditioning,
- 6 related to eigenvalue computation,
- 3 on multigrid techniques,
- 2 on nonlinear problems,
- and 1 with a new variant of an iterative method (TQMR).

This shows that preconditioning continues to attract much research attention, with many more challenging problems left open rather than being solved. Eigenvalue computation continues to gain more attention after initial emphasis in these conferences on linear systems.

We expect that the readers of these papers will enjoy reading them as much as we did and that it will help many of us make further progress in research. We thank all the authors for their efforts to get their papers ready under often very tight time constraints. I thank my colleagues the associate editors for their dedication and help. Above all, we are indebted to Tom Manteuffel and Steve McCormick for organizing such excellent conferences.

Henk van der Vorst
guest editor

Guest associate editors:

Iain Duff	Tim Kelley	Nick Trefethen	Olof Widlund
Howard Elman	Seymour Parter	Panayot Vassilevski	
Roland Freund	Gerhard Starke	Homer Walker	

SIAM J. SCI. COMPUT.
Vol. 23, No. 2, pp. 363–380

FQMR: A FLEXIBLE QUASI-MINIMAL RESIDUAL METHOD WITH INEXACT PRECONDITIONING*

DANIEL B. SZYLD† AND JUDITH A. VOGEL‡

Abstract. A flexible version of the QMR algorithm is presented which allows for the use of a different preconditioner at each step of the algorithm. In particular, inexact solutions of the preconditioned equations are allowed, as well as the use of an (inner) iterative method as a preconditioner. Several theorems are presented relating the norm of the residual of the new method with the norm of the residual of other methods, including QMR and flexible GMRES (FGMRES). In addition, numerical experiments are presented which illustrate the convergence of flexible QMR (FQMR), and show that in certain cases FQMR can produce approximations with lower residual norms than QMR.

Key words. Krylov subspace methods, flexible preconditioning, inner-outer iterations

AMS subject classification. 65F10

PII. S106482750037336X

1. Introduction.

The quasi-minimal residual (QMR) method [11] is a well-established Krylov subspace method for solving large systems of non-Hermitian linear equations of the form

$$A\mathbf{x} = \mathbf{b}, \tag{1}$$

where the $n \times n$ matrix A is assumed to be nonsingular. The algorithm makes use of a three-term recurrence, and thus, unlike some other Krylov methods for non-Hermitian systems, e.g., GMRES [23], storage requirements are fixed and known a priori.

The strength of Krylov subspace methods is most apparent when a preconditioner is used. For a general introduction to these methods see, e.g., [17], [22]. In the case of right preconditioning, one solves the equivalent linear system

$$AM^{-1}(M\mathbf{x}) = \mathbf{b}$$

with some appropriate preconditioner M. The matrix AM^{-1} is never formed explicitly. Instead, when $\mathbf{z} = M^{-1}\mathbf{v}$ is required, one solves the corresponding system

$$M\mathbf{z} = \mathbf{v}. \tag{2}$$

In an analogous manner, left preconditioning consists of solving $M^{-1}A\mathbf{x} = M^{-1}\mathbf{b}$.

In this paper we present a flexible version of QMR, where the matrix M in (2) can vary from one iteration to the next. Let us denote by M_i the preconditioner used at the ith iteration. The need to allow for a variable preconditioner arises, e.g., when the solution of (2) is not obtained exactly (say, by a direct method), but is approximated by the use of a second (inner) iterative method. This is the case, e.g., when the preconditioner used is some version of multigrid, such as in [10].

*Received by the editors June 7, 2000; accepted for publication (in revised form) December 21, 2000; published electronically July 10, 2001.
http://www.siam.org/journals/sisc/23-2/37336.html

†Department of Mathematics, Temple University (038-16), 1805 N. Broad Street, Philadelphia, PA 19122-6094 (szyld@math.temple.edu). The work of this author was supported by the National Science Foundation grant DMS-9973219.

‡Richard Stockton College, Pomona, NJ 08240 (judith.vogel@stockton.edu).

In recent years, several authors worked on the idea of preconditioning with a different matrix at each outer iteration of a Krylov subspace method [1], [16], [20], [21], [25]; see also [6], [14], [15] for other instances of inner-outer iterations. Preconditioning of this form is referred to as flexible preconditioning, also known as variable or inexact preconditioning. Our approach to a flexible version of QMR, which we call FQMR, is similar to that of Saad for FGMRES [21]. In addition, we were influenced by ideas presented by Golub and Ye for inexact conjugate gradients [16].

The new FQMR method, in the same way as the other inexact methods just mentioned, is not strictly speaking a Krylov subspace method. This is because the minimization at each step is done over a subspace which is not in general a Krylov subspace; also cf. [7]. Nevertheless, the convergence theory developed by Eiermann and Ernst [9] applies to these methods as well.

We emphasize that FQMR can be used whenever FGMRES is used, with the advantage that FQMR has a fixed low memory requirement.

FQMR is not intended as an alternative to QMR when the latter works well but rather as an option when no fixed preconditioner is available, as in [10] and in [18], or when the preconditioner can be improved from one step to the next with newly available information; cf. [2]. Nevertheless, we have found in our numerical experiments that FQMR can produce approximations with lower residual norms than QMR.

In the next section we briefly review the QMR method and present the new flexible version. We describe several properties of this new version, including the quasi-minimization property over a certain subspace. In section 3 we present our main theorem relating the residual norm of FQMR with that of FGMRES, in a way analogous to the well-known relation between QMR and GMRES. As a corollary we obtain a new relation between the residual norm of FGMRES and that of QMR.

The same techniques are used in section 4 to obtain bounds for the FQMR residual norm. As is to be expected, these bounds are in terms of how inexactly each preconditioned step is solved. In a similar way new bounds for the FGMRES residual norm are obtained.

Finally, in section 5 numerical experiments are reported which describe the convergence behavior of FQMR using several different iterative methods as flexible preconditioners. Furthermore, a comparison is made between FQMR and the corresponding version of QMR with fixed preconditioner. This comparison shows that in many cases the maximal attainable accuracy of FQMR, i.e., the minimum relative residual norm, is better than that of QMR.

2. The FQMR algorithm.

We begin this section by first considering the standard QMR algorithm with a fixed right preconditioner M; for full details see [11], [17], or [22]. An analogous description can be given for QMR with fixed left preconditioning. Let $\mathbf{x}_0$ be the initial guess and let $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$ be the initial residual. In QMR, the two-sided Lanczos algorithm is used to construct biorthogonal bases corresponding to the Krylov subspaces generated by AM^{-1} and $(AM^{-1})^H$, namely,

$$K_m(AM^{-1}, \mathbf{r}_0) = \text{span}\{\mathbf{r}_0, AM^{-1}\mathbf{r}_0, \ldots, (AM^{-1})^{m-1}\mathbf{r}_0\}$$

and

$$K_m((AM^{-1})^H, \mathbf{r}_0) = \text{span}\{\mathbf{r}_0, M^{-H}A^H\mathbf{r}_0, \ldots, (M^{-H}A^H)^{m-1}\mathbf{r}_0\}.$$

Let the basis vectors for $K_m(AM^{-1}, \mathbf{r}_0)$ and $K_m(M^{-H}A^H, \mathbf{r}_0)$ be $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m\}$ and $\{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_m\}$, the columns of V_m and W_m, respectively. The approximation to the solution of (1) at the mth QMR iteration is of the form

$$\mathbf{x}_m = \mathbf{x}_0 + M^{-1}V_m\mathbf{y}_m, \quad \mathbf{y}_m = \arg\min_{\mathbf{y}} \| \|\mathbf{r}_0\|\mathbf{e}_1 - T_{m+1,m}\mathbf{y}\|, \tag{3}$$

where $T_{m+1,m}$ is the $(m+1)\times m$ tridiagonal matrix of recurrence coefficients associated with two-sided Lanczos, namely,

$$T_{m+1,m} = \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \gamma_1 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_{m-1} \\ & & \gamma_{m-1} & \alpha_m \\ & & & \gamma_m \end{bmatrix};$$

see Algorithm 2.1 below. In (3) and throughout the paper the norm is the 2-norm.

By the construction of the two-sided Lanczos, the following relation holds:

$$AM^{-1}V_m = V_{m+1}T_{m+1,m}, \tag{4}$$

from which it follows that

$$\begin{aligned} \mathbf{r}_m &= \mathbf{b} - A\mathbf{x}_m \\ &= \mathbf{r}_0 - AM^{-1}V_m\mathbf{y}_m \qquad (5)\\ &= V_{m+1}(\|\mathbf{r}_0\|\mathbf{e}_1 - T_{m+1,m}\mathbf{y}_m). \qquad (6) \end{aligned}$$

This establishes that QMR produces the approximation $\mathbf{x}_m$ in such a way as to minimize the norm of the second factor of the residual (6); see (3). Thus, a quasi-minimization of the residual norm takes place. One can see from (5) that the residual at the mth step is of the form $\mathbf{r}_0 - AM^{-1}\mathbf{v}$ with $\mathbf{v} \in K_m(AM^{-1}, \mathbf{r}_0)$. Relation (4) can be rewritten

$$AZ_m^Q = V_{m+1}^Q T_{m+1,m}, \tag{7}$$

where $Z_m^Q = M^{-1}V_m$. We use superscripts Q and FQ to distinguish the vectors and matrices generated by QMR and FQMR, respectively. Correspondingly, rewriting (3), the approximate solution is of the form

$$\mathbf{x}_m = \mathbf{x}_0 + Z_m^Q\mathbf{y}_m, \quad \mathbf{y}_m = \arg\min_{\mathbf{y}} \| \|\mathbf{r}_0\|\mathbf{e}_1 - T_{m+1,m}\mathbf{y}\|.$$

Relation (4) illustrates that the action of AM^{-1} on a vector $\mathbf{v}$ of the Krylov subspace is in $K_{m+1}(AM^{-1}, \mathbf{r}_0)$ a basis of which are the columns of V_{m+1}^Q, while relation (7) will be useful in comparing QMR with FQMR. With this background in place, we present the following algorithm for FQMR.

Algorithm 2.1 (FQMR).

```
    Given $\mathbf{x}_0$, form $\mathbf{r}_0$ and choose $\hat{\mathbf{r}}_0$ such that $\langle \mathbf{r}_0, \hat{\mathbf{r}}_0 \rangle \neq 0$
    set $\mathbf{v}_1 = \mathbf{r}_0/\|\mathbf{r}_0\|$ and $\mathbf{w}_1 = \hat{\mathbf{r}}_0/\langle \hat{\mathbf{r}}_0, \mathbf{v}_1 \rangle$
    set $\beta_0 = \gamma_0 = 0$ and $\mathbf{v}_0 = \mathbf{w}_0 = 0$
    For $i = 1, 2, \ldots$
(8)     Set $\mathbf{z}_i = M_i^{-1}\mathbf{v}_i$
```

```
(9)     Compute: $A\mathbf{z}_i$ and $M_i^{-H}A^H\mathbf{w}_i$
        $\alpha_i = \langle A\mathbf{z}_i, \mathbf{w}_i\rangle$
(10)    $\tilde{\mathbf{v}}_{i+1} = A\mathbf{z}_i - \alpha_i\mathbf{v}_i - \beta_{i-1}\mathbf{v}_{i-1}$
        $\tilde{\mathbf{w}}_{i+1} = M_i^{-H}A^H\mathbf{w}_i - \bar{\alpha}_i\mathbf{w}_i - \gamma_{i-1}\mathbf{w}_{i-1}$
        $\gamma_i = \|\tilde{\mathbf{v}}_{i+1}\|$
(11)    $\mathbf{v}_{i+1} = \tilde{\mathbf{v}}_{i+1}/\gamma_i$
        $\beta_i = \langle \mathbf{v}_{i+1}, \tilde{\mathbf{w}}_{i+1}\rangle$
        $\mathbf{w}_{i+1} = \tilde{\mathbf{w}}_{i+1}/\bar{\beta}_i$
(12)    $\mathbf{x}_i = \mathbf{x}_0 + Z_i^{FQ}\mathbf{y}_i$,   where $\mathbf{y}_i = \arg\min_{\mathbf{y}} \|\|\mathbf{r}_0\|\mathbf{e}_1 - T_{i+1,i}\mathbf{y}\|$
                                and $Z_i^{FQ} = [\mathbf{z}_1, \ldots, \mathbf{z}_i]$
```

In this algorithm, $\bar{\alpha}_i$ and $\bar{\beta}_i$ stand for the conjugate of the complex numbers α_i and β_i, respectively. Note that the above algorithm represents a theoretical rendition of FQMR and not a full description of its implementation. In the actual implementation of FQMR the minimization in (12) is solved by performing a QR factorization of $T_{i+1,i}$. Using Givens rotations, this factorization of $T_{i+1,i}$ is easily updated from that of $T_{i,i-1}$, thus eliminating the need to save and construct $Z_i^{FQ} = [\mathbf{z}_1, \ldots, \mathbf{z}_i]$ at each step.

If we replace M_i with M, a fixed preconditioner, the above algorithm reduces to the standard QMR method. Thus, implementation of FQMR requires only a slight modification of the code for QMR, and this is one of the strengths of this new algorithm.

Let us discuss the variable preconditioned step (8) in some detail. If the preconditioned equations $M\mathbf{z} = \mathbf{v}$ are solved approximately by a second iterative solver, then we can write (8) in terms of M^{-1} and $\boldsymbol{\varepsilon}_i$, where $\boldsymbol{\varepsilon}_i$ is the error associated with the inner solve at step i. Thus, for this case we write

$$\mathbf{z}_i = M_i^{-1}\mathbf{v}_i = M^{-1}\mathbf{v}_i + \boldsymbol{\varepsilon}_i. \tag{13}$$

This description of flexible preconditioning is used in [16] for investigating inexact conjugate gradient.

A consequence of flexible preconditioning is the relation

$$AZ_m^{FQ} = V_{m+1}^{FQ}T_{m+1,m}, \tag{14}$$

where Z_m^{FQ} is the matrix whose ith column is $\mathbf{z}_i$, the vector constructed by FQMR in (8); cf. (7). Let us denote by $\hat{K}_m$ the subspace spanned by the columns of Z_m^{FQ}, which, in general, is not a Krylov subspace. Consequently, relation (14) cannot be simplified into a form similar to relation (4) since the action AM_i^{-1} on a vector $\mathbf{v}$ of the Krylov subspace is no longer in the span of the columns of V_{m+1}. Using (14), however, we can still display a quasi-minimization property held by $\mathbf{x}_m$ over this new $\hat{K}_m$, and this is why the convergence theory in [9] applies to FQMR. The proof is identical to the one for QMR as we show in what follows. For an arbitrary vector in the space $\mathbf{x}_0 + \hat{K}_m$, i.e., of the form $\mathbf{z} = \mathbf{x}_0 + Z_m^{FQ}\mathbf{y}$, for some $\mathbf{y} \in \mathbb{C}^m$, we have the following identities

$$\begin{aligned}\mathbf{b} - A\mathbf{z} = \mathbf{b} - A(\mathbf{x}_0 + Z_m^{FQ}\mathbf{y}) &= \mathbf{r}_0 - AZ_m^{FQ}\mathbf{y}\\ &= V_{m+1}^{FQ}(\|r_0\|\mathbf{e}_1 - T_{m+1,m}\mathbf{y}).\end{aligned}$$

Now, since $\mathbf{x}_m$ is chosen to minimize the norm of $\|r_0\|\mathbf{e}_1 - T_{m+1,m}\mathbf{y}$, we see that FQMR maintains the quasi-minimal residual property over the affine space $\mathbf{x}_0 + \hat{K}_m$.

We remark that FQMR with left preconditioning does not follow an analogous construction. For flexible left preconditioning, relation (14) no longer holds, and therefore $\mathbf{x}_i$ cannot be updated as in (12).

Another noteworthy observation is that FQMR, by construction, maintains the three-term recurrence of QMR. However, in so doing, there is a loss of the global biorthogonality held by the bases generated by QMR. However, a local biorthogonality property is maintained, as shown in the following theorem. That is, consecutive Lanczos vectors constructed by the flexible two-sided Lanczos process are biorthogonal. We note, however, that for the results of this theorem to hold, we are assuming that at each step of the outer iteration the matrix M_i is the same matrix used in (8) and in (9). Theoretically this is a logical assumption, and in some cases this is also true in practice, but in general this assumption does not necessarily hold.

THEOREM 2.2. *If the two-sided Lanczos vectors are defined at steps* $1, \ldots, k+1$ *in Algorithm* 2.1*, i.e., if* $\langle \mathbf{v}_i, \mathbf{w}_i \rangle \neq 0$ *for* $i = 1, \ldots, k+1$*, then*

$$\langle \mathbf{v}_{k+1}, \mathbf{w}_k \rangle = 0 \qquad \textit{and} \qquad \langle \mathbf{w}_{k+1}, \mathbf{v}_k \rangle = 0. \tag{15}$$

Proof. We first note that $\|\mathbf{v}_i\| = 1$ for all i by the choice of γ_{i-1}. Likewise $\langle \mathbf{v}_i, \mathbf{w}_i \rangle = 1$ for all i by the choice of γ_{i-1} and $\bar{\beta}_{i-1}$. We prove (15) by induction. For $k = 0$ the result is obvious since $\mathbf{v}_0 = \mathbf{w}_0 = 0$. Assume that (15) holds for $i \leq k-1$; then

$$\begin{aligned} \langle \tilde{\mathbf{v}}_{k+1}, \mathbf{w}_k \rangle &= \langle AM_k^{-1}\mathbf{v}_k, \mathbf{w_k} \rangle - \alpha_k \langle \mathbf{v}_k, \mathbf{w}_k \rangle - \beta_{k-1} \langle \mathbf{v}_{k-1}, \mathbf{w}_k \rangle \\ &= \langle AM_k^{-1}\mathbf{v}_k, \mathbf{w_k} \rangle - \alpha_k = 0 \end{aligned}$$

and

$$\begin{aligned} \langle \tilde{\mathbf{w}}_{k+1}, \mathbf{v}_k \rangle &= \langle M_k^{-H} A^H \mathbf{w}_k, \mathbf{v}_k \rangle - \bar{\alpha}_k \\ &= \langle M_k^{-H} A^H \mathbf{w}_k, \mathbf{v}_k \rangle - \overline{\langle AM_k^{-1}\mathbf{v}_k, \mathbf{w}_k \rangle} \\ &= \langle M_k^{-H} A^H \mathbf{w}_k, \mathbf{v}_k \rangle - \overline{\langle \mathbf{v}_k, M_k^{-H} A^H \mathbf{w}_k \rangle} = 0. \qquad \square \end{aligned}$$

3. FQMR and FGMRES.

Generalized minimal residual (GMRES) [23] is another successful Krylov subspace method for solving non-Hermitian systems of linear equations. Like QMR, GMRES can be implemented with a flexible preconditioner, and the flexible version is called FGMRES [21]. In this section we establish a relationship between FQMR and FGMRES which is reminiscent of a relationship held by QMR and GMRES [19]. As a precursor to the statement of this relationship, we give a brief summary of GMRES with a fixed right preconditioner; see [17], [22], or [23] for a full description.

At the mth iteration, preconditioned GMRES produces an approximation $\mathbf{x}_m$ to the solution of (1) in $\mathbf{x}_0 + K_m(AM^{-1}, \mathbf{r}_0)$ in such a way as to minimize the norm of the residual over $K_m(AM^{-1}, \mathbf{r}_0)$, i.e.,

$$\mathbf{x}_m = \mathbf{x}_0 + \arg \min_{\mathbf{z} \in K_m(AM^{-1}, \mathbf{r}_0)} \| \mathbf{r}_0 - AM^{-1}\mathbf{z} \|.$$

This minimization is accomplished by constructing an orthonormal basis for $K_m(AM^{-1}, \mathbf{r}_0)$ using the Arnoldi method [17]. In the Arnoldi method the basis vectors, $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m\}$, are formed recursively by orthogonalizing a new vector, $\hat{\mathbf{v}}_{i+1}$, in $K_{i+1}(AM^{-1}, \|\mathbf{r}_0\|)$ against the previous vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_i$ in $K_i(AM^{-1}, \|\mathbf{r}_0\|)$ and then normalizing it, so that $\mathbf{v}_{i+1} = \hat{\mathbf{v}}_{i+1}/\|\hat{\mathbf{v}}_{i+1}\|$. Thus,

$$\mathbf{v}_{i+1} = \frac{1}{h_{i+1,i}} \left(AM^{-1}\mathbf{v}_i - \sum_{k=1}^{i} h_{k,i}\mathbf{v}_k \right), \tag{16}$$

where the coefficients $h_{k,i},\ k = 1, \ldots, i+1$, are appropriately chosen to orthogonalize and normalize $\mathbf{v}_{i+1}$. From (16) we get the relation

$$AM^{-1}V_m = V_{m+1}H_{m+1,m}, \tag{17}$$

where $H_{m+1,m}$ is the upper-Hessenberg matrix whose nonzero entries are the coefficients $h_{k,i},\ k = 1, \ldots, i+1$, from (16), $i = 1, \ldots, m$.

With this groundwork in place, we can now establish that GMRES computes the approximation $\mathbf{x}_m = \mathbf{x}_0 + V_m\mathbf{y}_m$ such that

$$\begin{aligned} \mathbf{y}_m &= \arg\min_{\mathbf{y}} \|\mathbf{r}_0 - AM^{-1}V_m\mathbf{y}\| = \arg\min_{\mathbf{y}} \|\mathbf{r}_0 - V_{m+1}H_{m+1,m}\mathbf{y}\| \\ &= \arg\min_{\mathbf{y}} \| \|\mathbf{r}_0\|\mathbf{e}_1 - H_{m+1,m}\mathbf{y}\|. \end{aligned} \tag{18}$$

Observe that while in (3) QMR only minimizes the norm of a factor of the residual, in (18) we have that GMRES minimizes the norm of the actual residual, since V_{m+1} is unitary.

We end this introduction of GMRES by pointing out that when the preconditioner changes from step to step, i.e., when one uses M_i instead of M in (16), we have FGMRES [21]. In what follows, we use superscripts G and FG to distinguish the vectors and matrices generated by GMRES and FGMRES, respectively.

Let $\mathbf{z}_i = M_i^{-1}\mathbf{v}_i$, and let Z_m^{FG} be the matrix with columns $\mathbf{z}_1, \ldots, \mathbf{z}_m$; then the following relation analogous to (17) holds for FGMRES:

$$AZ_m^{FG} = V_{m+1}^{FG}H_{m+1,m}. \tag{19}$$

When QMR and GMRES are implemented with fixed preconditioning, we have the following result due to Nachtigal [19], where κ_2 denotes the condition number using the 2-norm; see also [17], [22].

Theorem 3.1. *If $\mathbf{r}_m^G$ denotes the GMRES residual at step m and $\mathbf{r}_m^Q$ denotes the QMR residual at step m, then*

$$\|\mathbf{r}_m^Q\| \le \kappa_2(V_{m+1}^Q)\|\mathbf{r}_m^G\|, \tag{20}$$

where V_{m+1}^Q is the matrix whose columns are the basis vectors generated by QMR.

This inequality works out nicely since V_{m+1}^G constructed by GMRES and V_{m+1}^Q constructed by QMR are both a basis for the same Krylov subspace. When flexible preconditioning is used, this is no longer the case. To avoid confusion, let us denote by $\mathbf{v}_{i+1}^{FQ}$ and $\mathbf{v}_{i+1}^{FG}$ the vectors computed by FQMR in (11) and by FGMRES in (16), respectively. We can, however, prove a relation similar to (20) for FGMRES and FQMR, but to do so we first need the following lemma relating Z_m^{FG} and Z_m^{FQ}, the basis of the subspaces constructed by FGMRES and FQMR, respectively; see (19) and (14).

Lemma 3.2. *Let $\mathbf{z}_i^{FG}$ be the ith column of Z_m^{FG} satisfying* (19). *Likewise, let $\mathbf{z}_i^{FQ}$ be the ith column of Z_m^{FQ} satisfying* (14). *If $\boldsymbol{\varepsilon}_i^{FQ}$ is the ith error vector defined by*

$$\mathbf{z}_i^{FQ} = M^{-1}\mathbf{v}_i^{FQ} + \boldsymbol{\varepsilon}_i^{FQ} \tag{21}$$

and if $\boldsymbol{\varepsilon}_i^{FG}$ is the ith error vector defined by

$$\mathbf{z}_i^{FG} = M^{-1}\mathbf{v}_i^{FG} + \boldsymbol{\varepsilon}_i^{FG}, \tag{22}$$

then

$$\mathbf{z}_i^{FG} \in S^m, \quad i = 1, \dots, m, \tag{23}$$

where

$$S^m = \operatorname{span}\{\mathbf{z}_i^{FQ}, (M^{-1}A)^j\boldsymbol{\varepsilon}_i^{FQ}, (M^{-1}A)^j\boldsymbol{\varepsilon}_i^{FG};\ i = 1, \dots, m,\ j = 0, \dots, m-1\}.$$

Proof. We show (23) by induction on m. For $m = 1$, since $\mathbf{v}_1^{FG} = \mathbf{v}_1^{FQ} = \mathbf{r}_0/\|\mathbf{r}_0\|$, we have $\mathbf{z}_1^{FG} = M^{-1}\mathbf{v}_1^{FG} + \boldsymbol{\varepsilon}_1^{FG} = M^{-1}\mathbf{v}_1^{FQ} + \boldsymbol{\varepsilon}_1^{FG} = \mathbf{z}_1^{FQ} - \boldsymbol{\varepsilon}_1^{FQ} + \boldsymbol{\varepsilon}_1^{FG} \in S^1$. Assuming that (23) holds for $n \le m$, then

$$\begin{aligned}
\mathbf{z}_{m+1}^{FG} &= M^{-1}\mathbf{v}_{m+1}^{FG} + \boldsymbol{\varepsilon}_{m+1}^{FG} \\
&= M^{-1}(\tfrac{1}{h_{m+1,m}})\left(Az_m^{FG} - \sum_{k=1}^{m} h_{k,m}\mathbf{v}_k^{FG}\right) + \boldsymbol{\varepsilon}_{m+1}^{FG} \\
&= (\tfrac{1}{h_{m+1,m}})\left(M^{-1}A\mathbf{z}_m^{FG} - \sum_{k=1}^{m} h_{k,m}M^{-1}\mathbf{v}_k^{FG} + h_{m+1,m}\boldsymbol{\varepsilon}_{m+1}^{FG}\right) \\
&= (\tfrac{1}{h_{m+1,m}})\left(M^{-1}A\mathbf{z}_m^{FG} - \sum_{k=1}^{m}(h_{k,m}\mathbf{z}_k^{FG} - h_{k,m}\boldsymbol{\varepsilon}_k^{FG}) + h_{m+1,m}\boldsymbol{\varepsilon}_{m+1}^{FG}\right),
\end{aligned}$$

where the first and last equalities follow from (22) and the second equality follows from the definition of $\mathbf{v}_{m+1}^{FG}$.

By the induction hypothesis and the observation that $S^r \subseteq S^t$ for $r \le t$, we have that $\mathbf{z}_i^{FG} \in S^{m+1}$ for $i \le m$. By the definition of S^{m+1}, $\boldsymbol{\varepsilon}_i^{FG} \in S^{m+1}$ for $i \le m+1$. Therefore, all that remains to be shown is that the first term $M^{-1}A\mathbf{z}_m^{FG} \in S^{m+1}$. Again by the induction hypothesis, we know that $\mathbf{z}_m^{FG} \in S^m$; hence, there are scalars $a_i, b_{i,j}, c_{i,j},\ i = 1, \dots m,\ j = 0, \dots, m-1$, such that

$$\mathbf{z}_m^{FG} = \sum_{i=1}^{m} a_i\mathbf{z}_i^{FQ} + \sum_{k=0}^{m-1}\left(\sum_{i=1}^{m} b_{i,k}(M^{-1}A)^k\boldsymbol{\varepsilon}_i^{FQ}\right) + \sum_{k=0}^{m-1}\left(\sum_{i=1}^{m} c_{i,k}(M^{-1}A)^k\boldsymbol{\varepsilon}_i^{FG}\right),$$

and therefore

$$\begin{aligned}
M^{-1}A\mathbf{z}_m^{FG} = {} & \sum_{i=1}^{m} a_i(M^{-1}A)\mathbf{z}_i^{FQ} + \sum_{k=0}^{m-1}\left(\sum_{i=1}^{m} b_{i,k}(M^{-1}A)^{k+1}\boldsymbol{\varepsilon}_i^{FQ}\right) \\
& + \sum_{k=0}^{m-1}\left(\sum_{i=1}^{m} c_{i,k}(M^{-1}A)^{k+1}\boldsymbol{\varepsilon}_i^{FG}\right).
\end{aligned}$$

Since $(M^{-1}A)^j\boldsymbol{\varepsilon}_i^{FG} \in S^{m+1}$ and $(M^{-1}A)^j\boldsymbol{\varepsilon}_i^{FQ} \in S^{m+1}$ for $j = 0, \dots, (m+1)-1$, $i = 1, \dots, m+1$ by definition of S^{m+1}, all that remains to be shown is $(M^{-1}A)\mathbf{z}_i^{FQ} \in S^{m+1}$ for $i = 1, \dots, m$. Solving in (10) for $A\mathbf{z}_j$ and multiplying through by M^{-1} gives

$$(M^{-1}A)\mathbf{z}_i^{FQ} = \gamma_i M^{-1}\mathbf{v}_{i+1}^{FQ} + \alpha_i M^{-1}\mathbf{v}_i^{FQ} + \beta_{i-1}M^{-1}\mathbf{v}_{i-1}^{FQ}. \tag{24}$$

Next, solving (21) for $M^{-1}\mathbf{v}_i^{FQ}$ and substituting this into (24) gives

$$(M^{-1}A)\mathbf{z}_i^{FQ} = \gamma_i\mathbf{z}_{i+1}^{FQ} - \gamma_i\boldsymbol{\varepsilon}_{i+1}^{FQ} + \alpha_i\mathbf{z}_i^{FQ} - \alpha_i\boldsymbol{\varepsilon}_i^{FQ} + \beta_{i-1}\mathbf{z}_{i-1}^{FQ} - \beta_{i-1}\boldsymbol{\varepsilon}_{i-1}^{FQ} \in S^{m+1}. \qquad \square \tag{25}$$

We can now prove our main result.

THEOREM 3.3. *Assume that V_{m+1}^{FQ}, the matrix satisfying* (14), *is of full rank. Let $\mathbf{r}_m^{FQ}$ and $\mathbf{r}_m^{FG}$ be the residuals obtained after m steps of the FQMR and FGMRES algorithms, respectively, and let $\mathcal{E}_m^{FQ} = [\boldsymbol{\varepsilon}_1^{FQ}, \ldots, \boldsymbol{\varepsilon}_m^{FQ}]$ and $\mathcal{E}_m^{FG} = [\boldsymbol{\varepsilon}_1^{FG}, \ldots, \boldsymbol{\varepsilon}_m^{FG}]$, where $\boldsymbol{\varepsilon}_i^{FQ}$ and $\boldsymbol{\varepsilon}_i^{FG}$ are the error vectors in* (21) *and* (22), *respectively. Then there exist vectors $\mathbf{y}_k^{FQ}, \mathbf{y}_k^{FG} \in \mathbb{R}^m$, $k = 1, \ldots, m-1$, such that the following holds:*

$$\|\mathbf{r}_m^{FQ}\| \le \kappa_2(V_{m+1}^{FQ}) \left[\|\mathbf{r}_m^{FG}\| + \sum_{k=0}^{m-1} \|A(M^{-1}A)^k\mathcal{E}_m^{FQ}\mathbf{y}_k^{FQ}\| + \sum_{k=0}^{m-1} \|A(M^{-1}A)^k\mathcal{E}_m^{FG}\mathbf{y}_k^{FG}\| \right].$$

Proof. Step 1. Consider the set defined by

$$\mathcal{R} = \{\mathbf{r} : \ \mathbf{r} = V_{m+1}^{FQ}\mathbf{t}; \ \mathbf{t} = \beta\mathbf{e}_1 - T_{m+1,m}\mathbf{y}; \ \mathbf{y} \in \mathbb{C}^m\}.$$

Let $\mathbf{y}_m$ denote the vector $\mathbf{y}$ that minimizes $\|\beta e_1 - T_{m+1,m}\mathbf{y}\|$, and denote by $\mathbf{t}_m = \beta e_1 - T_{m+1,1}\mathbf{y}_m$. Then by definition we have $\mathbf{r}_m^{FQ} = V_{m+1}^{FQ}\mathbf{t_m}$. By hypothesis, V_{m+1}^{FQ} is of full rank. Therefore, there is an $(m+1) \times (m+1)$ nonsingular matrix S such that $W_{m+1} = V_{m+1}^{FQ}S$ is unitary. Then for any element of the set $\mathcal{R}$,

$$\mathbf{r} = W_{m+1}S^{-1}\mathbf{t} \ , \qquad \mathbf{t} = SW_{m+1}^H\mathbf{r},$$

and, in particular, $\mathbf{r}_m^{FQ} = W_{m+1}S^{-1}\mathbf{t}_m$, which implies

$$\|\mathbf{r}_m^{FQ}\| \le \|S^{-1}\|\|\mathbf{t}_m\|. \tag{26}$$

From (3), it follows that the norm $\|\mathbf{t}_m\|$ is the minimum of $\|\beta e_1 - T_{m+1,m}\mathbf{y}\|$ over all vectors $\mathbf{y}$, and therefore

$$\|\mathbf{t}_m\| = \|SW_{m+1}^H\mathbf{r}_m^{FQ}\| \le \|SW_{m+1}^H\mathbf{r}\| \le \|S\|\|\mathbf{r}\| \quad \text{for all } \mathbf{r} \in \mathcal{R}. \tag{27}$$

Step 2. We now consider

$$\mathbf{r}_m^{FG} = \mathbf{r}_0 - AZ_m^{FG}\mathbf{y}_m^{FG}, \qquad \text{where } \mathbf{y}_m^{FG} \text{ minimizes } \|\mathbf{r}_0 - AZ_m^{FG}\mathbf{y}\|. \tag{28}$$

By Lemma 3.2 there exists vectors $\mathbf{y}_z$, $\mathbf{y}_k^{FQ}$, $\mathbf{y}_k^{FG} \in \mathbb{R}^m$; $k = 0, \ldots, m-1$, such that

$$\begin{aligned}
\mathbf{r}_m^{FG} &= \mathbf{r}_0 - AZ_m^{FQ}\mathbf{y}_z - \sum_{k=0}^{m-1} A(M^{-1}A)^k\mathcal{E}_m^{FQ}\mathbf{y}_k^{FQ} - \sum_{k=0}^{m-1} A(M^{-1}A)^k\mathcal{E}_m^{FG}\mathbf{y}_k^{FG} \\
&= \mathbf{r}_0 - V_{m+1}^{FQ}T_{m+1,m}\mathbf{y}_z - \sum_{k=0}^{m-1} A(M^{-1}A)^k\mathcal{E}_m^{FQ}\mathbf{y}_k^{FQ} - \sum_{k=0}^{m-1} A(M^{-1}A)^k\mathcal{E}_m^{FG}\mathbf{y}_k^{FG}.
\end{aligned}$$

By rearrangement of terms,

$$\mathbf{r}_0 - V_{m+1}^{FQ}T_{m+1,m}\mathbf{y}_z = \mathbf{r}_m^{FG} + \sum_{k=0}^{m-1} A(M^{-1}A)^k\mathcal{E}_m^{FQ}\mathbf{y}_k^{FQ} + \sum_{k=0}^{m-1} A(M^{-1}A)^k\mathcal{E}_m^{FG}\mathbf{y}_k^{FG}.$$

Let $\hat{\mathbf{r}} = V^{FQ}_{m+1}(\|\mathbf{r}_0\|e_1 - T_{m+1,m}\mathbf{y}_z)$; then

$$\begin{aligned} \|\hat{\mathbf{r}}\| &= \|V^{FQ}_{m+1}(\|\mathbf{r}_0\|\mathbf{e}_1 - T_{m+1,m}\mathbf{y}_z)\| = \|\mathbf{r}_0 - V^{FQ}_{m+1}T_{m+1,m}\mathbf{y}_z\| \\ &\le \|\mathbf{r}^{FG}_m\| + \sum_{k=0}^{m-1}\|A(M^{-1}A)^k\mathcal{E}^{FQ}_m\mathbf{y}^{FQ}_k\| + \sum_{k=0}^{m-1}\|A(M^{-1}A)^k\mathcal{E}^{FG}_m\mathbf{y}^{FG}_k\|. \end{aligned} \tag{29}$$

Since $\hat{\mathbf{r}} \in \mathcal{R}$, by (27),

$$\|\mathbf{t}_m\| \le \|S\|\|\hat{\mathbf{r}}\|. \tag{30}$$

Hence, by (26), (29), and (27),

$$\begin{aligned} \|\mathbf{r}^{FQ}_m\| &\le \|S^{-1}\|\|\mathbf{t}_m\| \\ &\le \|S^{-1}\|\|S\|\left[\|\mathbf{r}^{FG}_m\| + \sum_{k=0}^{m-1}\|A(M^{-1}A)^k\mathcal{E}^{FQ}_m\mathbf{y}^{FQ}_k\| + \sum_{k=0}^{m-1}\|A(M^{-1}A)^k\mathcal{E}^{FG}_m\mathbf{y}^{FG}_k\|\right] \end{aligned}$$

and since $\kappa_2(V^{FQ}_{m+1}) = \kappa_2(S) = \|S^{-1}\|\|S\|$, the theorem follows. □

We comment that as $\boldsymbol{\varepsilon}_i$ goes to $\mathbf{0}$, the bound on $\|\mathbf{r}^{FQ}_m\|$ in Theorem 3.3 is reduced. However, we cannot know how sharp this bound is due to the existence of vectors $\mathbf{y}^{FQ}_k, \mathbf{y}^{FG}_k$ in the right-hand side of the inequality.

By considering exact solutions of the preconditioned equations (13), i.e., if $\boldsymbol{\varepsilon}_i = 0$ in (13), or equivalently if $M_i = M$ for all i, then FQMR and FGMRES are reduced to QMR and GMRES with fixed preconditioners and Theorem 3.3 reduces to Theorem 3.1. There are two other special situations, which we want to highlight. If $\mathcal{E}^{FG}_m = 0$, i.e., $\boldsymbol{\varepsilon}^{FQ}_i = 0$ for all i, then FGMRES reduces to GMRES, and we have the following corollary.

Corollary 3.4. *Let V^{FQ}_{m+1}, $\mathcal{E}^{FQ}_m$, and $\mathbf{r}^{FQ}_m$ be as described in Theorem* 3.3*, and let $\mathbf{r}^G_m$ be the residual obtained after m steps of the GMRES algorithm. Then there exists vectors $\mathbf{y}^{FQ}_k \in \mathbb{R}^m$, $k = 1, \ldots, m-1$, such that the following holds:*

$$\|\mathbf{r}^{FQ}_m\| \le \kappa_2(V^{FQ}_{m+1})\left(\|\mathbf{r}^G_m\| + \sum_{k=0}^{m-1}\|A(M^{-1}A)^k\mathcal{E}^{FQ}_m\mathbf{y}^{FQ}_k\|\right).$$

Likewise, if $\mathcal{E}^{FQ}_m = 0$, i.e., $\boldsymbol{\varepsilon}^{FQ}_i = 0$ for all i, then the following corollary, which is a new result for FGMRES, is also established.

Corollary 3.5. *Let V^Q_{m+1}, $\mathcal{E}^{FG}_m$, and $\mathbf{r}^{FG}_m$ be as described in Theorems* 3.1 *and* 3.3*, and let $\mathbf{r}^Q_m$ be the residual obtained after m steps of the QMR algorithm. Then there exists vectors $\mathbf{y}^{FG}_k \in \mathbb{R}^m$, $k = 1, \ldots, m-1$, such that the following holds:*

$$\|\mathbf{r}^Q_m\| \le \kappa_2(V^Q_{m+1})\left(\|\mathbf{r}^{FG}_m\| + \sum_{k=0}^{m-1}\|A(M^{-1}A)^k\mathcal{E}^{FG}_m\mathbf{y}^{FG}_k\|\right).$$

We end this section with a comment on the hypothesis in Theorem 3.3 that V^{FQ}_{m+1} be of full rank. This implies that the subspace $\hat{K}_m$ has dimension m, i.e., that at each step we are minimizing over increasingly larger subspaces. Note that this hypothesis implies that the subspaces $\hat{K}_m$ are nested, and this is precisely the assumption made in [9] for the convergence proofs.

4. Bounds on residual norms. Using the same techniques used in Lemma 3.2 and Theorem 3.3, we provide bounds on the norm of the residual of FQMR in terms of the norm of the residual of QMR. The following lemma relates Z_m^{FQ} to Z_m^Q, the matrices whose columns are the basis of the subspaces generated by FQMR and QMR, respectively; see (14) and (7). We first establish an auxiliary lemma.

LEMMA 4.1. *Let* $\mathbf{z}_i^{FQ}$ *be the ith column of* Z_m^{FQ}*, and let* $\mathbf{z}_i^Q$ *be the ith column of* Z_m^Q*. If* $\boldsymbol{\varepsilon}_i^{FQ}$ *is the ith error vector defined by* (21)*, then*

$$\mathbf{z}_i^Q \in S^m, \quad i = 1, \ldots, m, \tag{31}$$

where $S^m = \operatorname{span}\{\mathbf{z}_i^{FQ}, (M^{-1}A)^j \boldsymbol{\varepsilon}_i^{FQ};\ i = 1, \ldots, m,\ j = 0, \ldots, m-1\}$.

Proof. We show (31) by induction on m. For $m = 1$, we have $\mathbf{z}_1^Q = M^{-1}\mathbf{v}_1^Q = M^{-1}\mathbf{v}_1^{FQ} = \mathbf{z}_1^{FQ} - \boldsymbol{\varepsilon}_1^{FQ} \in S^1$. Assuming that (31) holds for $n \leq m$, then

$$\begin{aligned}
\mathbf{z}_{m+1}^Q &= M^{-1}\mathbf{v}_{m+1}^Q \\
&= M^{-1}(\tfrac{1}{\gamma_m})(A\mathbf{z}_m^Q - \alpha_m \mathbf{v}_m^Q - \beta_{m-1}\mathbf{v}_{m-1}^Q) \\
&= (\tfrac{1}{\gamma_m})(M^{-1}A\mathbf{z}_m^Q - \alpha_m M^{-1}\mathbf{v}_m^Q - \beta_{m-1}M^{-1}\mathbf{v}_{m-1}^Q) \\
&= (\tfrac{1}{\gamma_m})(M^{-1}A\mathbf{z}_m^Q - \alpha_m \mathbf{z}_m^Q - \beta_{m-1}\mathbf{z}_{m-1}^Q),
\end{aligned}$$

where the first and last equalities follow from the relation (2), and the second equality follows from the definition of $\mathbf{v}_{m+1}^Q$; see (10). By the induction hypothesis and the observation that $S^r \subseteq S^t$ for $r \leq t$, we have that $\mathbf{z}_i^Q \in S^{m+1}$ for $i \leq m$. Therefore, all that remains to be shown is that $M^{-1}A\mathbf{z}_m^Q \in S^{m+1}$. Again by the induction hypothesis, we know that $\mathbf{z}_m^Q \in S^m$; hence, there are scalars $a_i, b_{i,j},\ i = 1, \ldots m,\ j = 0, \ldots, m-1$, such that

$$\mathbf{z}_m^Q = \sum_{i=1}^m a_i \mathbf{z}_i^{FQ} + \sum_{k=0}^{m-1} \left(\sum_{i=1}^m b_{i,k} (M^{-1}A)^k \boldsymbol{\varepsilon}_i^{FQ} \right),$$

and therefore

$$M^{-1}A\mathbf{z}_m^Q = \sum_{i=1}^m a_i (M^{-1}A)\mathbf{z}_i^{FQ} + \sum_{k=0}^{m-1} \left(\sum_{i=1}^m b_{i,k} (M^{-1}A)^{k+1} \boldsymbol{\varepsilon}_i^{FQ} \right).$$

Since $(M^{-1}A)^j \boldsymbol{\varepsilon}_i^{FQ} \in S^{m+1}$ for $j = 0, \ldots, (m+1)-1,\ i = 1, \ldots, m+1$, all that remains to show is $(M^{-1}A)\mathbf{z}_i^{FQ} \in S^{m+1}$ for $i = 1, \ldots, m$, but this follows from (25). □

Using Lemma 4.1, we now present the following theorem.

THEOREM 4.2. *Assume that* V_{m+1}^{FQ}*, the matrix satisfying* (14)*, is of full rank. Let* $\mathbf{r}_m^{FQ}$ *and* $\mathbf{r}_m^Q$ *be the residuals obtained after m steps of the FQMR and QMR algorithms, respectively, and let* $\mathcal{E}_m^{FQ} = [\boldsymbol{\varepsilon}_1^{FQ}, \ldots, \boldsymbol{\varepsilon}_m^{FQ}]$*, where* $\boldsymbol{\varepsilon}_i^{FQ}$ *are the error vectors defined by* (21)*. Then there exists vectors* $\mathbf{y}_k^{FQ} \in \mathbb{R}^m$*,* $k = 1, \ldots, m-1$*, such that the following holds:*

$$\|\mathbf{r}_m^{FQ}\| \leq \kappa_2(V_{m+1}^{FQ}) \left(\|\mathbf{r}_m^Q\| + \sum_{k=0}^{m-1} \|A(M^{-1}A)^k \mathcal{E}_m^{FQ} \mathbf{y}_k^{FQ}\| \right).$$

Proof. Step 1 of the proof is identical to Step 1 of the proof of Theorem 3.3.

Step 2. Consider

$$\mathbf{r}_m^Q = \mathbf{r}_0 - AZ_m^Q \mathbf{y}_m^Q, \qquad \text{where } \mathbf{y}_m^Q \text{ minimizes } \|\mathbf{r}_0 - AZ_m^Q \mathbf{y}\|. \tag{32}$$

By Lemma 4.1 there exists vectors $\mathbf{y}_z$, $\mathbf{y}_k^{FQ}$, $k = 0, \dots, m-1$, such that

$$\begin{aligned}\mathbf{r}_m^Q &= \mathbf{r}_0 - AZ_m^{FQ}\mathbf{y}_z - \sum_{k=0}^{m-1} A(M^{-1}A)^k \mathcal{E}_m^{FQ}\mathbf{y}_k^{FQ} \\ &= \mathbf{r}_0 - V_{m+1}^{FQ} T_{m+1,m}\mathbf{y}_z - \sum_{k=0}^{m-1} A(M^{-1}A)^k \mathcal{E}_m^{FQ}\mathbf{y}_k^{FQ}.\end{aligned}$$

By rearrangement of terms,

$$\mathbf{r}_0 - V_{m+1}^{FQ} T_{m+1,m}\mathbf{y}_z = \mathbf{r}_m^Q + \sum_{k=0}^{m-1} A(M^{-1}A)^k \mathcal{E}_m^{FQ}\mathbf{y}_k^{FQ}.$$

Let $\hat{\mathbf{r}} = V_{m+1}^{FQ}(\|\mathbf{r}_0\|\mathbf{e}_1 - T_{m+1,m}\mathbf{y}_z) = \mathbf{r}_0 - V_{m+1}^{FQ}T_{m+1,m}\mathbf{y}_z$; then

$$\|\hat{\mathbf{r}}\| \le \|\mathbf{r}_m^Q\| + \sum_{k=0}^{m-1} \|A(M^{-1}A)^k \mathcal{E}_m^{FQ}\mathbf{y}_k^{FQ}\|.$$

Since $\hat{\mathbf{r}} \in \mathcal{R}$, by (27)

$$\|\mathbf{t}_m\| \le \|S\|\|\hat{\mathbf{r}}\|. \tag{33}$$

Hence, by (26) and (33) we obtain

$$\|\mathbf{r}_m^{FQ}\| \le \|S^{-1}\|\|\mathbf{t}_m\| \le \|S^{-1}\|\|S\| \left(\|\mathbf{r}_m^Q\| + \sum_{k=0}^{m-1} \|A(M^{-1}A)^k \mathcal{E}_m^{FQ}\mathbf{y}_k^{FQ}\| \right),$$

and since $\kappa_2(V_{m+1}^{FQ}) = \kappa_2(S) = \|S^{-1}\|\|S\|$, the theorem follows. □

Using identical techniques as in Lemma 4.1 and Theorem 4.2 one can prove the following new result relating the norm of the residuals associated with GMRES and FGMRES.

THEOREM 4.3. *Assume that V_{m+1}^{FG}, the matrix satisfying* (19), *is of full rank. Let $\mathbf{r}_m^{FG}$ and $\mathbf{r}_m^G$ be the residuals obtained after m steps of the FGMRES and GMRES algorithms, respectively, and let $\mathcal{E}_m^{FG} = [\boldsymbol{\varepsilon}_1^{FG}, \dots, \boldsymbol{\varepsilon}_m^{FG}]$, where $\boldsymbol{\varepsilon}_i^{FG}$ are the ith error vectors given by* (22). *Then there exists vectors $\mathbf{y}_k^{FG} \in \mathbb{R}^m$, $k = 1, \dots, m-1$, such that the following holds:*

$$\|\mathbf{r}_m^{FG}\| \le \kappa_2(V_{m+1}^{FG}) \left(\|\mathbf{r}_m^G\| + \sum_{k=0}^{m-1} \|A(M^{-1}A)^k \mathcal{E}_m^{FG}\mathbf{y}_k^{FG}\| \right).$$

5. Numerical experiments.

To illustrate the behavior of FQMR we performed several numerical experiments. We begin by considering a set of examples from [21] given by a finite difference discretization of the partial differential equation

$$-\Delta u + \gamma(xu_x + yu_y) + \beta u = f \tag{34}$$

on a unit square, where f is such that the exact solution to the discretized equation $A\mathbf{x} = \mathbf{b}$ is $\mathbf{x}^H = (1, \ldots, 1)$. The parameters γ and β are chosen in one case to make the system indefinite ($\gamma = 10$ and $\beta = -100$) and in another to have a highly nonsymmetric matrix ($\gamma = 1000$ and $\beta = 10$). For $\gamma \neq 0$, A is non-Hermitian, and it is appropriate to use FQMR. The mesh is chosen as in [21] of equal size in both dimensions (32 nodes), and the corresponding matrix is thus of order 1024.

In our experiments, we ran FQMR preconditioned with the standard QMR method (FQMR-QMR), FQMR preconditioned with QMR which, in turn, is preconditioned with ILU(0) (FQMR-QMR(ILU(0))), and FQMR preconditioned with CGNE (FQMR-CGNE); see, e.g., [17], [22] for descriptions of these preconditioners. When possible, each of these was run with an outer relative residual norm tolerance of 10^{-7} and an inner relative residual tolerance ranging from 10^{-1} to 10^{-6}. In all of our experiments, our stopping criteria uses the 2-norm in computing the relative residual norm $\|\mathbf{r}_k\|/\|r_0\|$, which is consistent with our theoretical analysis. Our FORTRAN code is derived from the existing code for QMR taken from [12], and ILU(0) and CGNE are taken from [3]. Our experiments were run on a DEC-alpha machine; however, for easy comparisons with runs on other machines, in our tables and figures we report number of operations and not computer times. The trends observed in our tables and figures are very similar to those one would observe using CPU times.

TABLE 1
FQMR-QMR. $\beta = -100, \gamma = 10$, *out. tol.* $= 10^{-7}$.

Inner tol.	Out. it.	Avg. inner it.	Oper.
10^{-1}	15	97	1.70×10^{8}
10^{-2}	5	110	6.47×10^{7}
10^{-3}	3	124	4.35×10^{7}
10^{-4}	2	131	$\mathbf{3.06\times10^{7}}$
10^{-5}	2	158	3.70×10^{7}
10^{-6}	2	183	4.28×10^{7}

TABLE 2
FQMR-QMR. $\beta = 10, \gamma = 1000$, *out. tol.* $= 10^{-7}$.

Inner tol.	Out. it.	Avg. inner it.	Oper.
10^{-1}	10	122	1.43×10^{8}
10^{-2}	4	149	7.00×10^{7}
10^{-3}	3	171	6.02×10^{7}
10^{-4}	2	204	$\mathbf{4.77\times10^{7}}$
10^{-5}	2	230	5.39×10^{7}
10^{-6}	2	248	5.80×10^{7}

Tables 1 and 2 display the results of FQMR-QMR, Tables 3 and 4 display the results of FQMR-QMR(ILU(0)), and Tables 5 and 6 display the results of FQMR-CGNE. We list the average number of inner iterations needed to reach the inner tolerance, the exact number of outer iterations needed to reach the outer tolerance, and the number of operations used to complete the solution. In Tables 3 and 4, we point out that, for an inner tolerance of 10^{-1}, FQMR cannot achieve full accuracy in the outer iteration; thus for this case, QMR(ILU(0)) is not a good preconditioner. For these tables we list the outer tolerance separately at each step. We remark that a more efficient implementation of FQMR-QMR is possible where steps (8) and (9)

Table 3
FQMR-QMR(ILU(0)). $\beta = -100, \gamma = 10$.

Out. tol.	Inner tol.	Out. it.	Avg. inner it.	Oper.
10^{-4}	10^{-1}	126	31	5.96×10^{8}
10^{-7}	10^{-2}	43	36	2.31×10^{8}
10^{-7}	10^{-3}	30	38	1.74×10^{8}
10^{-7}	10^{-4}	35	40	2.09×10^{8}
10^{-7}	10^{-5}	13	37	7.28×10^{7}
10^{-7}	10^{-6}	11	39	$\mathbf{6.44\times10^{7}}$

Table 4
FQMR-QMR(ILU(0)). $\beta = 10, \gamma = 1000$.

Out. tol.	Inner tol.	Out. it.	Avg. inner it.	Oper.
10^{-3}	10^{-1}	64	31	3.03×10^{8}
10^{-7}	10^{-2}	8	36	4.37×10^{7}
10^{-7}	10^{-3}	3	59	2.67×10^{7}
10^{-7}	10^{-4}	2	78	$\mathbf{2.34\times10^{7}}$
10^{-7}	10^{-5}	2	103	3.08×10^{7}
10^{-7}	10^{-6}	2	123	3.70×10^{7}

Table 5
FQMR-CGNE. $\beta = -100, \gamma = 10$, *out. tol.* $= 10^{-7}$.

Inner tol.	Out. it.	Avg. inner it.	Oper.
10^{-1}	10	636	4.69×10^{8}
10^{-2}	10	729	5.38×10^{8}
10^{-3}	8	775	4.57×10^{8}
10^{-4}	7	729	3.77×10^{8}
10^{-5}	2	565	$\mathbf{8.35\times10^{7}}$
10^{-6}	2	669	9.89×10^{7}

Table 6
FQMR-CGNE. $\beta = 10, \gamma = 1000$, *out. tol.* $= 10^{-7}$.

Inner tol.	Out. it.	Avg. inner it.	Oper.
10^{-1}	15	512	5.67×10^{8}
10^{-2}	4	209	6.42×10^{7}
10^{-3}	3	289	6.21×10^{7}
10^{-4}	2	278	$\mathbf{4.02\times10^{7}}$
10^{-5}	2	423	6.26×10^{7}
10^{-6}	2	441	6.52×10^{7}

of Algorithm 2.1 are performed with a single call to QMR, but we have not used this feature here.

As it can be observed, reducing the inner tolerance produces a better preconditioner, and the overall convergence is improved. As is to be expected, the average number of inner iterations increases. An important observation comes from looking at the progression of total number of operations as the inner tolerance decreases from 10^{-1} to 10^{-6}. For our given outer tolerance of 10^{-7}, the amount of required operations in relation to the inner tolerance will decrease to a point and then begin to increase.

TABLE 7
FQMR-QMR: $\beta = -100, \gamma = 10$, outer tol.= 10^{-4}.

Inner tol.	Matrix dimension	Out. it.	Inner it.	Operations
10^{-1}	1024	5	96	5.63×10^{7}
10^{-1}	4096	5	187	4.38×10^{8}
10^{-1}	10000	5	276	1.58×10^{9}
10^{-1}	40000	5	1055	2.41×10^{10}
10^{-2}	1024	2	113	3.88×10^{7}
10^{-2}	4096	8	828	3.10×10^{9}
10^{-2}	10000	3	1134	3.88×10^{9}
10^{-2}	40000	3	1527	2.09×10^{10}
10^{-3}	1024	2	124	2.91×10^{7}
10^{-3}	4096	2	249	4.38×10^{8}
10^{-3}	10000	3	1181	4.05×10^{9}
10^{-3}	40000	2	2294	2.09×10^{10}

This phenomenon is consistent with the behavior of other inner-outer methods; see e.g., [5], [24]. In Tables 1, 2, 4, and 6, the smallest amount of work was achieved for an inner tolerance of 10^{-4}; in Table 5 the smallest amount of work was achieved for an inner tolerance of 10^{-5}; while in Table 3 an inner tolerance of 10^{-6} yielded the least amount of work. This demonstrates that the inner iterative method need not, in every case, be solved to the fullest precision in order to have a good preconditioner; see [4] for other examples of this occurrence. Thus, for a given flexible preconditioner and a specific outer residual norm tolerance, we can find an optimal choice of inner tolerance for minimizing work. It can be observed that this optimal choice is problem dependent. Note also that choosing an inner tolerance equal to the outer one, say, 10^{-7} in most of our cases, would imply only one outer iteration. This is equivalent to just running the inner iterative method.

We next give the results of our experiments with larger matrices A. We solve the indefinite problem, $\beta = -100, \gamma = 10$, using FQMR-QMR with varying grid sizes of 32, 64, 100, and 200, giving us matrices of dimension 1024, 4096, 10000, and 40000, respectively. Table 7 records the results for an outer tolerance of 10^{-4}. We give the results for each of the matrix sizes using the inner tolerances 10^{-1}, 10^{-2}, and 10^{-3}. We point out that for the total number of outer iterations there is little or no change. Thus, the increase in the work per variable is wholly a result of an increase in the number of inner iterations.

We present now several figures which display that in many cases the maximum attainable accuracy of FQMR is better than that of the corresponding QMR. By this we mean the smallest possible relative residual norm. Figure 1 shows the maximum attainable accuracy of FQMR-QMR(ILU(0)) and QMR(ILU(0)) for the indefinite problem, $\beta = -100$, $\gamma = 10$. Here FQMR-QMR(ILU(0)) was able to achieve a relative residual norm of 10^{-7}, while QMR(ILU(0)) terminated at 10^{-4}. This observation is typical of all our experiments with FQMR. One particularly strong example of this behavior is observed for the matrix created with $\beta = -1000.1$ and $\gamma = 10.0$. The convergence curves of QMR and FQMR-QMR for this matrix are shown in Figure 2. Notice that while QMR stagnates at 10^{-2}, we achieve a relative residual norm of 10^{-9} using FQMR-QMR for the same number of operations. FQMR-QMR can achieve an even lower relative residual norm if we allow for additional work. A value of 10^{-15} is reached in 7.42×10^{8} operations.

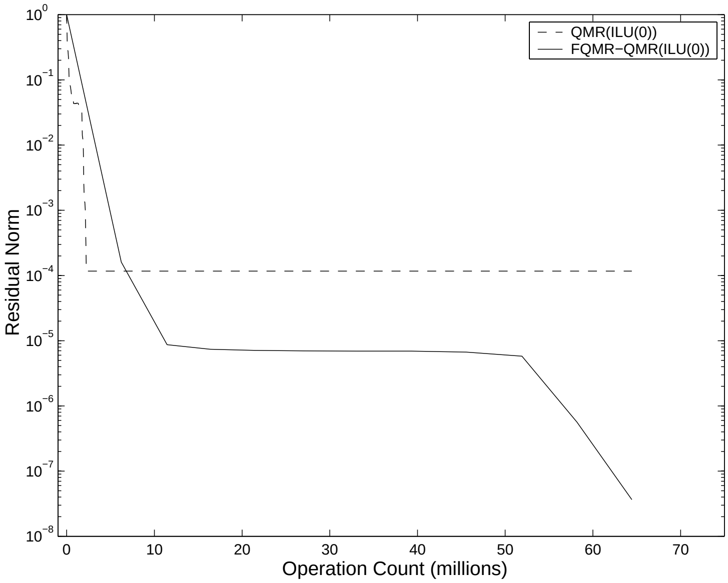


Fig. 1. *FQMR convergence:* $\beta = -100, \gamma = 10$*, inner tol.* $= 0.01$.

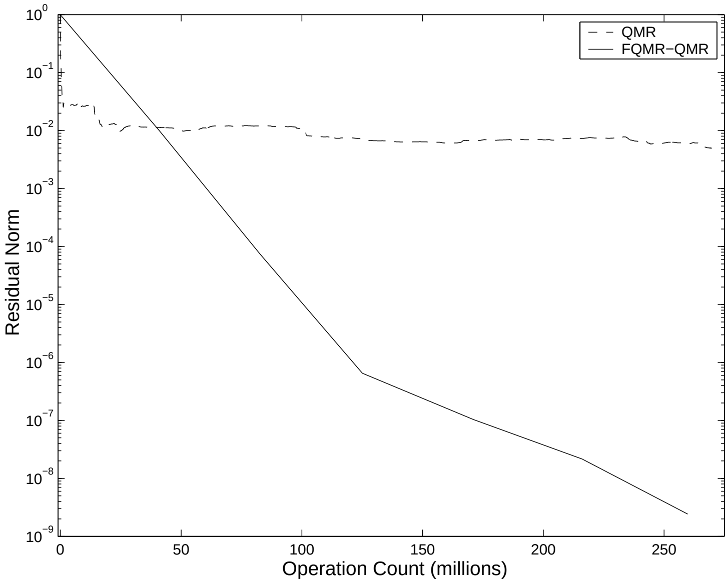


Fig. 2. *FQMR convergence:* $\beta = -1000.1, \gamma = 10$.

In Table 8, we further confirm these findings by recording achievable relative residual norms of FQMR and QMR for other choices of the matrix A. Notice that even when QMR performs well, i.e., it successfully converges to an appropriately small relative residual norm before reaching a plateau, FQMR can be shown to perform better by reaching an even smaller value. The ability to reach a lower relative residual norm by using a flexible preconditioner was observed both in the case of breakdown and stagnation. At the present time, we do not have a full explanation for this

Table 8
Comparison of achievable residual norms for FQMR-QMR and QMR.

β	γ	Res. norm - QMR	Res. norm - FQMR-QMR
-1000	10	10^{-8}	5.2×10^{-15}
1000	10	10^{-8}	6.1×10^{-15}
100	10	10^{-13}	1.42×10^{-15}
-100	10	10^{-11}	1.64×10^{-15}
10	1000	10^{-12}	5.9×10^{-15}

Table 9
*FQMR-QMR: Sherman*1 *matrix. Out. tol.* $= 10^{-7}$.

Inner tol.	Out. it.	Avg. inner it.	Oper.
10^{-1}	111	98	1.07×10^9
10^{-2}	10	148	1.47×10^8
10^{-3}	4	189	7.50×10^7
10^{-4}	2	180	$\mathbf{3.56\times10^7}$
10^{-5}	2	272	5.40×10^7
10^{-6}	2	327	6.28×10^7

phenomenon.

We comment that there are of course several ways of going beyond the point where QMR stagnates, for example, by using iterative refinement or restarting QMR. Also, if one uses the coupled two-term recurrences version of QMR, the maximum attainable residual norm is better than with the three-term recurrence version of QMR used here; see [13].

To emphasize the robustness of the new FQMR method, we ran FQMR on two matrices whose structure is different from the previous examples. These are the Sherman1 matrix and the Sherman5 matrix from [8]. They are the first and fifth matrices from the Sherman collection, respectively. Both represent oil reservoir simulations, with Sherman1 coming from a black oil simulation with shale barriers on a $10 \times 10 \times 10$ grid with one unknown per grid point, and Sherman5 coming from a fully implicit black oil simulator on a $16 \times 23 \times 3$ grid with three unknowns per grid point. The Sherman1 matrix is of dimension 1000 and has 3750 nonzeros, and the Sherman5 matrix is of dimension 3312 and has 20793 nonzeros.

Table 9 displays the convergence behavior of FQMR-QMR for the Sherman1 matrix, and Table 10 displays the convergence behavior of FQMR-QMR for the Sherman5 matrix. Notice that for both of these matrices the convergence behavior of FQMR-QMR remains comparable to what we have seen in all of the previous examples. Tables 9 and 10, also, display a consistency with our other examples in that the total work required for solving these problems with an outer tolerance of 10^{-7} reaches a minimum when the inner tolerance is 10^{-4} for Table 9 and 10^{-5} for Table 10. Thus, once more we see an optimal preconditioner for our implementation is achieved when using a less precise inner iteration.

To emphasize the property that we have observed in FQMR of achieving a lower relative residual norm than QMR, we display full convergence results of both QMR and FQMR-QMR on the Sherman5 matrix in Figure 3. Here QMR can achieve a tolerance of only 2.0×10^{-8}, while FQMR-QMR reaches 3.5×10^{-16}. Therefore, once again, FQMR can outperform QMR when an extremely low residual norm is required.

Table 10
FQMR-QMR: Sherman5 matrix.

Out. tol.	Inner tol.	Out. it.	Avg. inner it.	Oper.
10^{-2}	10^{-1}	15	138	8.89×10^{8}
10^{-3}	10^{-2}	2	570	4.09×10^{8}
10^{-7}	10^{-3}	16	2495	1.70×10^{10}
10^{-7}	10^{-4}	3	1475	1.90×10^{9}
10^{-7}	10^{-5}	2	1832	$\mathbf{1.57\times10^{9}}$
10^{-7}	10^{-6}	2	2069	1.77×10^{9}

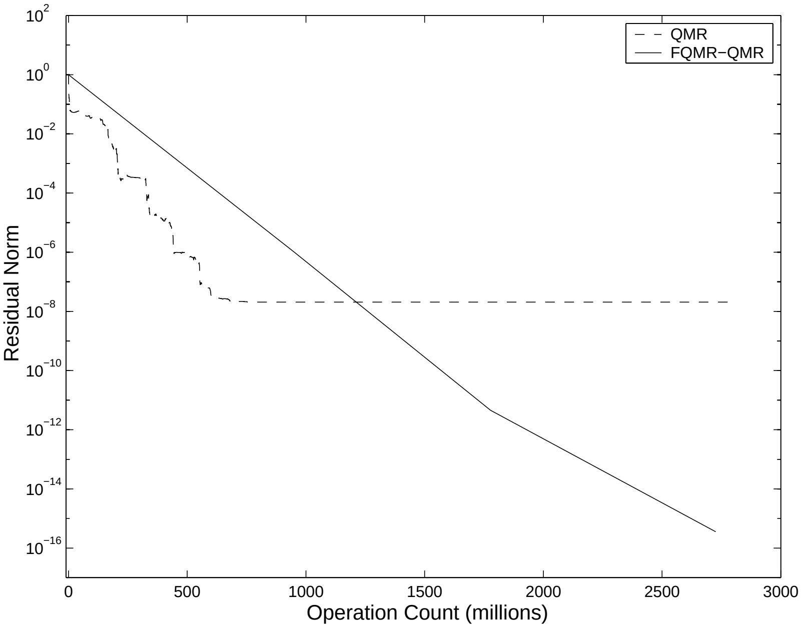


Fig. 3. *QMR vs. FQMR-QMR: Sherman5 matrix.*

6. Conclusions. We have formulated a flexible version of QMR for the solution of large sparse non-Hermitian nonsingular systems of linear equations. This new method, FQMR, converges to the solution of the linear system, as long as the new vectors generated at each step are linearly independent of the previous ones. Theoretical bounds on the norm of the residual at each step were given. These bounds are (as is to be expected) not as good as those obtained using the same preconditioner at each step. The advantage is that the variable preconditioner can be less onerous. Furthermore, there is the potential of great gains in cases of an adaptive preconditioner. One of the advantages of FQMR over FGMRES is the fact that the low memory requirements are fixed from the beginning.

Using the methodology developed to produce the mentioned bounds, we have also contributed to the analysis of FGMRES [21].

Numerical experiments of the type found in [21] illustrate the convergence behavior of FQMR and demonstrate that FQMR is capable of attaining lower residual norms than the standard QMR with three-term recurrence. Other experiments presented show a similar behavior.

Acknowledgments. We thank the referees, as well as Michele Benzi and Valeria Simoncini, for their valuable comments and suggestions. We greatly benefited from them.

REFERENCES

[1] O. Axelsson and P. S. Vassilevski, *A black box generalized conjugate gradient solver with inner iterations and variable-step preconditioning*, SIAM J. Matrix Anal. Appl., 12 (1991), pp. 625–644.

[2] J. Baglama, D. Calvetti, G. H. Golub, and L. Reichel, *Adaptively preconditioned GMRES algorithms*, SIAM J. Sci. Comput., 20 (1998), pp. 243–269.

[3] R. B. Bramley, *SPLIB Software Package*, Indiana University, Bloomington, IN, 1995.

[4] A. Bouras and V. Frayssé, *A Relaxation Strategy for Inexact Matrix-Vector Products for Krylov Methods*, Technical Report TR/PA/00/15, CERFACS, Toulouse, France, 2000.

[5] M. J. Castel, V. Migallón, and J. Penadés, *On Parallel two-stage methods for Hermitian positive definite matrices with applications to preconditioning*, Electron. Trans. Numer. Anal., 12 (2001), pp. 88–112.

[6] R. S. Dembo, S. C. Eisenstat, and T. Steihaug, *Inexact Newton methods*, SIAM J. Numer. Anal., 19 (1982), pp. 400–408.

[7] E. De Sturler, *Truncation strategies for optimal Krylov subspace methods*, SIAM J. Numer. Anal., 36 (1999), pp. 864–889.

[8] I. S. Duff, R. G. Grimes, and J. G. Lewis, *Sparse matrix test problems*, ACM Trans. Math. Software, 15 (1989), pp. 1–14.

[9] M. Eiermann and O. G. Ernst, *Geometric aspects in the theory of Krylov subspace methods*, Acta Numer., to appear.

[10] H. C. Elman, O. G. Ernst, and D. P. O'Leary, *A Multigrid Method Enhanced by Krylov Subspace Iteration for Discrete Helmholtz Equations*, Technical Report CS-TR 4029, University of Maryland, Institute for Advanced Computer Studies, College Park, MD, 1999.

[11] R. W. Freund and N. M. Nachtigal, *QMR: A quasi-minimal residual method for non-Hermitian linear systems*, Numer. Math., 60 (1991), pp. 315–339.

[12] R. W. Freund and N. M. Nachtigal, *QMRPACK Software Package*, NASA Ames Research Center, Moffet Field, CA, 1993.

[13] R. W. Freund and N. M. Nachtigal, *An implementation of the QMR method based on coupled two-term recurrences*, SIAM J. Sci. Comput., 15 (1994), pp. 313–337.

[14] A. Frommer and D. B. Szyld, *H-splittings and two-stage iterative methods*, Numer. Math., 63 (1992), pp. 345–356.

[15] G. H. Golub and M. L. Overton, *The convergence of inexact Chebyshev and Richardson iterative methods for solving linear systems*, Numer. Math., 53 (1988), pp. 571–593.

[16] G. H. Golub and Q. Ye, *Inexact preconditioned conjugate gradient method with inner-outer iteration*, SIAM J. Sci. Comput., 21 (1999), pp. 1305–1320.

[17] A. Greenbaum, *Iterative Methods for Solving Linear Systems*, Frontiers Appl. Math. 17, SIAM, Philadelphia, 1997.

[18] P. Guillaume, Y. Saad, and M. Sosonkina, *Rational Approximation Preconditioners for General Sparse Linear Systems*, Technical Report umsi-99-209, University of Minnesota, Minneapolis, 1999.

[19] N. M. Nachtigal, *A Look-Ahead Variant of the Lanczos Algorithm and Its Application to the Quasi-Minimal Residual Method for Non-Hermitian Linear Systems*, Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA, 1991.

[20] Y. Notay, *Flexible conjugate gradient*, SIAM J. Sci. Comput., 22 (2000), pp. 1444–1460.

[21] Y. Saad, *A flexible inner-outer preconditioned GMRES algorithm*, SIAM J. Sci. Comput., 14 (1993), pp. 461–469.

[22] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing, Boston, 1996.

[23] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[24] D. B. Szyld and M. T. Jones, *Two-stage and multisplitting methods for the parallel solution of linear systems*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 671–679.

[25] H. A. Van der Vorst and C. Vuik, *GMRESR: a family of nested GMRES methods*, Numer. Linear Algebra Appl., 1 (1994), pp. 369–386.

SIAM J. Sci. Comput.
Vol. 23, No. 2, pp. 381–397

ON PRECONDITIONING NEWTON–KRYLOV METHODS IN SOLIDIFYING FLOW APPLICATIONS[*]

D. A. KNOLL[†], W. B. VANDERHEYDEN[†], V. A. MOUSSEAU[†], and D. B. KOTHE[‡]

Abstract. Solidifying flow equations can be used to model industrial metallurgical processes such as casting and welding, and material science applications such as crystal growth. These flow equations contain locally stiff nonlinearities at the moving phase-change interface. We are developing a three-dimensional parallel simulation tool for such problems using a Jacobian-free Newton–Krylov solver and unstructured finite volume methods. A segregated (distributed, block triangular) preconditioning strategy is being developed for the Newton–Krylov solver. In this preconditioning approach we are only required to approximately invert matrices coming from a single field variable, not matrices arising from a coupled system. Additionally, simple linearizations are used in constructing our preconditioning operators. The preconditioning strategy is presented along with the performance of the methods. We consider problems in phase-change heat transfer and the thermally driven incompressible Navier–Stokes equations separately. This is a required intermediate step toward developing a successful preconditioning strategy for the fully coupled physics problem.

Key words. Newton–Krylov methods, preconditioning, solidifying flow

AMS subject classifications. 65H10, 65M50, 76T05

PII. S1064827500374303

1. Introduction.

Numerical simulation of solidifying flow equations plays an important role in the analysis of industrial metallurgical processes such as casting and welding and material science applications such as crystal growth. These flow equations contain locally stiff nonlinearities at the moving phase-change interface. The typical equation set is a combination of the incompressible Navier–Stokes equations and an energy equation which includes phase-change and latent heat evolution. Examples of numerical solutions to solidifying flow equations are [VCM87, Dan89].

There are many challenges to numerically integrating the equations of solidifying flow. We focus here on fixed-grid approaches, although moving mesh methods can also be used. In constructing a solution algorithm one must choose a set of dependent variables. In most applications temperature is used as the dependent variable for the energy equation, whereas we use total enthalpy [KKL99]. One may chose to iteratively converge the heat transfer phase-change evolution within a time step as in [VCM87] or approximately linearize this coupling within a time step using an "effective heat capacity" method as in [Dan89]. Additionally, one may choose a segregated solution procedure to couple the flow equations and the energy equation within a time step, as is done in [VCM87], or one can solve this system with a Newton based approach, as is done in [Dan89]. Note that while [Dan89] has demonstrated the applicability of a Newton based approach to solidifying flow, the more modern Jacobian-free Newton–Krylov (JFNK) approach was not applied in that study. However, the convergence of a Newton based approach applied to solidifying flow has been established in [Dan89].

[*] Received by the editors June 26, 2000; accepted for publication (in revised form) February 7, 2001; published electronically July 10, 2001. This work was supported under the auspices of the U.S. Department of Energy under DOE contract W-7405-ENG-36 at Los Alamos National Laboratory.
http://www.siam.org/journals/sisc/23-2/37430.html

[†] Los Alamos National Laboratory, Theoretical Division, MS B216, Los Alamos, NM 87545 (nol@lanl.gov, wbv@lanl.gov, vmss@lanl.gov).

[‡] Los Alamos National Laboratory, Computer and Computational Science Division, MS B250, Los Alamos, NM 87545 (dbk@lanl.gov).

Our algorithmic approach will use total enthalpy as a dependent variable, accurately capturing the latent-heat exchange within a time step [KKL99]. We employ a fully coupled Newton–Krylov method to integrate the system in a Jacobian-free approach [CJ84, BS90]. We have recently had success using this approach for boundary value problems involving coupling between fluid flow and phase-change or finite rate chemistry [Kno98, KMK96]. We use the "effective heat capacity" method and a segregated flow solver approach to precondition our outer Newton–Krylov solver. In this study, the scalar elliptic systems which arise in the preconditioner will be approximately inverted using low-complexity multilevel methods [KR00, MKR00, KM00].

Our equation system is comprised of a continuity equation,

$$\nabla \cdot \mathbf{u} = 0, \tag{1.1}$$

a set of momentum equations,

$$\frac{\partial[\rho \mathbf{u}]}{\partial t} + \nabla \cdot (\rho_l \mathbf{u}\mathbf{u}) - \nabla \cdot (\nu \nabla \mathbf{u}) = -\nabla P + \rho \mathbf{g} \beta (T - T_{ref}) - \alpha(\epsilon_s)\mathbf{u}, \tag{1.2}$$

and an energy equation for total mixture enthalpy,

$$\frac{\partial[\rho H]}{\partial t} + \nabla \cdot ([\rho H]\mathbf{u}) - \nabla \cdot (\kappa(\epsilon_s, T)\nabla T) = 0. \tag{1.3}$$

Here the mixture density and volumetric enthalpy are

$$[\rho] = \epsilon_s \rho_s + (1 - \epsilon_s)\rho_l, \;\; [\rho H] = \epsilon_s \rho_s h_s + (1 - \epsilon_s)\rho_l h_l. \tag{1.4}$$

The subscript s indicates solid, and the subscript l indicates liquid. The phase enthalpies and total enthalpy are

$$h_s = C_{p,s}T, \;\; h_l = C_{p,l}T + L, \;\; H = C_p T + (1 - \epsilon_s)L. \tag{1.5}$$

ρ is density, $\mathbf{u}$ is velocity, T is temperature, P is pressure, ϵ_s is the solid volume fraction, κ is the thermal conductivity, ν is the kinematic viscosity, α is a solid-liquid friction coefficient, β is the thermal expansion coefficient, $\mathbf{g}$ is the gravity vector, C_p is the mixture heat capacity ($C_p = \epsilon_s C_{p,s} + (1 - \epsilon_s)C_{p,l}$), and L is the latent heat of fusion. T_{ref} is the temperature around which the density is linearized for the Boussinesq approximation. The functional form for the mixture thermal conductivity is

$$\kappa(\epsilon_s, T) = \epsilon_s \kappa_s(T) + (1 - \epsilon_s)\kappa_l(T). \tag{1.6}$$

The solid phase, assumed to be stationary in this model, has the ability to absorb momentum from the fluid. The drag (friction) at the solid-liquid interface is modeled as a loss term in the momentum equations, although an artificially increased viscosity has also been used for this purpose [Dan89]. For simplicity of presentation we have also ignored the effects of solute transport and solutal buoyancy effects, which are important for alloys.

In section 2 we outline our proposed solution algorithm. It is this section which will isolate the two individual components of our preconditioning strategy to be studied in this paper. In section 3 we study separate components of the algorithm as applied to phase-change heat transfer and thermally driven incompressible flow. It is this section which contains our new results and lays the foundation for the fully coupled preconditioning strategy. Conclusions are given in section 4.

2. Solution algorithm. We use a JFNK method with segregated solution method (only scalar inversions) as a preconditioner. We have recently applied a similar approach to time dependent reaction-diffusion systems [MKR00]. Much of the material in this section is not new; however, we include it for completeness. Newton's method requires the solution of the linear system

$$\mathbf{J}^k \delta w^k = -\mathbf{F}(\mathbf{w}^k), \qquad \mathbf{w}^{k+1} = \mathbf{w}^k + S^k \delta w^k, \tag{2.1}$$

where $\mathbf{J}$ is the Jacobian matrix, $\mathbf{F}(\mathbf{w})$ is the nonlinear system of equations, $\mathbf{w}$ is the state vector, S is a damping scalar, and k is the nonlinear iteration index. The Newton iteration continues until $||\mathbf{F}(\mathbf{w}^k)||_2/||\mathbf{F}(\mathbf{w}^0)||_2$ falls below a specified tolerance. We will use 1.0×10^{-6} for this tolerance. When using a Krylov method to solve (2.1) (we are using GMRES [SS86]), the action of the Jacobian is only required in the form of matrix-vector products which can be approximated with finite differences [CJ84, BS90]. The right preconditioned form of this approximation, where $\mathbf{v}$ is a general Krylov vector, is

$$\mathbf{J}\mathbf{P}^{-1}\mathbf{v} \approx \left[\mathbf{F}(\mathbf{w} + \epsilon \mathbf{P}^{-1}\mathbf{v}) - \mathbf{F}(\mathbf{w})\right] / \epsilon, \tag{2.2}$$

where $\mathbf{P}$ is our preconditioning matrix. In this study ϵ, the perturbation parameter, is given by

$$\epsilon = \frac{1}{N||\mathbf{v}||_2} \sum_{m=1}^{N} b|w_m|, \tag{2.3}$$

where N is the linear system dimension and b is a constant whose magnitude is within a few orders of magnitude of the square root of machine roundoff ($b = 10^{-5}$ for this study), and w_m is the mth component of $\mathbf{w}$.

In the JFNK method only the matrix $\mathbf{P}$ is formed, and only the matrix $\mathbf{P}$ is iteratively inverted. There are two important decisions to make regarding $\mathbf{P}$.

1. What linearization should be used to form $\mathbf{P}$?
2. What linear iterative method should be used to solve $\mathbf{P}y = \mathbf{v}$?

It is in answering these questions where our approach becomes novel in its application to solidifying flow problems.

Additionally, we use an inexact Newton method, where the required linear convergence tolerance on a given nonlinear iteration is proportional to the nonlinear residual,

$$\parallel \mathbf{J}^k \delta w^k + \mathbf{F}(\mathbf{w}^k) \parallel_2 < \gamma \parallel \mathbf{F}(\mathbf{w}^k) \parallel_2 . \tag{2.4}$$

For this work we use a constant $\gamma = 5.0 \times 10^{-3}$.

As indicated above, we allow for damping in each Newton iteration. This damping limits the maximum change in $\mathbf{w}$ to a fixed percent per Newton iteration. This process allows for an increase in nonlinear residual but prevents thermodynamic variables from becoming unphysical (i.e., becoming negative). Our expression for the damping scalar, S, is

$$S = \min[1, \min_i(f * w_i/\delta w_i)], \tag{2.5}$$

where f is a user supplied fraction for a 20 percent allowed change, $f = 0.2$. Only the thermodynamic variables in $\mathbf{w}$ are used to evaluate S. It can be seen that S approaches unity as the Newton method produces updates which are small.

Next our preconditioning strategy is described. First, we present the strategy in general, and in the next section we present specifics. Our segregated preconditioning strategy can be viewed as belonging to a broader class of lower block triangular preconditioners, examples of which include [ES96, Kla98]. However, it is most closely related to the recent preconditioning methods put forth for thermally driven flows in [PT01]. The relationship between these approaches will be discussed more at the end of this section.

We are interested in time accurate simulation and expect, for the most part, to integrate with time steps which respect the advective time scales. This allows us to perform a simple Picard-type linearization of advection in the preconditioner without a significant increase in outer GMRES iterations. When evaluating the preconditioning operator, the advecting velocity is evaluated at the previous nonlinear iteration level. This is similar to the approach used in [ES96], except that in our approach the nonlinearity of the advection operator is treated with an outer Newton–Krylov iteration. This simplified linearization applied in the preconditioner should not be viewed as restricting the application of the method to some limited range of Reynolds numbers. It may be viewed as restricting the time step to some small multiple (say, 10) of the explicit advection stability limit. However, in practice, it has worked well on steady-state problems [KR00, KM00]. Within a time step we expect that stiffness will come from phase-change, solid-liquid friction, pressure-velocity (or stream function–vorticity) coupling, thermal body forces, and possibly the conduction and viscous operators. In this study we focus on constructing effective preconditioning for phase-change and stream function–vorticity–thermal body force coupling separately. These methods will form the foundation of a total preconditioning strategy. Again, we feel that the convergence of a Newton based method on the complete problem can be inferred from the success of [Dan89]. Our goal is to construct an effective preconditioning strategy for a more modern JFNK approach.

Consider a two-dimensional (2D) Newton system where the dependent variables are u, v, P, and H in each control volume with $\mathbf{w} = [u, v, P, H]^T$, $\delta\mathbf{w}^k = [\delta u, \delta v, \delta P, \delta H]^T$, and $\mathbf{F}(\mathbf{w}^k) = [F_u, F_v, F_P, F_H]^T$, in matrix form

$$\begin{bmatrix} J_{uu} & J_{uv} & J_{uP} & J_{uH} \\ J_{vu} & J_{vv} & J_{vP} & J_{vH} \\ J_{Pu} & J_{Pv} & 0 & 0 \\ J_{Hu} & J_{Hv} & 0 & J_{HH} \end{bmatrix} \begin{bmatrix} \delta u \\ \delta v \\ \delta P \\ \delta H \end{bmatrix} = - \begin{bmatrix} F_u \\ F_v \\ F_P \\ F_H \end{bmatrix}.$$

Here $J_{uu} = \frac{\partial F_u}{\partial u}$ (this submatrix has a dimension equal to the number of finite volumes), and F_u is the nonlinear function for the u momentum equation. The system has been written in block form, blocked by a conservation equation. The preconditioning "process" requires the mapping of a vector in a similar manner:

$$[\delta u, \delta v, \delta P, \delta H]^T = \mathbf{P}^{-1}(-[F_u, F_v, F_P, F_H]^T). \tag{2.6}$$

$\mathbf{P}$, however, has only to approximate $\mathbf{J}$. Assuming a Picard linearization for the advecting velocity, $J_{uv} = J_{vu} = J_{Hu} = J_{Hv} = 0$, and the preconditioner will be

$$\begin{bmatrix} P_{uu} & 0 & P_{uP} & P_{uH} \\ 0 & P_{vv} & P_{vP} & P_{vH} \\ P_{Pu} & P_{Pv} & 0 & 0 \\ 0 & 0 & 0 & P_{HH} \end{bmatrix} \begin{bmatrix} \delta u \\ \delta v \\ \delta P \\ \delta H \end{bmatrix} = - \begin{bmatrix} F_u \\ F_v \\ F_P \\ F_H \end{bmatrix}.$$

P_{uu} is an approximation to J_{uu}, formed by a Picard linearization, not a Newton linearization. It is a linear convection diffusion operator with a diagonal contribution from the time derivative. We need to approximate the action of $\mathbf{P}^{-1}$ on a vector. The first step of the preconditioning process is to approximately invert the phase-change heat transfer problem:

$$P_{HH}\delta H = -F_H.$$

We will focus on this step and the formation of P_{HH} in the next section. Incorporating approximate phase-change effects into P_{HH} is one of the main contributions of this paper. P_{HH} is a linear convection diffusion operator with a diagonal contribution from the time derivative, as well as phase-change. With δH, we are left with a pressure-velocity coupling problem, given by

$$\begin{bmatrix} P_{uu} & 0 & P_{uP} \\ 0 & P_{vv} & P_{vP} \\ P_{Pu} & P_{Pv} & 0 \end{bmatrix} \begin{bmatrix} \delta u \\ \delta v \\ \delta P \end{bmatrix} = - \begin{bmatrix} F_u - P_{uH}\delta H \\ F_v - P_{vH}\delta H \\ F_P \end{bmatrix}.$$

Solutions to this problem can be obtained in a number of ways. For example, a few passes of the segregated SIMPLE algorithm [Pat80] could be used. In fact, the SIMPLE algorithm has recently been used as a smoother for a multigrid (MG) preconditioner inside a JFNK method [PT01]. As is shown in [PT01], the SIMPLE algorithm effectively puts the above linear system in lower block triangular form with a required outer iteration. This can also be seen in [Wes92], where SIMPLE is discussed as an MG smoother. The performance of the SIMPLE algorithm as a solver is well documented in the enginering community, although frequently several iterations are required for convergence. When used as a preconditioner to JFNK, only a few iterations of the segregated solver are employed.

The philosophy behind our segregated preconditioner is the same as in [MKR00]. One can use an operator split, or a segregated, solution method (a non-Newton approach) as a preconditioner to an outer JFNK solver. The resulting preconditioner will have a block triangular form with an outer iteration. The individual blocks are only approximately inverted in each outer iteration of the preconditioner. This is a different approach to producing a lower block triangular preconditioner as compared to [ES96], although the resulting preconditioners are similar. Further investigation of the similarities between these approaches is clearly warranted, but it is out of the scope of current study.

In the next section we will apply a segregated (block triangular) preconditioner to a time dependent thermally driven flow problem and compare its performance to a coupled MG based preconditioner.

3. Algorithm specifics.

We have described our JFNK method and our preconditioning philosophy as applied to solidifying flow. In this section we give specific examples of algorithm application and performance, considering the phase-change heat conduction problem and thermally driven incompressible flow separately.

3.1. The phase-change heat conduction problem.

In this subsection we ignore fluid flow, assume constant density, heat capacity, and thermal conductivity, and focus on JFNK as applied to the phase-change heat conduction problem. Our initial work on this problem is detailed in [KKL99]. Here we focus on significant improvements to the formation of the preconditioner, and we demonstrate the applicability of the JFNK algorithm to a problem with multiple phase transitions. Additionally, we will briefly describe the method used for time step control on phase change problems. As stated, we use total enthalpy as a dependent variable, and thus we need to express temperature as a function of enthalpy, $T = \mathcal{T}(H)$, to perform implicit conduction. Given our assumptions, a one-dimensional (1D) version of the energy equation is

$$\frac{\partial H}{\partial t} - \frac{\kappa}{\rho}\frac{\partial}{\partial x}\left[\frac{\partial \mathcal{T}(H)}{\partial x}\right] = 0 \tag{3.1}$$

with $H = C_pT + (1-\epsilon_s)L$. An implicit discrete version of this equation is

$$F_{H,i} = \frac{H_i^{n+1} - H_i^n}{\Delta t} - \frac{\kappa}{\rho \Delta x^2}[\mathcal{T}(H_{i+1}^{n+1}) - 2\mathcal{T}(H_i^{n+1}) + \mathcal{T}(H_{i-1}^{n+1})] = 0, \tag{3.2}$$

where the diffusion operator is approximated with second order centered differences.

For a pure material (Stefan problem, isothermal solidification) with a melting temperature T_m, the function $\mathcal{T}(H)$ is given by

$$\mathcal{T}(H) = \begin{cases} H/C_p, & H < C_pT_m, \\ T_m, & C_pT_m \le H \le C_pT_m + L, \\ (H-L)/C_p, & H > C_pT_m + L, \end{cases}$$

where $\epsilon_s = 1 - \frac{H - C_pT_m}{L}$. With this function one can evaluate the nonlinear residual given a guess for H_i^{n+1}. For a more complex multicomponent system, such as a binary eutectic alloy (nonisothermal solidification), the function $\mathcal{T}(H)$ becomes more complex:

$$\mathcal{T}(H) = \begin{cases} H/C_p, & H < C_pT_{eut}, \\ T_{eut}, & C_pT_{eut} \le H \le C_pT_{eut} + (1-\epsilon_{s,eut})L, \\ (H-(1-\epsilon_s)L)/C_p, & C_pT_{eut} + (1-\epsilon_{s,eut})L \le H \le C_pT_l + L, \\ (H-L)/C_p, & H > C_pT_l + L, \end{cases}$$

where T_{eut} is the eutectic temperature, T_l is the liquidus temperature, and $\epsilon_{s,eut}$ is the solid volume fraction at T_{eut}. The region of nonisothermal solidification ($T_{eut} < T < T_l$) is referred to as the "mushy zone," where solid and liquid can coexist. In the "mushy zone," one must simultaneously satisfy the energy balance, the phase diagram, and a local scale model. In this region a small nonlinear system must be solved to extract T as a function of H.

The local scale model is a subgrid model used to account for solute diffusion in the solid. For example, if the Schiel local scale model (no solute diffusion in the solid) is being used, then the two-equation nonlinear system becomes

$$C_pT + (1-\epsilon_s)L - H = 0,$$

$$\epsilon_s + \left(\frac{T_f - T_l}{T_f - T}\right)^{-(1-k)} - 1.0 = 0. \tag{3.3}$$

Solutions to this system must be found in each finite volume in the "mushy zone." (k, T_f, T_l, and H are known.) We are solving for T and ϵ_s, using a local Newton method.

Next we need to form a preconditioning matrix, $\mathbf{M}$, which is an approximation to $\mathbf{J}$. In this subsection we refer to the preconditioning operator as $\mathbf{M}$, not $\mathbf{P}$. We do this to symbolize that we are only considering the phase-change heat conduction problem. The Jacobian for the constant property Stefan problem (isothermal solidification) is easy to form analytically [KR90].

$$\begin{aligned} J_{i,i-1} &= \frac{\partial F_i}{\partial H_{i-1}} = -\frac{\kappa}{\rho \Delta x^2}\mathcal{T}'(H_{i-1}),\\ J_{i,i} &= \frac{\partial F_i}{\partial H_i} = \frac{1}{\Delta t} + 2\frac{\kappa}{\rho \Delta x^2}\mathcal{T}'(H_i),\\ J_{i,i+1} &= \frac{\partial F_i}{\partial H_{i+1}} = -\frac{\kappa}{\rho \Delta x^2}\mathcal{T}'(H_{i+1}). \end{aligned} \tag{3.4}$$

Here, $\mathcal{T}'$ is the derivative of the temperature function with respect to enthalpy for the Stefan problem, given by

$$\mathcal{T}'(H) = \begin{cases} 1/C_p, & H < C_pT_m,\\ 0, & C_pT_m \le H \le C_pT_m + L,\\ 1/C_p, & H > C_pT_m + L. \end{cases}$$

Forming the Jacobian is not as simple for alloys (nonisothermal solidification). The simplest approximation, which we call M1, results from assuming a constant value, $\mathcal{T}'(H) = 1/C_p$, for all values of enthalpy when evaluating the preconditioning matrix $\mathbf{M}$. This is equivalent to ignoring phase change in the preconditioner and preconditioning only heat conduction. Thus each row of the preconditioning matrix becomes

$$\begin{aligned} M_{i,i-1} &= -\frac{\kappa}{\rho C_p \Delta x^2},\\ M_{i,i} &= \frac{1}{\Delta t} + 2\frac{\kappa}{\rho C_p \Delta x^2},\\ M_{i,i+1} &= -\frac{\kappa}{\rho C_p \Delta x^2}. \end{aligned} \tag{3.5}$$

This preconditioning matrix is used in [KKL99]. Here we also consider a simple improvement to this preconditioning matrix obtained by using the "effective heat capacity" approach [Dan89]. In the effective heat capacity approach the time derivative of enthalpy is transformed into a time derivative of temperature. The effective heat capacity, $\frac{dH}{dT}$, is evaluated at the previous time step.

$$\left(\frac{dH}{dT}\right)^n \frac{\partial T}{\partial t} - \frac{\kappa}{\rho}\frac{\partial}{\partial x}\left[\frac{\partial T}{\partial x}\right]^{n+1} = 0, \quad \left(\frac{dH}{dT}\right)^n = \frac{H^n - H^{n-1}}{T^n - T^{n-1}}. \tag{3.6}$$

In evaluating the improved preconditioner, M2, $\mathcal{T}'(H)^{n+1} = (\mathcal{T}(H^n+\delta H)-\mathcal{T}(H^n))/\delta H$. For both M1 and M2 five passes of symmetric Gauss–Seidel (SGS) are used to approximate $\mathbf{M}^{-1}$.

Before studying algorithm performance we briefly describe the time step control used to follow solidification fronts which we have borrowed from [RK99]. Assume that our nonlinear parabolic equation can be expressed as a hyperbolic equation, i.e.,

$$\frac{\partial H}{\partial t} - \frac{\kappa}{\rho}\frac{\partial}{\partial x}\left[\frac{\partial \mathcal{T}(H)}{\partial x}\right] = 0 \tag{3.7}$$

as

$$\frac{\partial H}{\partial t} + V_f \frac{\partial H}{\partial x} = 0. \tag{3.8}$$

We are motivated to make this assumption since many nonlinear parabolic problems, such as radiation diffusion [RK99] and phase-change heat transfer produce a solution with a hyperbolic character (a transporting front). Here V_f is the effective velocity with which the solidification front moves. If one can approximately evaluate V_f, then it can used to set a time step which does not allow the front to propagate more than one mesh spacing in one time step. We use

$$V_f = \frac{\parallel \frac{\partial H}{\partial t} \parallel_1}{\parallel \frac{\partial H}{\partial x} \parallel_1}, \quad \Delta t = C\frac{\Delta x}{V_f}, \tag{3.9}$$

evaluated on the discrete solution at the previous time step, and we choose $C \leq 1$. We have found this method for choosing the time step for solidification problems a robust way to maintain temporal accuracy.

As a first model problem we consider a 1D binary eutectic problem, which has been adapted from [SV96], in the region $0 \leq x \leq 0.4$. This problem is done in dimensional form with $C_p = 1000\frac{J}{kgK}$, $L = 4.0 \times 10^5 \frac{J}{kg}$, $\rho = 2400\frac{kg}{m^3}$, and $\kappa = 100\frac{W}{mK}$. The liquidus temperature, at an initial solute concentration of 5.0 percent, is $T_l = 904.2\ K$. The eutectic temperature is $T_{eut} = 821.2\ K$, and the melting temperature of the pure solvent is $T_f = 921.2\ K$. The slope of the liquidus line is $m_l = 3.4$, and the partition coefficient is $k = 0.15$. The initial condition for temperature is $905.2\ K$. At time equal to zero, the left boundary is set to a temperature of $621.2\ K$, and solidification proceeds. The accuracy of the solution to this problem was demonstrated in [KKL99]. Here we demonstrate the efficiency gained from the improved preconditioner (M2).

Table 1 presents results for different fixed time step sizes with the Scheil local scale model, with nx (the number of finite volumes)= 100 . We can see that M2 performs significantly better than M1. Tables 2 and 3 present results for a 2D version of this problem on different grids using a variable time step with $C = 0.25$. For this problem $0 \leq x \leq 0.4$ and $0 \leq y \leq 0.4$. The temperature on all four boundaries is set to $621.2\ K$. Again, M2 is significantly better than M1. We see the impact on the 2D problem being greater than on the 1D problem. This is because there is a larger fraction of grid cells in the solidification front. Comparing Tables 2 and 3 for the 60×60 grid, M2 has reduced the required GMRES iterations by a factor of five and has reduced the CPU time by a factor of four. It should be mentioned that over the course of the 2D simulation the variable time step increased by nearly two orders of magnitude.

Figure 1 shows the solution of the 1D problem at three different times. It is normalized to show the effect of phase change on the enthalpy field. In these plots temperature is equal to $\frac{T}{621.2}$, and enthalpy is equal to $\frac{H}{621.2C_p}$. With this normalization temperature is equal to enthalpy in the solidified region. In the pure liquid

TABLE 1

Algorithm performance as a function of time step size and preconditioning matrix on the binary alloy problem; $nx = 100$, time $= 500$.

Precond. and time step	Number of Newton its. per time step	GMRES its. per Newton it.	CPU time
M1, Δt=1.0	2.8	8.1	22
M2, Δt=1.0	2.4	1.5	8.3
M1, Δt=2.5	3.3	10.7	12.6
M2, Δt=2.5	3.0	2.2	4.8
M1, Δt=5.0	3.7	12.0	7.7
M2, Δt=5.0	3.6	2.9	3.2

TABLE 2

Algorithm performance as a function of grid dimension on the 2D binary alloy problem; M2, $C = 0.25$, time $= 150$.

Grid dim.	Number of time steps	Number of Newton its. per time step	GMRES its. per Newton it.	CPU time
30x30	43	3.67	3.0	1
40x40	45	4.67	3.3	2.5
50x50	47	5.33	4.5	5.7
60x60	50	5.0	5.2	10.7

TABLE 3

Algorithm performance as a function of grid dimension on the 2D binary alloy problem; M1, $C = 0.25$, time $= 150$.

Grid dim.	Number of time steps	Number of Newton its. per time step	GMRES its. per Newton it.	CPU time
30x30	43	4.33	17.0	3.25
40x40	45	4.67	18.4	7.3
50x50	47	5.67	21.3	22.5
60x60	50	5.0	25.6	40.8

region temperature is equal to enthalpy plus $\frac{L}{621.2C_p}$. Note that M1 ignores the jump in the enthalpy field, while M2 attempts to incorporate it.

Finally, consider a material which goes through both a solid-liquid phase transformation and then a solid-solid phase transformation. For a pure material (Stefan problem, isothermal solidification) with two phase transitions we have the following function:

$$\mathcal{T}(H) = \begin{cases} H/C_p, & H < C_pT_{m,2}; \\ T_{m,2}, & C_pT_{m,2} \leq H \leq C_pT_{m,2} + L_2; \\ (H - L_2)/C_p, & C_pT_{m,2} + L_2 \leq H \leq C_pT_{m,1} + L_2; \\ T_{m,1}, & C_pT_{m,1} + L_2 \leq H \leq C_pT_{m,1} + L_2 + L_1; \\ (H - L_2 - L_1)/C_p, & H > C_pT_{m,1} + L_1 + L_2. \end{cases}$$

Here $T_{m,1} > T_{m,2}$. Phase transition 1 occurs at temperature $T_{m,1}$ with latent heat

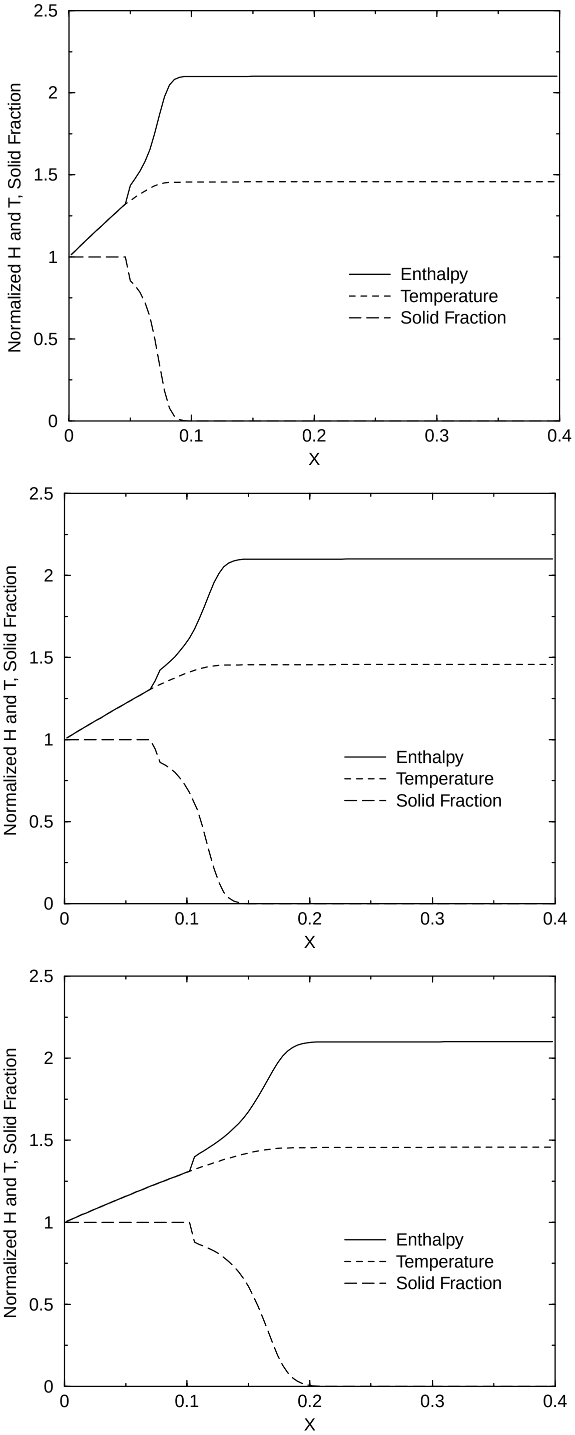


FIG. 1. *Normalized solutions of temperature, enthalpy, and solid fraction for the* 1*D binary alloy problem at* $t = 100$, 250, *and* 500.

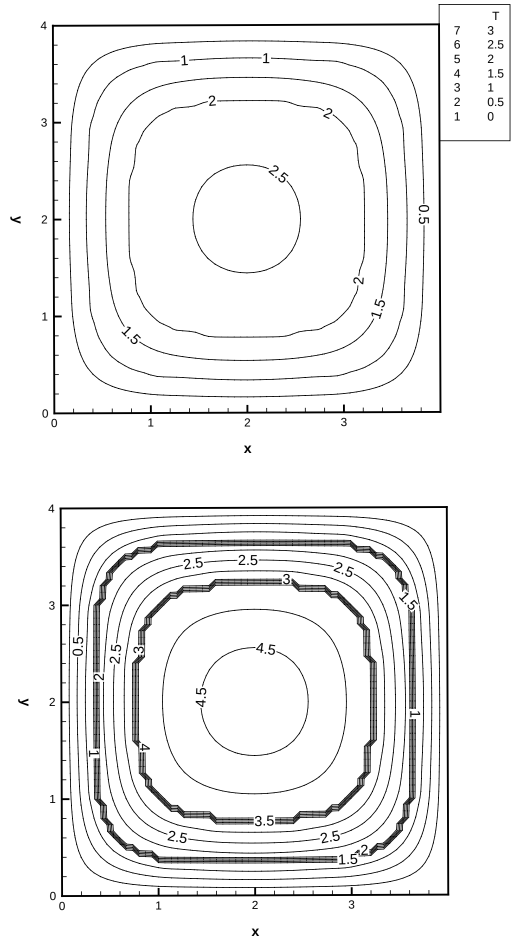


FIG. 2. *Solution of the two phase transition Stefan problem; temperature (top) and enthalpy (bottom).*

L_1. Simulation of this problem will result in the propagation of two sharp fronts. We consider a 2D model problem with an initial temperature of 3.0, $T_{m,1} = 2.0$, $T_{m,2} = 1.0$, $L_1 = L_2 = 1.0$, and $\kappa = C_p = \rho = 1.0$. At time equal to zero the boundary conditions are a zero applied temperature. Figure 2 depicts the temperature and enthalpy solution at time equal to 0.5. We see the two fronts in the enthalpy solution. The results of Table 4 show that the improved preconditioner (M2) makes a positive impact on this problem as well, decreasing the required CPU time by as much as a factor of two.

3.2. Prototype flow solver. In this subsection we demonstrate algorithmic performance of our distributed preconditioner on a thermally driven incompressible flow problem. We consider a time dependent version of the natural convection model problem in stream function (ψ), vorticity (ω), and temperature (T) form [Dav83]. Here the pressure-velocity coupling has been transfromed into a stream function–vorticity coupling. The boundary conditions on the ω equation are a function of ψ, and the source term for the ψ equation is ω. This is a stiff coupling, and our pre-

TABLE 4

Algorithm performance as a function of grid dimension and preconditioner on 2D two-phase transition Stefan problem; time = 0.5 and dt = 0.01.

Grid dim.	Number of time steps	Number of Newton its. per time step	GMRES its. per Newton it.	CPU time
30x30, M1	50	3.33	3.75	1.5
30x30, M2	50	2.76	1.5	1.0
60x60, M1	50	4.67	7.0	21.5
60x60, M2	50	3.67	2.0	10.9

conditioning strategy must contain an iteration on this coupling to be effective. The Newton–Krylov method is used for the solution of the fully coupled system (including implicit advection).

$$\nabla^2\psi = \omega, \tag{3.10}$$

$$\frac{\partial\omega}{\partial t} + \nabla\cdot(\vec{V}\omega) - \frac{1}{Re}\nabla^2\omega = \frac{Gr}{Re^2}\nabla_x T, \tag{3.11}$$

$$\frac{\partial T}{\partial t} + \nabla\cdot(\vec{V}T) - \frac{1}{Re\;Pr}\nabla^2 T = 0 \tag{3.12}$$

with $\vec{V} = u\hat{x} + v\hat{y}$, $u = \frac{\partial\psi}{\partial y}$, and $v = -\frac{\partial\psi}{\partial x}$. Here Re is the Reynolds number, Gr is the Grashof number, and Pr is the Prandtl number. As in [Dav83], we use $Re = 1$ and $Pr = 0.71$. Note that in this nondimensionalization the maximum velocity is proportional to Gr. Thus, for large Gr, convection will dominate diffusion even for $Re = 1$. We use centered differences in space and forward differences in time. Time step size will be such that the explicit limit based on advection is not severely violated. This choice of time step is motivated by time accuracy not stability. The preconditioner employs a Picard linearization for the advection terms in (3.11) and (3.12), and $\mathbf{P}^{-1}$ results from an approximate inverse of a segregated system. Note that (3.10) has no time derivative and is therefore purely elliptic. A low-complexity MG method [KR00, MKR00, KM00] is used to solve the separate (scalar) elliptic problems. Our preconditioning system is

$$\begin{bmatrix} P_{\psi\psi} & P_{\psi\omega} & 0 \\ P_{\omega\psi} & P_{\omega\omega} & P_{\omega T} \\ 0 & 0 & P_{TT} \end{bmatrix} \begin{bmatrix} \delta\psi \\ \delta\omega \\ \delta T \end{bmatrix} = - \begin{bmatrix} F_\psi \\ F_\omega \\ F_T \end{bmatrix}.$$

Here $P_{\psi\psi} = \nabla^2$, $P_{\psi\omega}$ is the source term coupling in (3.10), and $P_{\omega\psi}$ results from boundary condition on (3.11) being a function of ψ. The coupling from $P_{\psi\omega}$ and $P_{\omega\psi}$ can be represented as a 2 x 2 matrix at each finite volume. The two parabolic operators are

$$P_{TT} = \frac{I}{\Delta t} + \vec{V}^n\cdot\nabla - \frac{1}{Re\;Pr}\nabla^2$$

and

$$P_{\omega\omega} = \frac{I}{\Delta t} + \vec{V}^n\cdot\nabla - \frac{1}{Re}\nabla^2,$$

where $\vec{V}^n$ is the velocity at the previous Newton step.

The approximation to

$$\begin{bmatrix} \delta\psi \\ \delta\omega \\ \delta T \end{bmatrix} = -\mathbf{P}^{-1} \begin{bmatrix} F_\psi \\ F_\omega \\ F_T \end{bmatrix}$$

proceeds as follows.

Step 1. One MG V-cycle on

$$P_{TT}\delta T = -F_T.$$

Step 2. This step requires the solution of

$$\begin{bmatrix} P_{\psi\psi} & P_{\psi\omega} \\ P_{\omega\psi} & P_{\omega\omega} \end{bmatrix} \begin{bmatrix} \delta\psi \\ \delta\omega \end{bmatrix} = - \begin{bmatrix} F_\psi \\ F_\omega - P_{\omega T}\delta T \end{bmatrix},$$

which is similar to pressure-velocity coupling, since it has an elliptic constraint. With $\bar{F}_\omega = F_\omega - P_{\omega T}\delta T$, step 2 proceeds in three separate steps (a,b,c).

2a. Solve

$$\begin{bmatrix} 0 & P_{\psi\omega} \\ P_{\omega\psi} & 0 \end{bmatrix} \begin{bmatrix} \delta\psi^* \\ \delta\omega^* \end{bmatrix} = - \begin{bmatrix} F_\psi \\ \bar{F}_\omega \end{bmatrix},$$

which is a 2 × 2 matrix at each control volume.

2b. Next, approximately solve

$$P_{\omega\omega}\delta\omega^{**} = \bar{F}_\omega - P_{\omega\psi}\delta\psi^*,$$

$$P_{\psi\psi}\delta\psi^{**} = F_\psi - P_{\psi\omega}\delta\omega^*,$$

which requires two separate MG solves, each only one V-cycle.

2c. Finally, solve

$$\begin{bmatrix} 0 & P_{\psi\omega} \\ P_{\omega\psi} & 0 \end{bmatrix} \begin{bmatrix} \delta\psi \\ \delta\omega \end{bmatrix} = - \begin{bmatrix} F_\psi - P_{\psi\psi}\delta\psi^{**} \\ \bar{F}_\omega - P_{\omega\omega}\delta\omega^{**} \end{bmatrix},$$

which is again a 2 x 2 matrix at each control volume. Thus approximating the action of $\mathbf{P}^{-1}$ on a vector requires three separate (scalar) MG V-cycles and two sweeps of a block 2 × 2 matrix on the fine grid. The 2 × 2 matrix solves which incorporate the stiff stream function–vorticity coupling in the preconditioner (and their placement) is crucial to the performance of this approach. As in [KR00, MKR00], the MG here uses piecewise constant restriction and prolongation and a Galerkin coarse grid operator. SGS is used as the smoother. We have also used a coupled MG method for this problem. Here the smoother is a block SGS, where the block is a 3 × 3 matrix coupling the unknowns at a cell [KR00, KM00].

Our model problem is a fixed number of time steps for a simulation with $Gr = 1.0 \times 10^6$. The initial conditions are a steady-state solution for $Gr = 1.0 \times 10^3$. We take a fixed number of constant time steps and collect performance data. Since we wish to have the same advective Courant number $(u\frac{\Delta t}{\Delta x})$ on each grid, the time step is reduced as the grid is refined. Figure 3 shows the steady-state temperature for $Gr = 1.0 \times 10^3$ and $Gr = 1.0 \times 10^6$.

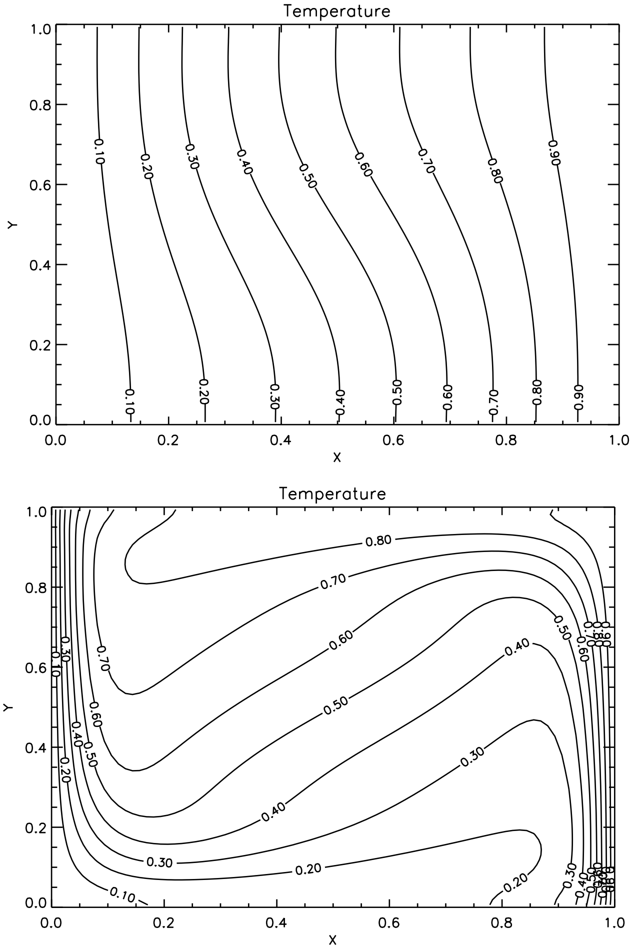


FIG. 3. *Temperature solution of natural convection problem; Gr* $= 1.0 \times 10^3$ *(top) and* 1.0×10^6 *(bottom).*

TABLE 5

Algorithm performance as a function of preconditioner on the 2*D natural convection problem;* 40×40 *grid,* $dt = 1.0e-4$*,* $time = 2.0e-3$ *(*20 *time steps).*

Precond. method	Number of time steps	Number of Newton its. per time step	GMRES its. per Newton it.	CPU time norm.
MG dist.	20	3.9	2.5	1.0
MG coup.	20	3.4	2.4	1.1
ILU(0)	20	4.0	37.0	2.8

The performance on 40 × 40, 80 × 80, and 160 × 160 grids is given in Tables 5 through 7. In the nondimensionalization of this problem the maximum "effective" Reynolds number is equivalent to the peak velocity. For the problem considered this is approximately 1000 at the end of the simulation. Both algorithmic and CPU scalings are good for the MG methods. The coupled MG results in the fewest Krylov iterations per Newton iteration, while the segregated MG method produces the lowest

Table 6

Algorithm performance as a function of preconditioner on the 2*D natural convection problem;* 80×80 *grid,* $dt = 5.0e - 5$, *time* $= 2.0e - 3$ (40 *time steps).*

Precond. method	Number of time steps	Number of Newton its. per time step	GMRES its. per Newton it.	CPU time norm.
MG dist.	40	3.0	2.8	6.2
MG coup.	40	3.2	2.5	8.1
ILU(0)	40	4.0	78.0	65.1

Table 7

Algorithm performance as a function of preconditioner on the 2*D natural convection problem;* 160×160 *grid,* $dt = 2.5e - 5$, *time* $= 2.0e - 3$ (80 *time steps).*

Precond. method	Number of time steps	Number of Newton its. per time step	GMRES its. per Newton it.	CPU time norm.
MG dist.	80	3.0	3.1	57.5
MG coup.	80	3.6	2.8	94.6

Table 8

Algorithm performance as a function of preconditioner on the 2*D natural convection problem;* 80×80 *grid,* $dt = 2.0e - 4$, *time* $= 2.0e - 3$ (10 *time steps).*

Precond. method	Number of time steps	Number of Newton its. per time step	GMRES its. per Newton it.	CPU time norm.
MG dist.	10	4.0	4.4	2.7
MG coup.	10	4.0	3.6	3.2

CPU times. Both MG methods outperform an ILU(0) preconditioner. Note that both the coupled MG and ILU(0) preconditioners are approximately inverting a $3N \times 3N$ matrix, while the segregated MG method is approximately inverting three $N \times N$ matrices. For interest, Table 8 presents data for the 80×80 grid using a time step which is four times larger than that of Table 6. We can see that the CPU savings, for using a time step which is four times larger, are slightly better than a factor of two. We mention that this segregated preconditioner has previously been shown to perform well for a steady-state version for this problem with $Gr = 1.0 \times 10^7$ on a 320×320 grid [KM00] (a maximum Reynolds number of approximately 5000). Also, results in [PT01] have shown a similar segregated preconditioner to perform well on the lid-driven cavity problem at a Reynolds number of 10,000.

4. Conclusions. We have presented a model set of equations for solidifying flow and have presented a JFNK method for the solution of these equations. We have discussed a segregated solution procedure as a preconditioner, and we have studied specific issues of algorithmic performance on the phase-change heat conduction problem and thermally driven incompressible flow separately. These results will form the foundation of a preconditioning strategy for the fully coupled problem

In the phase-change heat conduction problem, a preconditioner which incorporated the effective heat capacity method was shown to be superior to a preconditioner which only used heat conduction. Comparing Tables 2 and 3 for the 60×60 grid, the effective heat capacity preconditioner reduced the required GMRES iterations by a factor of five and reduced the CPU time by a factor of four. Additionally, the JFNK method was shown to perform well on a problem which contained two phase transitions.

In the thermally driven incompressible flow problem, a segregated preconditioner using a low-complexity MG on each equation was presented. The relationship to other block triangular preconditioners for the Navier–Stokes equations was briefly discussed. The segregated preconditioner was compared to a coupled MG based preconditioner. The segregated preconditioner was shown to be competitive in terms of outer GMRES iterations while being superior in terms of CPU perfromance. The CPU advantage of the segregated preconditioner was shown to grow with increasing problem size, growing to nearly a factor of two on the 160 × 160 grid.

Acknowledgment. The authors would like to thank Mike Pernice for his discussions on block triangular preconditioners for the Navier–Stokes equations.

REFERENCES

[BS90] P. N. Brown and Y. Saad, *Hybrid Krylov methods for nonlinear systems of equations*, SIAM J. Sci. Statist. Comput., 11 (1990), pp. 450–481.

[CJ84] T. F. Chan and K. R. Jackson, *Nonlinearly preconditioned Krylov subspace methods for discrete Newton algorithms*, SIAM J. Sci. Statist. Comput., 5 (1984), pp. 533–542.

[Dan89] J. A. Dantzig, *Modelling liquid-solid phase change with melt convection*, Internat. J. Numer. Methods Engrg., 28 (1989), pp. 1769–1785.

[Dav83] G. De Vahl Davis, *Natural convection of air in a square cavity: a benchmark numerical solution*, Internat. J. Numer. Methods Fluids, 3 (1983), p. 249.

[ES96] H. Elman and D. Silvester, *Fast nonsymmetric iterations and preconditioning for Navier–Stokes equations*, SIAM J. Sci. Comput., 17 (1996), pp. 33–46.

[KKL99] D. A. Knoll, D. B. Kothe, and B. Lally, *A new nonlinear solution method for phase-change problems*, Numer. Heat Transfer, Part B, 35 (1999), pp. 439–459.

[Kla98] A. Klawonn, *Block-triangular preconditioners for saddle point problems with a penalty term*, SIAM J. Sci. Comput., 19 (1998), pp. 172–184.

[KM00] D. A. Knoll and V. A. Mousseau, *On Newton–Krylov multigrid methods for the incompressible Navier–Stokes equations*, J. Comput. Phys., 163 (2000), pp. 262–267.

[KMK96] D. A. Knoll, P. R. McHugh, and D. E. Keyes, *Newton–Krylov methods for low Mach number compressible combustion*, AIAA J., 34 (1996), pp. 961–967.

[Kno98] D. A. Knoll, *An improved convection scheme applied to recombining divertor plasma flows*, J. Comput. Phys., 142 (1998), pp. 473–488.

[KR90] C. T. Kelley and J. Rulla, *Solution of the time discretized Stefan problem by Newton's method*, Nonlinear Anal., 14 (1990), pp. 851–872.

[KR00] D. A. Knoll and W. J. Rider, *A multigrid preconditioned Newton–Krylov method*, SIAM J. Sci. Comput., 21 (1999), pp. 691–710.

[MKR00] V. A. Mousseau, D. A. Knoll, and W. J. Rider, *Physics-based preconditioning and the Newton–Krylov method for non-equilibrium radiation diffusion*, J. Comput. Phys., 160 (2000), pp. 743–765.

[Pat80] S. V. Patankar, *Numerical Heat Transfer and Fluid Flow*, Hemisphere Publishing Corp., New York, 1980.

[PT01] M. Pernice and M. D. Tocci, *A multigrid-preconditioned Newton–Krylov method for the incompressible Navier–Stokes equations*, SIAM J. Sci. Comput., 23 (2001), pp. 398–418.

[RK99] W. J. Rider and D. A. Knoll, *Time step size selection for radiation diffusion calculations*, J. Comput. Phys., 152 (1999), pp. 790–795.

[SS86] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[SV96] C. R. Swaminathan and V. R. Voller, *Towards a general numerical scheme for solidification systems*, Int. J. Heat Mass Transfer, 40 (1996), p. 2859.

[VCM87] V. R. Voller, M. Cross, and C. Markatos, *An enthalpy method for convection diffusion phase change*, Internat. J. Numer. Methods Engrg., 24 (1987), pp. 271–284.

[Wes92] P. Wesseling, *An Introduction to Multigrid Methods*, John Wiley, Chichester, UK, 1992.

SIAM J. SCI. COMPUT.
Vol. 23, No. 2, pp. 398–418

A MULTIGRID-PRECONDITIONED NEWTON–KRYLOV METHOD FOR THE INCOMPRESSIBLE NAVIER–STOKES EQUATIONS[*]

M. PERNICE[†] AND M. D. TOCCI[‡]

Abstract. Globalized inexact Newton methods are well suited for solving large-scale systems of nonlinear equations. When combined with a Krylov iterative method, an explicit Jacobian is never needed, and the resulting matrix-free Newton–Krylov method greatly simplifies application of the method to complex problems. Despite asymptotically superlinear rates of convergence, the overall efficiency of a Newton–Krylov solver is determined by the preconditioner. High-quality preconditioners can be constructed from methods that incorporate problem-specific information, and for the incompressible Navier–Stokes equations, classical pressure-correction methods such as SIMPLE and SIMPLER fulfill this requirement. A preconditioner is constructed by using these pressure-correction methods as smoothers in a linear multigrid procedure. The effectiveness of the resulting Newton–Krylov-multigrid method is demonstrated on benchmark incompressible flow problems.

Key words. Newton–Krylov methods, multigrid preconditioning, pressure-correction smoothers

AMS subject classifications. 65H10, 65F10, 65N55

PII. S1064827500372250

1. Introduction.

Efficient solution of the steady-state incompressible Navier–Stokes equations

$$\begin{aligned} (uu)_x + (uv)_y - \tfrac{1}{Re}\Delta u + p_x &= f_1, \\ (uv)_x + (vv)_y - \tfrac{1}{Re}\Delta v + p_y &= f_2, \\ u_x + v_y &= 0, \end{aligned} \tag{1.1}$$

where Re is the Reynolds number, has been a problem of central importance in computational science and engineering since its inception. Methods for solving (1.1) can also be leveraged to solve the systems of nonlinear equations that arise when an implicit method is used to solve the unsteady incompressible Navier–Stokes equations. Early efforts were constrained by limited memory capacity and were often based on strategies that involved solving a series of simplified and lower-dimensional problems. The simplifications generally arose from application of operator-splitting strategies, and solutions of individual equations were often built from iterative methods based on line solves. Pressure-correction algorithms such as SIMPLE [29] and SIMPLER [28] are typical of these approaches. While frugal in their use of memory, these methods lack theoretical foundations and converge slowly. Moreover, these convergence rates degrade with increasing problem size. Despite these drawbacks, pressure-correction methods are still in widespread use in production engineering

[*] Received by the editors May 18, 2000; accepted for publication (in revised form) November 29, 2000; published electronically July 10, 2001. This work was supported by the University of Utah Center for the Simulation of Accidental Fires and Explosions and was funded by the Department of Energy, Lawrence Livermore National Laboratory, under subcontract B341493.

http://www.siam.org/journals/sisc/23-2/37225.html

[†] Computer and Computational Sciences Division, Los Alamos National Laboratory, Los Alamos, NM 87545 (pernice@lanl.gov). This author was also supported by Reaction Engineering International, Salt Lake City, UT, through funding from the Department of Energy Office of Energy Research under grant DE–FG03–96ER82268, and by the University of Utah Center for High Performance Computing.

[‡] The MathWorks, Inc., Natick, MA 01760 (mtocci@mathworks.com).

codes. Consequently, there is great interest in exploring strategies for accelerating the convergence of pressure-correction methods while minimizing changes in software and data structures.

Recent decades have seen considerable progress in the development of new approaches for solving large-scale nonlinear problems, in particular, globalized inexact Newton methods [11, 12, 13]. The major shortcoming of classical Newton methods is the need to solve a large-scale system of linear equations at each iteration. Inexact Newton methods address this by using an iterative method to solve this system of equations to less than full accuracy. Inexact Newton methods can be considerably more expensive than the classical pressure-correction techniques, but rapid increases in both the speed and capacity of computing resources make it practical to consider inexact Newton methods for incompressible flow problems. At the same time, the desire to address new classes of problems sometimes makes it necessary to consider these methods.

Since their introduction, inexact Newton methods have been successfully applied to a large variety of nonlinear problems: integral equation descriptions of equilibrium states of liquid surfaces [3], multiphase flow in porous media [9, 43], radiation-diffusion problems [32, 27, 5], reacting flows [20, 21, 35], aerodynamics calculations [7, 6], and incompressible flow problems [25, 19, 22, 23]. While by no means exhaustive, these applications represent a cross-section of preconditioning strategies currently in use. Incomplete LU (ILU) factorizations [26, 33] are popular choices, but they require information about the Jacobian that may be difficult to determine in a matrix-free inexact Newton method. Depending on the amount of element fill-in that is allowed, ILU preconditioners can have high storage requirements. Calculating an ILU preconditioner can also be computationally expensive, and this cost is multiplied by the number of times the preconditioner is updated during the nonlinear solution process.

A notable trend reflected in this list of applications of Newton–Krylov methods is the use of multigrid methods as preconditioners [43, 32, 5, 22, 23]. While multigrid methods are highly efficient solvers on their own, they also serve as excellent preconditioners, and their use in this context makes the performance and robustness of the multigrid method less sensitive to the selection of components such as intergrid transfers and coarse grid solvers. Multigrid methods can be tailored to specific problems by selection of an appropriate smoother; it has been known for some time that SIMPLE can be used as a multigrid smoother [36, 37], and it has recently been demonstrated that SIMPLER can also be used as a smoother in a multigrid procedure [30]. In contrast to prior work on Newton–Krylov-multigrid (NK-MG) methods [22, 23], pressure-correction methods work directly on the primitive variable formulation (1.1) and so are readily extendible to problems in three dimensions. Also in contrast to prior work applying Newton–Krylov methods to incompressible flow problems [25], a multigrid preconditioner represents a considerable savings in storage and setup time over an ILU-based preconditioner. The combination of an NK-MG method with a pressure-correction smoother may also be regarded as a means for accelerating the convergence of the pressure-correction scheme; however, patterns of use of the pressure-correction methods must be reexamined in order to use them effectively in this context.

This paper is organized as follows. First, some notation is introduced. Following this, the relevant algorithmic components are described in section 3. The effectiveness of combining Newton–Krylov methods with multigrid preconditioners equipped with pressure-correction smoothers is examined in section 4. These results are summarized and some conclusions are drawn in section 5.

2. Notation. Pressure-correction methods have traditionally been developed for staggered grid discretizations [29, 28], and that convention is adhered to in this work. In this arrangement of variables, originally presented in [18], the domain is divided into a number of cells, with the horizontal component of velocity centered on vertical cell faces, vertical components of velocity centered on horizontal cell faces, and pressure located at cell centers. A second-order centered discretization of the Navier–Stokes equations (1.1) on a staggered grid produces a set of nonlinear equations $F(\mathbf{u}, p) = 0$, which may be written in block matrix form as

$$F(\mathbf{u}, p) = \mathcal{Q}[\mathbf{u}] \begin{pmatrix} \mathbf{u} \\ p \end{pmatrix} - \begin{pmatrix} \mathbf{f} \\ 0 \end{pmatrix}, \tag{2.1}$$

where

$$\mathcal{Q}[\mathbf{u}] = \begin{pmatrix} \mathbf{Q}[\mathbf{u}] & \nabla^h \\ \nabla^h\cdot & 0 \end{pmatrix}.$$

In this,

$$\mathbf{Q}[\mathbf{u}] = \begin{pmatrix} Q_1[\mathbf{u}] & 0 \\ 0 & Q_2[\mathbf{u}] \end{pmatrix} \tag{2.2}$$

is the discrete momentum operator, ∇^h is a discrete gradient operator, $\nabla^h\cdot$ is a discrete divergence operator,

$$\mathbf{u} = \begin{pmatrix} u \\ v \end{pmatrix}$$

is the discrete velocity field, and

$$\mathbf{f} = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix}$$

represents the body force.

Implicit methods for solving the time dependent version of the Navier–Stokes equations can be accommodated with a few minor modifications. If a backward Euler method is used to discretize the time derivative, the discrete momentum operator becomes

$$\mathbf{Q}[\mathbf{u}] = \begin{pmatrix} Q_1[\mathbf{u}] + u/\Delta t & 0 \\ 0 & Q_2[\mathbf{u}] + v/\Delta t \end{pmatrix},$$

and the set of nonlinear equations to be solved at each time step is

$$F(\mathbf{u}, p) = \mathcal{Q}[\mathbf{u}] \begin{pmatrix} \mathbf{u} \\ p \end{pmatrix} - \frac{1}{\Delta t} \begin{pmatrix} \mathbf{u}^n \\ 0 \end{pmatrix} - \begin{pmatrix} \mathbf{f} \\ 0 \end{pmatrix},$$

where $\mathbf{u}^n$ is the velocity at the previous time step.

3. Algorithms. This section describes the basic ingredients that are used to construct the NK-MG method. The discussion begins with a description of the inexact Newton method that will be used. This is followed by a discussion of classical pressure-correction methods and their use as smoothers in the multigrid preconditioner.

3.1. Inexact Newton methods. Newton's method for solving a system of nonlinear equations

$$F(x) = 0$$

requires, at the kth step, the solution of the linear *Newton equation*

$$F'(x_k)\, s_k = -F(x_k), \tag{3.1}$$

where x_k is the current approximate solution and F' is the Jacobian matrix of the system. Once the Newton step s_k is determined, the current approximation is updated via

$$x_{k+1} = x_k + s_k.$$

This process is continued until a satisfactory solution is found, which is usually judged by making $\|F(x_k)\|$ or $\|s_k\|$ (or both) sufficiently small. Traditionally, Newton's method was considered to be inappropriate for the solution of large-scale systems of nonlinear equations because of the high computational and storage costs of solving (3.1). However, Newton's method will still converge even if (3.1) is not solved exactly; indeed, under some circumstances an exact solution of (3.1) is undesirable. This can be seen by noting that (3.1) is obtained from a linearization of $F(x_k) = 0$ around x_k, which is valid only in a neighborhood of x_k. If the solution of (3.1) produces an s_k that is too large, there may be poor agreement between F and its local linear model, and the effort expended to accurately compute s_k may be wasted. Examples that illustrate this behavior appear in [13, 35].

Newton iterative methods relax the requirement to solve (3.1) exactly to an *inexact Newton condition* [11]

$$\|F(x_k) + F'(x_k)\, s_k\| \le \eta_k \|F(x_k)\|, \tag{3.2}$$

in which the "forcing term" $\eta_k \in (0, 1)$ can be specified either statically or dynamically [13] to enhance efficiency and convergence. The optimal choice of η_k is somewhat problem-specific; see, for example, [35] for some comparative studies. Nevertheless, superlinear and even quadratic convergence of the inexact Newton method can be obtained under certain choices of the forcing terms [11, 13]. In this work, an initial η_k is dynamically determined using a slight modification of the strategy from [13] labeled Choice 1. There, safeguards that prevent η_k from getting too small too quickly are disabled once η_k drops below a certain threshold. In this work, these safeguards are retained throughout the computation.

There are many ways to compute an inexact Newton step s_k that satisfies (3.2), and the efficiency of an inexact Newton method is strongly affected by this choice. Krylov subspace methods [15] are especially well suited for this purpose since they require only matrix-vector products $F'(x_k)v$. This further specialization of inexact Newton methods leads to the class of methods referred to as *Newton–Krylov methods.* The matrix-vector products needed in a Newton–Krylov method may be approximated with finite differences of function values

$$F'(x_k)v \approx \frac{F(x_k + \epsilon v) - F(x_k)}{\epsilon}, \tag{3.3}$$

and so the Jacobian F' never needs to be explicitly formed. Because the Jacobian matrix is neither formed nor stored, this approach is frequently referred to as a *matrix-free Newton–Krylov* method. While this greatly reduces storage requirements and simplifies implementation, the differencing parameter ϵ must be chosen carefully. Furthermore, some information about the Jacobian is still needed to construct a preconditioner. Finally, a matrix-free Newton–Krylov method generally requires more nonlinear iterations than a Newton–Krylov method that uses the Jacobian directly.

Among Krylov subspace methods, GMRES [34] is generally preferred, since it minimizes the residual at every iteration. Unfortunately, in order to enforce this, the work and storage requirements per iteration grow linearly with the number of iterations. In practice, this is dealt with by restarting the method, which can potentially slow down convergence or even cause divergence if restarting is done too frequently. Alternatives such as BiCGSTAB [38] and transpose-free QMR [14] can be used; however, they do not share the minimum residual property and, when used in a matrix-free Newton–Krylov method, are generally not as robust as GMRES [24], provided a sufficiently large restart value is used.

Finally, another traditional objection to using Newton's method for large-scale problems is the need to find a good initial approximation x_0. Newton's method (and its inexact counterpart) can fail to converge if x_0 is not chosen carefully. Fortunately, classical strategies for improving the likelihood of convergence from a poor initial approximation also apply to inexact Newton methods [12]. The backtracking globalization strategy given in Algorithm Inexact Newton Backtracking (INB) from [12] is employed in this work.

ALGORITHM 3.1 (INB [12]).

```
Let $x_0$, $\eta_{\max} \in [0,1)$, $t \in (0,1)$, and $0 < \theta_{\min} < \theta_{\max} < 1$ be given.
For $k = 0, 1, \ldots$ (until convergence) do:
    Choose initial $\eta_k \in [0, \eta_{\max}]$ and $s_k$ such that
        $\|F(x_k) + F'(x_k)\, s_k\| \le \eta_k \|F(x_k)\|$.
    While $\|F(x_k + s_k)\| > [1 - t(1 - \eta_k)]\, \|F(x_k)\|$ do:
        Choose $\theta \in [\theta_{\min}, \theta_{\max}]$.
        Update $s_k \longleftarrow \theta s_k$ and $\eta_k \longleftarrow 1 - \theta(1 - \eta_k)$.
    Set $x_{k+1} = x_k + s_k$.
```

In this, backtracking is invoked when the trial Newton iterate $x_k + s_k$ fails to adequately reduce $\|F\|$. This is determined by comparing the *actual reduction* in $\|F\|$,

$$\|F(x_k)\| - \|F(x_k + s_k)\|,$$

with the *predicted reduction*

$$\|F(x_k)\| - \eta_k \|F(x_k)\| = (1 - \eta_k)\|F(x_k)\|.$$

If the actual reduction exceeds some fraction t of the predicted reduction, then the step is accepted. The value $t = 10^{-4}$ is used to judge sufficient reduction, leading to acceptance of a step that produces even minimal reduction in $\|F\|$. If this criterion is not met, the step is damped by a factor θ that is chosen to minimize a quadratic that interpolates $\|F\|$ in the direction of s_k. Backtracking safeguard values $\theta_{\min}$ and $\theta_{\max}$ provide bounds on the step reduction and are given by 0.1 and 0.5, respectively. Once the Newton step is damped, the predicted reduction in $\|F\|$ is then approximated by noting

$$\begin{aligned}\|F(x_k+\theta s_k)\| &\approx \|F(x_k)+\theta F'(x_k)s_k\| \\ &\leq \theta\|F(x_k)+F'(x_k)s_k\|+(1-\theta)\|F(x_k)\| \\ &\leq (1-\theta(1-\eta_k))\|F(x_k)\|.\end{aligned}$$

This leads to the adjustment in the forcing term indicated in Algorithm INB, which is used either in the next pass through the backtracking loop or as an initial value in the selection of the forcing term for the next Newton iteration.

3.2. Pressure-correction methods. The pressure-correction methods SIMPLE and SIMPLER can be used to incorporate information that is specific to the incompressible Navier–Stokes equations into the preconditioner. These methods are described in this section. In the following, the dependence of the momentum transport operator (2.2) on $\mathbf{u}$ is suppressed.

3.2.1. SIMPLE. The SIMPLE algorithm, introduced in [29], begins by approximately solving the momentum equations, and then uses the discretized form of the continuity equation to derive an equation whose solution is used both to update the pressure field and to correct the velocity field so that mass is conserved. To begin with, the discrete momentum transport operator $\mathbf{Q}$ is updated to reflect the current approximate solution $\mathbf{u}^{(n)}$ to the momentum equations. The resulting linearized momentum equations can then be solved to determine an intermediate velocity field $\mathbf{u}^{(n+\frac{1}{2})}$ using the current approximate pressure field $p^{(n)}$:

$$\mathbf{Q}\mathbf{u}^{(n+\frac{1}{2})} = \mathbf{f} - \nabla^h p^{(n)}. \tag{3.4}$$

The resulting velocity field $\mathbf{u}^{(n+\frac{1}{2})}$ does not satisfy the continuity equation, and the residual of this equation is used to compute a correction δp to the pressure field whose gradient is also used to correct $\mathbf{u}^{(n+\frac{1}{2})}$.

To derive an equation for δp, let

$$\mathbf{D} = \operatorname{diag}\mathbf{Q}, \tag{3.5}$$

and introduce the approximations

$$\begin{array}{ccccc}\mathbf{u}^{(n+\frac{1}{2})} & \approx & \mathbf{D}^{-1}\mathbf{Q}\mathbf{u}^{(n+\frac{1}{2})} & = & \mathbf{D}^{-1}(\mathbf{f}-\nabla^h p^{(n)}), \\ \mathbf{u}^{(n+1)} & \approx & \mathbf{D}^{-1}\mathbf{Q}\mathbf{u}^{(n+1)} & = & \mathbf{D}^{-1}(\mathbf{f}-\nabla^h p^{(n+1)}).\end{array} \tag{3.6}$$

Subtracting these equations and changing the approximation to equality leads to

$$\delta\mathbf{u} \equiv \mathbf{u}^{(n+1)} - \mathbf{u}^{(n+\frac{1}{2})} = -\mathbf{D}^{-1}\nabla^h\delta p, \tag{3.7}$$

where $\delta p = p^{(n+1)} - p^{(n)}$. Applying the discrete divergence operator to these equations gives

$$\nabla^h\cdot(\mathbf{u}^{(n+1)} - \mathbf{u}^{(n+\frac{1}{2})}) = \nabla^h\cdot\mathbf{D}^{-1}\nabla^h\delta p.$$

Finally, requiring $\nabla^h\cdot\mathbf{u}^{(n+1)} = 0$ leads to

$$S\delta p = \nabla^h\cdot\mathbf{u}^{(n+\frac{1}{2})}, \tag{3.8}$$

where

$$S = -\nabla^h\cdot\mathbf{D}^{-1}\nabla^h \tag{3.9}$$

is symmetric. Equation (3.8) is a generalized Poisson equation which must be solved for the pressure correction δp. Since the intended correction (3.7) should be 0 at locations where the velocity field is specified, (3.8) is supplemented with homogeneous boundary conditions at these locations. In particular, for problems where the velocity field is specified at the boundaries, δp is determined only up to an additive constant, and S is positive semidefinite. Summarizing, we have the following algorithm.

ALGORITHM 3.2 (SIMPLE).

Determine $\mathbf{u}^{(n+\frac{1}{2})}$ *by solving* (3.4).

Find the pressure correction δp *from* (3.8).

Calculate the velocity corrections $\delta\mathbf{u}$ *using* (3.7).

Update the pressure

$$p^{(n+1)} = p^{(n)} + \delta p$$

and the velocities

$$\mathbf{u}^{(n+1)} = \mathbf{u}^{(n+\frac{1}{2})} + \delta\mathbf{u}.$$

Remark 3.1. In practice the pressure update is damped by a factor $\alpha \in (0,1]$:

$$p^{(n+1)} = p^{(n)} + \alpha\delta p.$$

Remark 3.2. SIMPLEC is a variation that replaces the entries in the diagonal matrices D_i with absolute rowsums from Q_i [39]. This variation is used in this work.

Remark 3.3. Observe that

$$\begin{pmatrix} \mathbf{u}^{(n+1)} \\ p^{(n+1)} \end{pmatrix} = \begin{pmatrix} \mathbf{u}^{(n)} \\ p^{(n)} \end{pmatrix} + \begin{pmatrix} \mathbb{I} & -\mathbf{D}^{-1}\nabla^h \\ 0 & \alpha\mathbb{I} \end{pmatrix} \begin{pmatrix} \mathbf{Q} & 0 \\ \nabla^h\cdot & S \end{pmatrix}^{-1} \mathbf{r}^{(n)}, \tag{3.10}$$

where $\mathbb{I}$ is an identity operator of the appropriate dimension and

$$\mathbf{r}^{(n)} = \begin{pmatrix} \mathbf{f} \\ 0 \end{pmatrix} - \mathcal{Q}\begin{pmatrix} \mathbf{u}^{(n)} \\ p^{(n)} \end{pmatrix}$$

is the residual of the system. This representation can be derived by employing the notion of transforming smoothers; see [42, 41].

Remark 3.4. When SIMPLE is used as a solver, the inversion in (3.10) is not computed exactly. This practice is followed when using SIMPLE as a smoother: the momentum equations are approximately solved with five sweeps of the point Gauss–Seidel method, and the pressure correction equation is solved with twenty sweeps of point Gauss–Seidel. These values were empirically determined to strike a good balance between cost and effectiveness of the preconditioner.

3.2.2. SIMPLER. SIMPLER is a variation due to Patankar [28]. It is similar to SIMPLE, but it determines $p^{(n+1)}$ from $\mathbf{u}^{(n)}$ and uses a separate potential field ϕ to enforce continuity in a manner similar to projection methods [8, 2].

As with SIMPLE, each cycle begins with an update of the momentum transport operator (2.2) to reflect the latest approximate solution $\mathbf{u}^{(n)}$. For the next iteration the pressure and velocity field should satisfy

$$\mathbf{Q}\mathbf{u}^{(n+1)} = \mathbf{f} - \nabla^h p^{(n+1)}.$$

To determine an equation for $p^{(n+1)}$, introduce the splitting $\mathbf{Q} = \mathbf{D} - (\mathbf{L} + \mathbf{U})$, where $\mathbf{D}$ is again given by (3.5) and $-(\mathbf{L} + \mathbf{U})$ are the off-diagonal elements of $\mathbf{Q}$. It follows that

$$\begin{aligned} \mathbf{u}^{(n+1)} &= \mathbf{D}^{-1}(\mathbf{f} + (\mathbf{L} + \mathbf{U})\mathbf{u}^{(n+1)} - \nabla^h p^{(n+1)}) \\ &\approx \mathbf{D}^{-1}(\mathbf{f} + (\mathbf{L} + \mathbf{U})\mathbf{u}^{(n)} - \nabla^h p^{(n+1)}). \end{aligned} \tag{3.11}$$

Taking the divergence of both sides of (3.11), requiring that $\nabla^h \cdot \mathbf{u}^{(n+1)} = 0$, changing the approximation to equality, and rearranging terms lead to

$$Sp^{(n+1)} = -\nabla^h \cdot \mathbf{D}^{-1}(\mathbf{f} + (\mathbf{L} + \mathbf{U})\mathbf{u}^{(n)}), \tag{3.12}$$

where S is again given by (3.9).

Once the updated pressure field is known, the updated velocity field is found by first solving

$$\mathbf{Q}\mathbf{u}^{(n+\frac{1}{2})} = \mathbf{f} - \nabla^h p^{(n+1)} \tag{3.13}$$

and then correcting $\mathbf{u}^{(n+\frac{1}{2})}$ so that the updated velocity field conserves mass. This is done using the gradient of an auxiliary variable ϕ. To determine an equation for ϕ, the following *ansatz* is made for the correction:

$$\mathbf{u}^{(n+1)} = \mathbf{u}^{(n+\frac{1}{2})} - \mathbf{D}^{-1}\nabla^h \phi. \tag{3.14}$$

An equation for ϕ is then determined by requiring that $\nabla^h \cdot \mathbf{u}^{(n+1)} = 0$:

$$\begin{aligned} 0 &= \nabla^h \cdot \mathbf{u}^{(n+1)} \\ &= \nabla^h \cdot \mathbf{u}^{(n+\frac{1}{2})} + S\phi, \end{aligned} \tag{3.15}$$

where S is again given by (3.9). Summarizing, we have the following algorithm.

Algorithm 3.3 (SIMPLER).
Determine $p^{(n+1)}$ by solving (3.12).
Determine $\mathbf{u}^{(n+\frac{1}{2})}$ by solving (3.13).
Determine ϕ by solving (3.15).
Correct $\mathbf{u}^{(n+\frac{1}{2})}$ using (3.14).

Remark 3.5. The correction (3.14) may be described in terms of a projection:

$$\begin{aligned} \mathbf{u}^{(n+1)} &= \mathbf{u}^{(n+\frac{1}{2})} + \mathbf{D}^{-1}\nabla^h S^{-1}\nabla^h \cdot \mathbf{u}^{(n+\frac{1}{2})} \\ &= \left(\mathbb{I} + \mathbf{D}^{-1}\nabla^h S^{-1}\nabla^h \cdot\right)\mathbf{u}^{(n+\frac{1}{2})} \\ &\equiv \mathbb{P}\mathbf{u}^{(n+\frac{1}{2})}. \end{aligned}$$

From (3.9) it follows that $\mathbb{P}^2 = \mathbb{P}$. Thus $\mathbb{P}$ is actually a *projection* (though it is not an *orthogonal* projection with respect to the standard inner product).

3.2.3. Alternative description. In order to employ SIMPLER as either a preconditioner or a smoother in a linear multigrid method, the method must be recast in a form that operates on residuals, analogous to (3.10). To derive such a representation, observe that

$$\begin{pmatrix} \mathbf{u}^{(n+\frac{1}{2})} \\ p^{(n+1)} \end{pmatrix} = \mathcal{M}^{-1} \begin{pmatrix} \mathbf{f} \\ -\nabla^h \cdot \mathbf{D}^{-1}(\mathbf{f} + (\mathbf{L} + \mathbf{U})\mathbf{u}^{(n)}) \end{pmatrix}, \tag{3.16}$$

where

$$\mathcal{M} = \begin{pmatrix} \mathbf{Q} & \nabla^h \\ 0 & S \end{pmatrix}.$$

The residual $\mathbf{r}^{(n)}$ may be partitioned into components r_1, r_2 that correspond, respectively, to the momentum and continuity equations. Then note that

$$\mathbf{Q}\mathbf{u}^{(n)} + \nabla^h p^{(n)} + r_1 = \mathbf{f}$$

so that

$$\mathbf{D}^{-1}(\mathbf{f} + (\mathbf{L} + \mathbf{U})\mathbf{u}^{(n)}) = \mathbf{u}^{(n)} + \mathbf{D}^{-1}\nabla^h p^{(n)} + \mathbf{D}^{-1} r_1.$$

Thus (3.16) may be rewritten as

$$\begin{aligned} \begin{pmatrix} \mathbf{u}^{(n+\frac{1}{2})} \\ p^{(n+1)} \end{pmatrix} &= \mathcal{M}^{-1}\left[\begin{pmatrix} \mathbf{f} \\ 0 \end{pmatrix} - \begin{pmatrix} 0 & 0 \\ \nabla^h\cdot & -S \end{pmatrix}\begin{pmatrix} \mathbf{u}^{(n)} \\ p^{(n)} \end{pmatrix} - \begin{pmatrix} 0 & 0 \\ \nabla^h\cdot\mathbf{D}^{-1} & 0 \end{pmatrix}\mathbf{r}^{(n)}\right] \\ &= \mathcal{M}^{-1}\left[\begin{pmatrix} \mathbf{f} \\ 0 \end{pmatrix} + (\mathcal{M} - \mathcal{Q})\begin{pmatrix} \mathbf{u}^{(n)} \\ p^{(n)} \end{pmatrix} - \begin{pmatrix} 0 & 0 \\ \nabla^h\cdot\mathbf{D}^{-1} & 0 \end{pmatrix}\mathbf{r}^{(n)}\right] \\ &= \begin{pmatrix} \mathbf{u}^{(n)} \\ p^{(n)} \end{pmatrix} + \mathcal{M}^{-1}\begin{pmatrix} \mathbb{I} & 0 \\ -\nabla^h\cdot\mathbf{D}^{-1} & \mathbb{I} \end{pmatrix}\mathbf{r}^{(n)}. \end{aligned}$$

Finally, add the projection step to obtain the complete update:

$$\begin{aligned} \begin{pmatrix} \mathbf{u}^{(n+1)} \\ p^{(n+1)} \end{pmatrix} &= \begin{pmatrix} \mathbb{P} & 0 \\ 0 & \mathbb{I} \end{pmatrix}\begin{pmatrix} \mathbf{u}^{(n+\frac{1}{2})} \\ p^{(n+1)} \end{pmatrix} \\ &= \begin{pmatrix} \mathbb{P} & 0 \\ 0 & \mathbb{I} \end{pmatrix}\left[\begin{pmatrix} \mathbf{u}^{(n)} \\ p^{(n)} \end{pmatrix} + \mathcal{M}^{-1}\begin{pmatrix} \mathbb{I} & 0 \\ -\nabla^h\cdot\mathbf{D}^{-1} & \mathbb{I} \end{pmatrix}\mathbf{r}^{(n)}\right]. \end{aligned}$$

Now note that

$$\begin{aligned} \mathbb{P}\mathbf{u}^{(n)} &= \mathbf{u}^{(n)} + \mathbf{D}^{-1}\nabla^h S^{-1}\nabla^h\cdot\mathbf{u}^{(n)} \\ &= \mathbf{u}^{(n)}, \end{aligned}$$

provided that each iterate $\mathbf{u}^{(n)}$ is calculated to accurately satisfy the continuity equation. Thus

$$(3.17)\quad \begin{pmatrix} \mathbf{u}^{(n+1)} \\ p^{(n+1)} \end{pmatrix} = \begin{pmatrix} \mathbf{u}^{(n)} \\ p^{(n)} \end{pmatrix} + \begin{pmatrix} \mathbb{P} & 0 \\ 0 & \alpha\mathbb{I} \end{pmatrix}\mathcal{M}^{-1}\begin{pmatrix} \mathbb{I} & 0 \\ -\nabla^h\cdot\mathbf{D}^{-1} & \mathbb{I} \end{pmatrix}\mathbf{r}^{(n)},$$

where a damping factor $\alpha \in (0, 1]$ has been included as in Remark 3.1.

Remark 3.6. Remark 3.4 also applies to (3.17).

Remark 3.7. In order to satisfy the condition $\mathbb{P}\mathbf{u}^{(n)} = \mathbf{u}^{(n)}$, it is necessary to compute the action of $\mathbb{P}$ to high accuracy. In light of (3.15), it is apparent that

$$\|\nabla^h\cdot\mathbf{u}^{(n+1)}\| = \|\nabla^h\cdot\mathbf{u}^{(n+\frac{1}{2})} + S\phi\|,$$

so that accurate computation of the action of $\mathbb{P}$ can be accomplished by solving (3.15) to a tight absolute tolerance; here an absolute tolerance of $\epsilon_{abs} = 10^{-10}$ was used. Since S is symmetric and positive semidefinite, a preconditioned conjugate gradient method can be used. The preconditioner was a multigrid method composed of volume-averaged restriction, piecewise constant prolongation, symmetric Gauss–Seidel smoothing, V(1,1) cycles, and a Galerkin coarse grid version of (3.9). This latter choice was made necessary by the fact that $\mathbf{D}$ is not conveniently available on the coarser grids, and was made easy to implement because of the choices made for intergrid transfers. The semidefiniteness was treated by requiring ϕ to have a 0 average value, which amounts to a least-squares projection against the null space of S. These choices led to a highly efficient solver with convergence rates of less than 0.1 that were observed to be independent of grid size.

3.3. Multigrid preconditioners with pressure-correction smoothers. Solving the Newton equations (3.1) with a preconditioned iterative method requires computing the action of the inverse of a preconditioner M,

$$z = M^{-1}r, \tag{3.18}$$

at every iteration of the linear solver. In this, r is some vector in the Krylov subspace generated by the Jacobian $F'(x_k)$, M, and the initial residual $F(x_k)$. In a matrix-free Newton–Krylov method, constructing M can be a challenge since the matrix entries of the Jacobian are not available. A good choice for solving the Navier–Stokes equations is

$$M = \mathcal{Q}[\mathbf{u}_k],$$

where $\mathbf{u}_k$ is the current Newton approximation to the velocity field. This captures most of the important features of the Jacobian. Moreover, $\mathcal{Q}[\mathbf{u}_k]$ is readily available since it is generally formed and stored at the beginning of each nonlinear iteration so that it can be used in (2.1) to evaluate F. Alternatively, it is possible to construct M from a lower-order discretization of the Navier–Stokes system. This entails a higher setup cost since it requires a rediscretization, but it can also lead to a system (3.18) that is easier to solve. This does not sacrifice accuracy in the solution, since convergence of the overall procedure is determined by $\|F\|$, which is based on a higher-order discretization.

Once M is chosen, a means for approximating the action of its inverse (3.18) is needed. Multigrid methods [4, 41] provide an efficient means for performing this operation. These approaches compute the solution to a problem discretized on a given grid by approximately solving related problems on coarser grids. The following is a recursive version of a V-cycle, where the h is used to indicate different grid levels and h_c denotes the coarsest grid that is used.

ALGORITHM 3.4 (MG-V(h, M^h, z^h, r^h)).

```
If $h = h_c$ then:
    Solve $M^h z^h = r^h$.
else
    Presmooth $z^h \longleftarrow z^h + B(r^h - M^h z^h)$ $\nu_1$ times.
    Restrict $r^{2h} = I_h^{2h}(r^h - M^h z^h)$.
    Set $z^{2h} = 0$.
    MG-V($2h$, $M^{2h}$, $z^{2h}$, $r^{2h}$).
    Correct $z^h = z^h + I_{2h}^h z^{2h}$.
    Postsmooth $z^h \longleftarrow z^h + B(r^h - M^h z^h)$ $\nu_2$ times.
```

A multigrid algorithm is constructed by supplying the following components:

- a *grid coarsening* strategy,

- a *restriction* operation I_h^{2h} to transfer the problem from a fine grid to a coarser grid,
- a *prolongation* operation I_{2h}^h to transfer the problem from a coarse grid to a finer grid,
- a *coarse grid operator* M^{2h},
- a *coarse grid solver* for solving the problem on the coarsest grid, and
- a *smoother* B that effectively reduces high frequency components of the error.

Grid coarsening is readily achieved by defining each coarse cell to be a union of underlying fine cells. Restriction and prolongation of both velocity and pressure on a staggered grid are accomplished via bilinear interpolation; these are rather standard choices [40]. While coarse grid operators M^{2h} can be computed using a Galerkin coarse grid approximation [44], for simplicity they are determined here by recomputing $M^{2h} = \mathbf{Q}[\mathbf{u}_k^{2h}]$ on the coarser grid. The solution on the coarsest grid h_c is obtained with a fixed number of sweeps of the smoother. Finally, the action of the smoother B is given by either (3.10) or (3.17), once again with $\mathbf{Q}$ fixed at the current inexact Newton approximation.

One additional consideration is required when SIMPLER is used as a smoother. While the projection $\mathbb{P}$ is used to ensure $\nabla^h \cdot \mathbf{u}^h = 0$ to machine accuracy on each grid level, the interpolated coarse grid correction $I_{2h}^h \mathbf{u}^{2h}$ does not satisfy this constraint. An additional application of the projection $\mathbb{P}$ must be applied to the coarse grid correction, so that the velocity portion of the coarse grid correction becomes

$$\mathbf{u}^h = \mathbf{u}^h + \mathbb{P} I_{2h}^h \mathbf{u}^{2h}.$$

4. Numerical evaluations.

The combination of multigrid methods, Newton–Krylov methods, and pressure-correction methods offer a wide range of algorithmic choices. Since these combinations embed iterative methods within iterative methods, it is difficult to predict the most effective combination. Numerical experimentation is necessary to help determine which combinations merit further investigation. This section illustrates effective combinations of these approaches. Except where noted, all reported execution times were measured on a MIPS R10000 processor with a 1 MB L2 cache running at 195 Mhz using the MipsPro 7.2 Fortran compiler and level 2 optimization. Solutions to steady-state problems were obtained using NITSOL [31] on a 128 × 128 grid using an initial approximation $x_0 = 0$ and a GMRES restart value of 100. Such a large restart value is too large for practical computation and is used here only to study the behavior of the preconditioner without the influence of restarting. The nonlinear iterations were terminated when $\|F(x_k)\| \leq 10^{-6}\|F(x_0)\|$.

4.1. Flow in a lid driven cavity.

The driven cavity problem is a classic difficult fluid dynamics benchmark problem. The governing equations consist of the incompressible Navier–Stokes equations (1.1) defined on the unit square $\Omega = [0,1]^2$ with boundary conditions

$$\begin{aligned} u = 1,\ v = 0, &\quad y = 1, \\ u = v = 0 &\quad \text{elsewhere on } \partial\Omega. \end{aligned}$$

In applying the NK-MG method to this problem, convergence difficulties were encountered when the preconditioner was based on the same second-order differences used to discretize the problem. Consequently, all results reported here use a first-order upwind discretization to construct the operator used by the preconditioner. Solutions

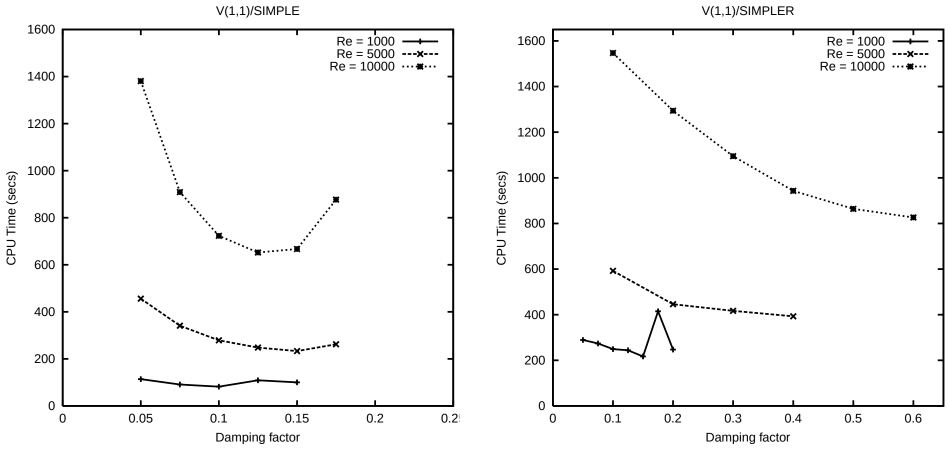


FIG. 4.1. *Sensitivity of multigrid preconditioner to variation in pressure damping factor for Reynolds numbers* 1000, 5000, *and* 10,000.

for Reynolds numbers up to 10,000 were obtained on a 128×128 grid and were within 10% of results published elsewhere [17].

The sensitivity of the performance of the multigrid preconditioners to the pressure damping factor α was investigated. The results for V(1,1) cycles and 2 grid coarsenings are summarized in Figure 4.1; results for larger numbers of pre- or postsmoothing sweeps and for more grid coarsenings are qualitatively similar.

In general, SIMPLER smoothing allowed larger values for the damping factor α. Increasing α beyond those shown in Figure 4.1 caused the inexact Newton method to fail to converge in 200 iterations. Sensitivity to α increased as the Reynolds number was increased. For $Re = 1000$, performance varies by around 25%, excluding the result obtained for SIMPLER smoothing with $\alpha = 0.175$. For this case, backtracking was invoked four times, twice as many as the other cases tested. At $Re = 5000$ performance with SIMPLE smoothing varied by 95%, whereas performance with SIMPLER smoothing varied by 50%; at $Re = 10,000$ the variations were 112% and 87%, respectively.

While multigrid preconditioning with SIMPLE smoothing produced solutions in less CPU time than when SIMPLER smoothing was used, this was the result of a larger number of less expensive iterations. This is due to the need to compute $\mathbb{P}$ to high accuracy: profiling showed that the net cost of this operation was around 20% of the calculation. Note that, by doing so, the preconditioner takes on all the responsibility for enforcing continuity. This is in contrast to the SIMPLE-based preconditioners, where the corrections that are designed to drive the velocity field to conserve mass are not computed with very high accuracy, and the outer Newton iteration is responsible for enforcing $\nabla \cdot \mathbf{u} = 0$. Some consequences of this division of labor are illustrated in Table 4.1, which compares the number of times the preconditioner was applied for the optimal value of α determined above. At lower Reynolds numbers, multigrid preconditioning with SIMPLE smoothing outperforms SIMPLER smoothing by a substantial margin. This gap narrows at $Re = 10,000$, however, as the total number of applications of the SIMPLER-based multigrid preconditioner grows at a slower rate.

SIMPLER smoothing in the multigrid preconditioner also appears to scale better with problem size. This is illustrated in Table 4.2, which shows the usual figures-of-

Table 4.1

Number of applications of multigrid preconditioner (NPRE) *and CPU times* (T) *for different smoothers.*

	SIMPLE		SIMPLER	
Re	NPRE	T	NPRE	T
1000	107	82	96	217
5000	283	233	166	393
10,000	834	719	333	826

Table 4.2

Algorithmic scaling for different smoothers, $Re = 10,000$. NNI: *number of nonlinear iterations;* NLI: *total number of linear iterations;* NBT: *number of backtracks;* T: *CPU time in seconds.*

	SIMPLE			SIMPLER		
	32 × 32	64 × 64	128 × 128	32 × 32	64 × 64	128 × 128
NLI	623	535	808	305	288	305
NNI	32	23	24	31	30	28
NBT	7	6	6	7	6	6
T	31	108	719	47	187	826

merit for evaluating the performance of an inexact Newton method. These results were obtained with the optimal value for α found above. As problem size increases, multigrid with SIMPLE smoothing requires fewer nonlinear iterations but an increasing number of linear iterations than when SIMPLER smoothing is used. While many factors contribute to the peformance of a globalized Newton–Krylov method as problem size increases, the final arbiter is CPU time. In this regard, the CPU time for the case of SIMPLER scales more favorably, owing to the fact that NLI is nearly constant as problem size is increased.

Finally, performance of the linear solver is examined. When a variable forcing term is used, it is difficult to simply characterize the performance of the linear solver. Table 4.3 summarizes some performance statistics for the linear solver over the entire inexact Newton solve. The minimum number of iteration counts occurs early in the computation, when a conservative value for the forcing term is used. During the course of the iteration, a more aggressive value for the forcing term is attempted, generally leading to a higher iteration count for solving the Newton equations. The maximum values always occurred in the last or next-to-last Newton iteration. Overall, the performance of the linear solver with SIMPLER smoothing is better than when SIMPLE smoothing is used, allowing a smaller restart value to be used. However, this is offset by the higher cost per iteration.

4.2. Buoyancy driven flow. Natural convection in an enclosed cavity is a standard benchmark problem that is frequently used to test different numerical schemes and solution methods [10]. The governing equations consist of the incompressible Navier–Stokes equations (1.1) coupled to an energy transport equation,

$$(uT)_x + (vT)_y - \frac{1}{RePr}\Delta T = 0,$$

together with a body force on the fluid that, under the Boussinesq approximation, is proportional to the temperature

$$\mathbf{f} = \begin{pmatrix} 0 \\ \frac{Ra}{Re^2 Pr}T \end{pmatrix},$$

Table 4.3

Performance statistics for solving the Newton equations with multigrid preconditioning and pressure-correction smoothers. The preconditioner is a V(1,1) cycle with two grid coarsenings.

	SIMPLE			SIMPLER		
Re	Min	Avg	Max	Min	Avg	Max
1000	1	7.5	22	1	6.4	21
5000	1	22.6	65	1	9.4	37
10,000	1	33.7	164	1	10.9	72

Table 4.4

Performance of NK-MG with SIMPLE-based smoothers. Multigrid preconditioning used a $V(\nu_1,\nu_2)$ cycle and three grid coarsenings.

	V(1,1)	V(2,1)	V(2,2)	V(4,2)	V(4,4)
NLI	285	221	193	155	133
NNI	19	18	18	19	16
NBT	4	3	3	3	2
T	313	308	329	368	400

where *Pr* is the Prandtl number and *Ra* is the Rayleigh number. Following [25], *Re* is fixed at 1, *Pr* is fixed at 0.71, and the Rayleigh number is varied. The problem is defined on the unit square $\Omega = [0,1]^2$ with boundary conditions

$$\begin{aligned} u = v &= 0 \quad \text{on } \partial\Omega, \\ T(0,y) = 0,\ T(1,y) &= 1, \quad y \in [0,1], \\ T_y(x,0) = T_y(x,1) &= 0, \quad x \in [0,1]. \end{aligned}$$

The additional transport equation is readily accommodated by the SIMPLE and SIMPLER smoothers. Five sweeps of the point Gauss–Seidel method are applied to the discretized energy equation prior to solving the momentum equations. The updated temperature field is then incorporated as a source term before solving the equation governing the vertical component of momentum.

Results for $Ra = 100,000$ using SIMPLE smoothing in the multigrid preconditioner are summarized in Table 4.4, which contains the usual figures-of-merit for evaluating a Newton–Krylov method. Use of a multigrid preconditioner reduces both the number of nonlinear iterations and the number of backtracking steps that were needed. Reducing the number of backtracking steps when employing a variable forcing term is particularly desirable, since each backtracking step increases the forcing term and delays onset of superlinear convergence behavior. In general, V(2,1) cycles in the preconditioner seem to provide a performance "sweet spot." Use of more pre- and/or postsmoothing sweeps decreases NLI, but these reductions are not always enough to offset the resulting higher cost of the preconditioner. Finally, note that use of a multigrid preconditioner improves the performance of GMRES to the extent that a smaller restart value can be used. Thus the high-quality multigrid preconditioner has an additional benefit of reducing the storage costs associated with the Newton–Krylov method. Similar results were obtained for Rayleigh numbers ranging from 20,000 to 500,000.

Results for the same problem that were obtained with the SIMPLER-based preconditioners are summarized in Table 4.5, which reports the same figures-of-merit that appear in Table 4.4. Here only two grid coarsenings were used. Note that the SIMPLER-based preconditioners generally require fewer linear iterations than their SIMPLE-based counterparts. Yet, as observed for the driven cavity problem, the CPU

TABLE 4.5

Performance of the inexact Newton method with SIMPLER-based preconditioners. Column labels are the same as for Table 4.4.

	V(1,1)	V(2,1)	V(2,2)	V(4,2)	V(4,4)
NLI	191	179	178	155	223
NNI	20	19	22	18	20
NBT	3	3	5	3	4
T	464	552	676	790	1418

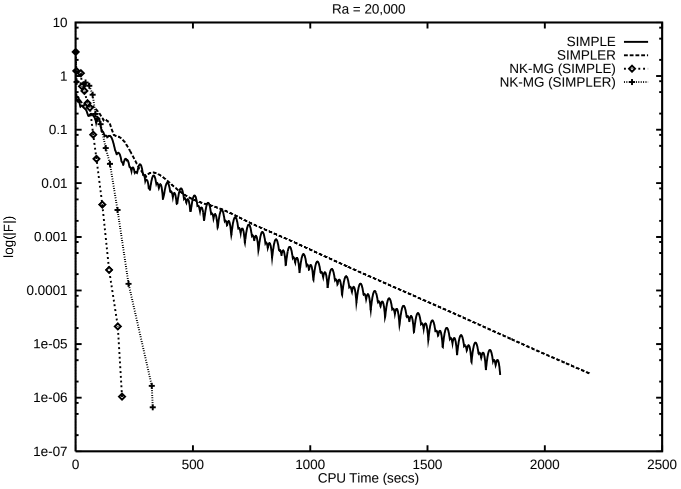


FIG. 4.2. *Convergence histories for SIMPLE, SIMPLER, and the NK-MG methods using these pressure-correction solvers as smoothers. V*(2,1) *cycles are used with SIMPLE smoothing, while V*(1,1) *cycles are used with SIMPLER smoothing.*

times are somewhat higher. Another interesting trend to note is that, while increasing the number of pre- or postsmoothing sweeps generally decreases the total number of linear iterations, it also increases the total execution time. This reflects the tradeoff between the cost and effectiveness of the preconditioner. Both of these observations can be attributed to the cost of enforcing $\nabla^h \cdot \mathbf{u} = 0$ in the preconditioner.

Next, the performance of the NK-MG method using different smoothers is compared with stand-alone versions of the pressure-correction methods. Figure 4.2 shows this comparison for $Ra = 20,000$. The NK-MG method with SIMPLE smoothing is nine times faster than SIMPLE as a stand-alone solver; the NK-MG method with SIMPLER smoothing is nearly seven times faster than SIMPLER as a solver. With SIMPLE smoothing, the largest number of linear iterations taken in an inexact Newton step was 27, so these same results could be obtained by storing only a maximum of 27 Krylov subspace basis vectors. Using a smaller restart value of 10 increases the solution time by roughly a third. On the other hand, when SIMPLER smoothing was used, the largest number of linear iterations taken in an inexact Newton step was 46; using a smaller restart value of 10 increases the solution time by almost 15%.

Dynamic selection of forcing terms as described in [13] employs a safeguard that prevents η_k from getting too small too quickly. More specifically, Choice 1 from [13] uses the safeguard

$$\eta_k \leftarrow \max\{\eta_k, \eta_{k-1}^{(1+\sqrt{5})/2}\} \text{ if } \eta_{k-1}^{(1+\sqrt{5})/2} > \text{threshold}. \tag{4.1}$$

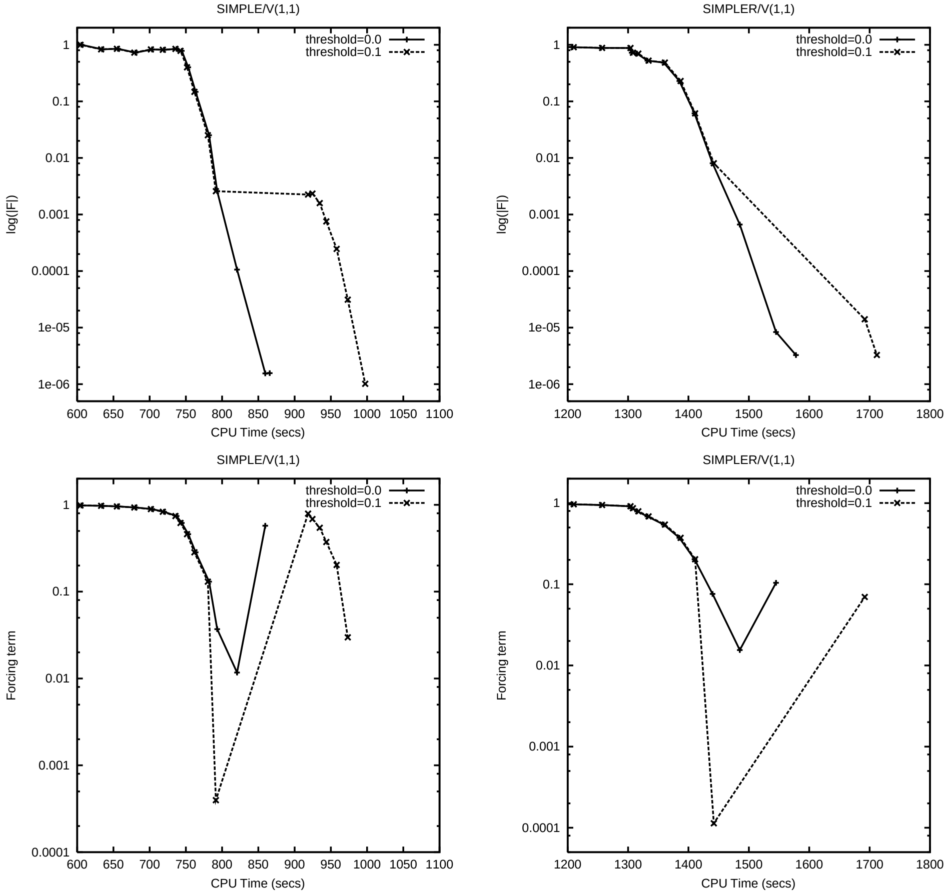


FIG. 4.3. *Convergence histories and forcing terms of the NK-MG method with and without safeguard* (4.1)*. When the threshold is set to* 0*, the safeguard is never disabled. SIMPLE smoothing is on the left, while SIMPLER smoothing is on the right.*

The motivation for this safeguard is that it is possible to request too much accuracy in the solution of the Newton equations, which can happen, for example, when there is serendipitously good agreement between the linear model and the nonlinear function. This safeguard is disabled once the forcing term drops below the threshold value. The choice of a threshold value is somewhat arbitrary, and the implementation of Algorithm INB in NITSOL allows a user to specify this while supplying a default value of 0.1 [31]. The safeguard can be retained by using a threshold value of 0.

While previous studies of the performance of Newton-GMRES with dynamic selection of forcing terms [13, 35, 31] disabled this safeguard, it was found that, when using a multigrid preconditioner, doing so sometimes led to undesirable behavior, as illustrated in Figure 4.3. These plots show the final stages in the solution of the buoyancy-driven flow problem at $Ra = 1,000,000$. The top two plots show $\|F(x_k)\|$, while the bottom two plots show the corresponding η_k. The bottom plots show that, with the safeguard disabled at a threshold of 0.1, a very small value for η_k is generated. For SIMPLE smoothing, the linear solver fails to converge in 100 iterations. Further, sufficient reduction in $\|F\|$ is not achieved, which triggers backtracking, increases the forcing term, and delays convergence of the nonlinear iterations. (The increase in η that is seen on the last iteration when the safeguard is retained is the result of a second safeguard implemented in NITSOL to prevent unnecessary work in reducing $\|F\|$ beyond the requested tolerance [31].) When SIMPLER smoothing is used, backtracking is not triggered, but when (4.1) is disabled, a smaller η is produced in the next to last iteration. While this saves one nonlinear iteration over the case where (4.1) is retained, overall it results in 49 more linear iterations and a higher overall cost to solve the problem.

4.3. Unsteady thermal convection. The issues that are important in a time dependent simulation are somewhat different than was the case for the steady-state case. One major difference is that unlike the steady-state case, there will usually be a very good initial guess for the Newton iteration. This means that globalization strategies are not as important as they were for steady-state problems. For the set of runs shown in this section, SIMPLE was used as the smoother in a multigrid preconditioner, point Gauss–Seidel relaxation was replaced by red-black Gauss–Seidel relaxation, and V(4,2) cycles were used. All of the tests were run over ten time steps using a constant time step size. The test problem is the threee-dimensional analogue of the natural convection problem described in section 4.2. The problem is defined on the unit square $\Omega = [0,1]^3$ with boundary conditions

$$\begin{aligned} u = v = w = 0 &\quad \text{on } \partial\Omega, \\ T(0,y,z) = 0,\ T(1,y,z) = 1, &\quad y,z \in [0,1], \\ T_y(x,0,z) = T_y(x,1,z) = 0, &\quad x,z \in [0,1], \\ T_z(x,y,0) = T_z(x,y,1) = 0, &\quad x,y \in [0,1]. \end{aligned}$$

Before presenting the numerical results, the structure of the computer program that is used is described. The package PETSc [1] is used to implement the inexact Newton solver that was described in section 3.1. In order to define the grid, the package SAMRAI [16] is used. SAMRAI is an object oriented framework for implementing structured adaptive mesh refinement (SAMR). Although adaptive mesh refinement is not used for the work presented in this paper, it is a feature that will be vital in future work.

In order to use PETSc to solve nonlinear systems that are defined on SAMRAI grids, either the data must be copied into a format that PETSc understands or the PETSc vector functions must be extended to act on SAMRAI grids. The former strategy will likely be very inefficient, as large amounts of data must be moved in memory. Instead, SAMRAI provides an interface to PETSc that redirects PETSc vector operations, such as norms and dot products, to act directly on the data that is stored in SAMRAI format. This solver structure is used to solve the natural convection problem and test several aspects of the algorithms.

First the effect of using a better initial iterate on the performance of the linear and nonlinear solver is examined. For this time integrator, the initial conditions are used as the initial iterate for the first time step, and then a linear predictor based on the previous two time steps is used for the remaining time steps in a simulation. The linear predictor should perform much better than the initial iterate for the first time step. The results shown in Table 4.6 are for a 128 × 128 × 128 grid, and they show that the solvers have to work much harder to solve the initial time step because of the poor initial iterate. Also, for all the results shown here, the backtracking scheme was never activated.

Table 4.6
Performance of the linear and nonlinear solvers per time step for the three-dimensional natural convection problem on a grid of $128 \times 128 \times 128$.

		First time step		10 time steps	
Ra	Δt	NNI	NLI	NNI	NLI
10^6	10^{-4}	5	14	4.1	10.4
10^7	10^{-5}	9	24	4.0	8.0
10^8	10^{-6}	8	19	2.8	4.2

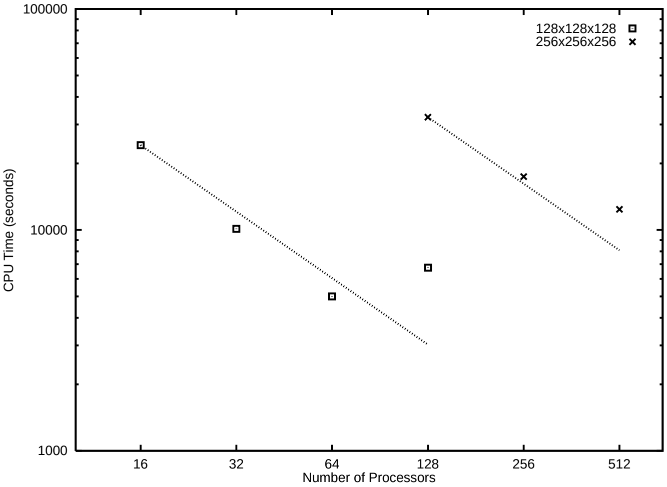


Fig. 4.4. *Parallel performance on an SGI Origin* 2000 *for the natural convection problem.*

The parallel performance of these algorithms is also very important, so the same test problem was run on the SGI Origin 2000 at Los Alamos National Laboratory with $Ra = 10^6$ and $\Delta t = 10^{-4}$ for the $128 \times 128 \times 128$ problem and $Ra = 10^6$ and $\Delta t = 10^{-5}$ for the $256 \times 256 \times 256$ problem. The results, shown in Figure 4.4, indicate that you can get very good scalability for the small problem until the problem size on each processor becomes too small. Then increasing the problem size allows us to get good scalability up to 512 processors. This type of behavior is typical of parallel algorithms for solving PDEs.

5. Summary and conclusions. The pressure-correction methods SIMPLE and SIMPLER have been adapted for use in preconditioning a matrix-free Newton–Krylov method applied to the incompressible Navier–Stokes equations. Application to both two- and three-dimensional problems, as well as unsteady problems, was demonstrated. NK-MG methods with pressure-correction smoothers substantially accelerate the convergence of the pressure-correction methods, albeit at an increase in storage cost, and their performance scales well as problem size is increased. The performance of both methods was found to be sensitive to damping the pressure update; analysis of this behavior will be the subject of future work and will be facilitated by the development of a new correction-oriented version of SIMPLER. SIMPLE-based preconditioning was found to require less CPU time. While SIMPLER-based preconditioning generally led to fewer total linear iterations to complete the inexact Newton solve, this reduction in iteration counts was offset by the additional expense of enforcing continuity at each step. However, SIMPLER-based preconditioning displayed some favorable robustness trends with respect to problem size and Reynolds number, so further investigation of ways to reduce the cost per iteration is warranted. Finally, these methods were found to be straightforward to parallelize, and encouraging initial parallel results have been obtained.

Acknowledgments. Thanks to Dana Knoll of Los Alamos National Laboratory for pointing out that **Q** could be directly reused in the Newton–Krylov preconditioner. The comments of an anonymous referee led to numerous improvements in the presentation. The computations presented in this paper were conducted on facilities provided by the University of Utah Center for High Performance Computing and by Los Alamos National Laboratory.

REFERENCES

[1] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith, *The Portable Extensible Toolkit for Scientific Computing, Version* 2.0.24, http://www.mcs.anl.gov/petsc, 1999.

[2] J. B. Bell, P. Colella, and H. M. Glaz, *A second-order projection method for the incompressible Navier-Stokes equations*, J. Comput. Phys., 85 (1989), pp. 257–283.

[3] M. J. Booth, A. G. Schlijper, L. E. Scales, and A. D. J. Haymet, *Efficient solution of liquid state integral equations using the Newton-GMRES algorithm*, Comput. Phys. Comm., 119 (1999), pp. 122–134.

[4] A. Brandt, *Multilevel adaptive solution to boundary value problems*, Math. Comp., 31 (1977), pp. 333–390.

[5] P. N. Brown, B. Chang, F. Graziani, and C. S. Woodward, *Implicit solution of large-scale radiation-material energy transfer problems*, in Iterative Methods in Scientific Computation IV, D. R. Kincaid and A. C. Elster, eds., Series in Computational and Applied Mathematics 5, IMACS, New Brunswick, NJ, 1999, pp. 343–356.

[6] X.-C. Cai, W. D. Gropp, D. E. Keyes, R. G. Melvin, and D. P. Young, *Parallel Newton–Krylov–Schwarz algorithms for the transonic full potential equation*, SIAM J. Sci. Comput., 19 (1998), pp. 246–265.

[7] X.-C. Cai, W. D. Gropp, D. E. Keyes, and M. D. Tidriri, *Newton-Krylov-Schwarz methods in CFD*, in Proceedings of an International Workshop on Numerical Methods for the Navier-Stokes Equations, F. Hebeker and R. Rannacher, eds., Vieweg Verlag, Braunschweig, 1994, pp. 17–30.

[8] A. J. Chorin, *Numerical solution of the Navier-Stokes equations*, Math. Comp., 22 (1968), pp. 745–762.

[9] C. N. Dawson, H. Klie, M. F. Wheeler, and C. S. Woodward, *A parallel, implicit, cell-centered method for two-phase flow with a Newton-Krylov solver*, Computational Geosci., 1 (1997), pp. 215–249.

[10] G. de Vahl Davis, *Natural convection of air in a square cavity: A benchmark numerical solution*, Internat. J. Numer. Methods Fluids, 3 (1983), pp. 249–264.

[11] R. S. Dembo, S. C. Eisenstat, and T. Steihaug, *Inexact Newton methods*, SIAM J. Numer. Anal., 19 (1982), pp. 400–408.

[12] S. C. Eisenstat and H. F. Walker, *Globally convergent inexact Newton methods*, SIAM J. Optim., 4 (1994), pp. 393–422.

[13] S. C. Eisenstat and H. F. Walker, *Choosing the forcing terms in an inexact Newton method*, SIAM J. Sci. Comput., 17 (1996), pp. 16–32.

[14] R. W. Freund, *A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems*, SIAM J. Sci. Comput., 14 (1993), pp. 470–482.

[15] R. W. Freund, G. H. Golub, and N. M. Nachtigal, *Iterative solution of linear systems*, Acta Numerica, (1992), pp. 87–100.

[16] X. Garaizar, R. Hornung, and S. Kohn, *The use of object oriented design patterns in the SAMRAI structured AMR framework*, in Proceedings of the SIAM Workshop on Object Oriented Methods for Interoperable Scientific and Engineering Computing, M. Henderson, C. Anderson, and S. Lyons, eds., SIAM, Philadelphia, 1998.

[17] U. Ghia, K. N. Ghia, and C. T. Shin, *High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method*, J. Comput. Phys., 48 (1982), pp. 387–411.

[18] F. H. Harlow and J. E. Welch, *Numerical study of large amplitude free surface motions*, Phys. Fluids, 8 (1965), pp. 2182–2189.

[19] P. G. Jacobs, V. A. Mousseau, P. R. McHugh, and D. A. Knoll, *Newton–Krylov–Schwarz techniques applied to the two-dimensional incompressible Navier–Stokes and energy equations*, in Domain Decomposition Methods in Scientific and Engineering Computing, D. Keyes and J. Xu, eds., AMS, Providence, RI, 1994, pp. 503–507.

[20] D. A. Knoll and P. R. McHugh, *An inexact Newton algorithm for solving the tokamak edge plasma fluid equations on multiply connected domains*, J. Comput. Phys., 116 (1995), pp. 281–291.

[21] D. A. Knoll, P. R. McHugh, and D. E. Keyes, *Newton-Krylov methods for low-mach-number compressible combustion*, AIAA J., 34 (1995), pp. 961–967.

[22] D. A. Knoll and V. A. Mousseau, *On Newton-Krylov-multigrid methods for the incompressible Navier-Stokes equations*, J. Comput. Phys., 163 (2000), pp. 262–267.

[23] D. A. Knoll and W. J. Rider, *A multigrid preconditioned Newton–Krylov method*, SIAM J. Sci. Comput., 21 (1999), pp. 691–710.

[24] P. R. McHugh and D. A. Knoll, *Comparison of standard and matrix-free implementations of several Newton-Krylov solvers*, AIAA J., 32 (1994), pp. 2394–2400.

[25] P. R. McHugh and D. A. Knoll, *Fully coupled finite volume solutions of the incompressible Navier–Stokes and energy equations using an inexact Newton method*, Internat. J. Numer. Methods Fluids, 19 (1994), pp. 439–455.

[26] J. Meijerink and H. A. van der Vorst, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix*, Math. Comp., 31 (1977), pp. 148–162.

[27] V. A. Mousseau, D. A. Knoll, and W. J. Rider, *Physics-based preconditioning and the Newton-Krylov method for non-equilibrium radiation diffusion*, J. Comput. Phys., 160 (2000), pp. 743–765.

[28] S. V. Patankar, *A calculation procedure for two-dimensional elliptic situations*, Numer. Heat Transfer, 4 (1981), pp. 409–425.

[29] S. V. Patankar and D. B. Spalding, *A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows*, Int. J. Heat Mass Transfer, 15 (1972), pp. 1787–1806.

[30] M. Pernice, *A hybrid multigrid method for the steady-state incompressible Navier-Stokes equations*, Electron. Trans. Numer. Anal., 10 (2000), pp. 74–91.

[31] M. Pernice and H. F. Walker, *NITSOL: A Newton iterative solver for nonlinear systems*, SIAM J. Sci Comput., 19 (1998), pp. 302–318.

[32] W. J. Rider, D. A. Knoll, and G. L. Olson, *A multigrid Newton-Krylov method for multimaterial equilibrium radiation diffusion*, J. Comput. Phys., 152 (1999), pp. 164–191.

[33] Y. Saad, *ILUT: A dual threshold incomplete LU factorization*, Numer. Linear Algebra Appl., 1 (1994), pp. 387–402.

[34] Y. Saad and M. H. Schultz, *GMRES: A generalized minimum residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[35] J. N. Shadid, R. S. Tuminaro, and H. F. Walker, *An inexact Newton method for fully coupled solution of the Navier-Stokes equations with heat and mass transport*, J. Comput. Phys., 137 (1997), pp. 155–185.

[36] G. J. Shaw and S. Sivaloganathan, *On the smoothing properties of the SIMPLE pressure-correction algorithm*, Internat. J. Numer. Methods Fluids, 8 (1988), pp. 441–461.

[37] S. Sivaloganathan and G. J. Shaw, *A multigrid method for recirculating flows*, Internat. J. Numer. Methods Fluids, 8 (1988), pp. 417–440.

[38] H. A. van der Vorst, *BI-CGSTAB: A fast and smoothly converging variant of BI-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 631–644.

[39] J. P. van Doormaal and G. D. Raithby, *Enhancements of the SIMPLE method for predicting incompressible fluid flows*, Numer. Heat Transfer, 7 (1984), pp. 147–163.

[40] S. P. Vanka, *Block implicit multigrid solution of Navier-Stokes equations in primitive variables*, J. Comput. Phys., 65 (1986), pp. 138–158.

[41] P. Wesseling, *An Introduction to Multigrid Methods*, John Wiley and Sons, New York, 1991.

[42] G. Wittum, *Multigrid methods for Stokes and Navier-Stokes equations*, Numer. Math., 54 (1989), pp. 543–563.

[43] C. S. Woodward, *A Newton-Krylov multigrid solver for variably saturated flow problems*, in Proceedings of the Twelfth International Conference on Computation Methods in Water Resources, Vol. 2, Computational Mechanics Publications, Southhampton, 1998, pp. 609–616.

[44] S. Zeng and P. Wesseling, *An Efficient Algorithm for the Computation of Galerkin Coarse Grid Approximation for the Incompressible Navier-Stokes Equations*, Tech. report 92-40, Faculty of Technical Mathematics and Informatics, Delft University of Technology, Delft, The Netherlands, 1992.

SIAM J. SCI. COMPUT.
Vol. 23, No. 2, pp. 419–429

ON THE INCOMPLETE CHOLESKY DECOMPOSITION OF A CLASS OF PERTURBED MATRICES*

GÉRARD MEURANT†

Abstract. We consider how to cheaply compute an incomplete Cholesky decomposition of symmetric perturbed matrices $C = \epsilon I + A$ with a small ϵ when knowing an incomplete decomposition of A. Numerical examples are provided that show the effectiveness of the proposed approach.

Key words. incomplete factorization, conjugate gradient

AMS subject classifications. 65F10, 65F50

PII. S1064827500371244

1. Introduction.

In this paper we are mainly concerned with the incomplete Cholesky decomposition of symmetric M-matrices

$$C = \epsilon I + A,$$

where A arises from the discretization of an elliptic partial differential equation, ϵ being a "small" positive real parameter. Such problems arise, for instance, from discretizing parabolic equations.

As a model problem we can use the two-dimensional heat equation

$$\frac{\partial u}{\partial t} - \Delta u = f$$

in the unit square with Dirichlet boundary conditions and an initial condition $u(x,0) = u_0(t)$. We discretize in space with finite differences with a stepsize h and a time implicit scheme. Then we obtain

$$\left(\frac{I}{k} + \frac{1}{h^2}A\right) u^{n+1} = \frac{u^n}{k} + f^{n+1},$$

where k is the time step and $1/h^2\, A$ is the matrix of the corresponding elliptic problem. For some problems it makes sense to choose $k \simeq h$. After multiplication by h^2 the matrix of the problem is $C = kI + A$, where k is "small."

Since the matrix C is symmetric positive definite, we would like to solve the linear system at each time step with the preconditioned conjugate gradient algorithm. A very popular preconditioner is the incomplete Cholesky decomposition without any fill-in IC(1,1) (sometimes also denoted as IC(0)); c.f. [3], [4] or [5], [6] for a review. Usually the time step k must be small to obtain the convergence of the approximation. Therefore, it is interesting to know if one can compute an approximation of the incomplete decomposition of C knowing the decomposition of A. Moreover, very often the time step is not constant, and therefore one cannot compute the decomposition of C once for all. It has to be recomputed at each time step. Hence it would be interesting to find a way to cheaply update the incomplete decomposition from one time step to the next.

*Received by the editors May 26, 2000; accepted for publication (in revised form) November 29, 2000; published electronically July 10, 2001.
http://www.siam.org/journals/sisc/23-2/37124.html

†CEA/DIF, DCSA/ED, BP 12, 91680 Bruyères le Chatel, France (gerard.meurant@cea.fr).

Matrices of this type have been considered in previous works, mainly to stabilize the incomplete factorization of A (see [2], [1]) when the straightforward factorization of A gives some small pivots.

In section 2, we describe algorithms corresponding to perturbations of order 0 and 1 in ϵ. Section 3 gives numerical examples for several problems with comparisons between the incomplete decomposition of C, that of A, and the SSOR preconditioning. For a definition of these algorithms, see [5]. We will also give some results for problems not arising from parabolic partial differential equations, even for cases where A is not an M-matrix. Then in section 4, we apply the previous results to the solution of the heat equation. Section 5 deals with another problem where $C = I + \epsilon A$.

2. Approximate preconditioners. We first recall the first step of the incomplete Cholesky decomposition of a matrix C whose elements are denoted $c_{i,j}$. Let G be a set of indices corresponding, for instance, to the nonzero structure of A and

$$C = C_1 = \begin{pmatrix} c_{1,1} & c_1^T \\ c_1 & E_2 \end{pmatrix} = \begin{pmatrix} c_{1,1} & f_1^T \\ f_1 & E_2 \end{pmatrix} - \begin{pmatrix} 0 & r_1^T \\ r_1 & 0 \end{pmatrix} = M_1 - R_1,$$

with

$$c_1 = f_1 - r_1,$$

$$(f_1)_i = 0, \text{ if } (i,1) \notin G \Rightarrow (r_1)_i = -(c_1)_i,$$

$$(f_1)_i = (c_1)_i, \text{ if } (i,1) \in G \Rightarrow (r_1)_i = 0.$$

Then, we factorize M_1:

$$M_1 = \begin{pmatrix} c_{1,1} & 0 \\ l_1 & I \end{pmatrix} \begin{pmatrix} c_{1,1}^{-1} & 0 \\ 0 & C_2 \end{pmatrix} \begin{pmatrix} c_{1,1} & l_1^T \\ 0 & I \end{pmatrix} = L_1 \Sigma_1 L_1^T.$$

We obtain

$$l_1 = f_1,$$

$$C_2 = E_2 - \frac{1}{c_{1,1}} f_1 f_1^T.$$

The next step is applying the same decomposition on C_2,

$$C_2 = \begin{pmatrix} c_{2,2}^{(2)} & c_2^T \\ c_2 & E_3 \end{pmatrix} = \begin{pmatrix} c_{2,2}^{(2)} & f_2^T \\ f_2 & E_3 \end{pmatrix} - \begin{pmatrix} 0 & r_2^T \\ r_2 & 0 \end{pmatrix} = M_2 - R_2,$$

where f_2 is obtained from c_2 by setting to zero the elements for which the indices $(i,2)$ do not belong to G. Let

$$L_2 = \begin{pmatrix} c_{1,1} & 0 \\ 0 & \begin{pmatrix} c_{2,2}^{(2)} & 0 \\ l_2 & I \end{pmatrix} \end{pmatrix}.$$

Note that $l_2 = f_2$ and that we shall never throw away a diagonal entry.

At the end of the second step, we have

$$C = L_1 L_2 \Sigma_2 L_2^T L_1^T - L_1 \begin{pmatrix} 0 & 0 \\ 0 & R_2 \end{pmatrix} L_1^T - R_1,$$

but

$$L_1 L_2 = \begin{pmatrix} c_{1,1} & 0 \\ l_1 & \begin{pmatrix} c_{2,2}^{(2)} & 0 \\ l_2 & I \end{pmatrix} \end{pmatrix} \quad \text{and } L_1 \begin{pmatrix} 0 & 0 \\ 0 & R_2 \end{pmatrix} L_1^T = \begin{pmatrix} 0 & 0 \\ 0 & R_2 \end{pmatrix}.$$

Therefore, we have a factor with the desired structure and we can go on as long as the pivots $c_{i,i}^{(i)}$ are nonzero. It has been proven that IC is feasible whatever the set G is when C is an H-matrix. We apply this algorithm to $C = \epsilon D + A$, where D is a diagonal matrix with nonzero diagonal elements d_i. We use a matrix $D \neq I$ because with the order 1 algorithm to be described soon, the diagonal entries are going to be modified so that we do not get the identity for all steps of the algorithm. We have

$$c_{1,1} = \epsilon d_1 + a_{1,1},$$

$$l_1 = f_1,$$

$$C_2 = \epsilon D_2 + A_2 - \frac{1}{\epsilon d_1 + a_{1,1}} f_1 f_1^T,$$

where these matrices are defined by

$$C = \begin{pmatrix} \epsilon d_1 + a_{1,1} & a_1^T \\ a_1 & \epsilon D_2 + A_2 \end{pmatrix}.$$

Rather than computing C_2 exactly, we would like to use an asymptotic expansion related to ϵ for the ratio. For the order 0 we have

$$C_2 = \epsilon D_2 + A_2 - \frac{1}{a_{1,1}} f_1 f_1^T;$$

that is to say, we add ϵD_2 to what we would have obtained for the incomplete decomposition of A.

The order 1 gives

$$C_2 = \epsilon \left(D_2 + \frac{d_1 f_1 f_1^T}{a_{1,1}^2} \right) + A_2 - \frac{1}{a_{1,1}} f_1 f_1^T.$$

To motivate the choices we are going to make later on, let us look at what we get for a two-dimensional finite difference matrix. As an example, we take the Poisson equation in a square with a mesh size $h = 1/(m+1)$. This leads to a matrix of order $n = m^2$, which can be written blockwise as

$$A = \begin{pmatrix} T & -I & & & \\ -I & T & -I & & \\ & \ddots & \ddots & \ddots & \\ & & -I & T & -I \\ & & & -I & T \end{pmatrix}$$

with blocks of order m

$$T = \begin{pmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 4 & -1 \\ & & & -1 & 4 \end{pmatrix}.$$

The matrix A has five nonzero diagonals. For a generic row i and more general problems, the nonzero coefficients are $a_{i,i-m}, a_{i,i-1}, a_{i,i}, a_{i+1,i}, a_{i+m,i}$. When we compute the incomplete decomposition IC(1,1) (with no fill-in) of this matrix obtaining $M = (\Sigma + L)\Sigma^{-1}(\Sigma + L^T)$, it is easy to see that we only need to compute the diagonal matrix Σ whose elements are denoted by $\sigma_{i,i}$. The matrix L is the strictly lower triangular part of A. The diagonal elements are given by

$$\sigma_{i,i} = a_{i,i} - \frac{a_{i,i-1}^2}{\sigma_{i-1,i-1}} - \frac{a_{i,i-m}^2}{\sigma_{i-m,i-m}}.$$

In this formula, entries of A (resp., Σ) whose indices do not exist are taken to be 0 (resp., 1). The two ratios arise from the two nonzero elements in each column of L and from the upper triangular part of A in each row of L^T.

Let us look at what we obtain in the first step of the decomposition of $C = \epsilon D + A$. For order 0 there is no problem as we have just to compute $\sigma_{i,i}$ as before and to add ϵd_i.

Handling order 1 is a little more tricky. We obtained

$$C_2 = \epsilon \left(D_2 + \frac{d_1 f_1 f_1^T}{a_{1,1}^2} \right) + A_2 - \frac{1}{a_{1,1}} f_1 f_1^T.$$

To what we would have obtained for A, that is,

$$A_2 - \frac{1}{a_{1,1}} f_1 f_1^T,$$

which corresponds to the computation of $\sigma_{i,i}$, we have to add the correction of order ϵ. That is,

$$\epsilon \left(D_2 + \frac{d_1 f_1 f_1^T}{a_{1,1}^2} \right).$$

Since there are only two nonzero elements in f_1 for the model problem, the outer product $f_1 f_1^T$ gives two diagonal modifications for indices $(2,2)$ and $(m+1, m+1)$ (when these terms exist). Therefore, we add

$$\epsilon \left(d_2 + d_1 \frac{a_{2,1}^2}{a_{1,1}^2} \right)$$

to the element of index $(2,2)$ and

$$\epsilon \left(d_{m+1} + d_1 \frac{a_{m+1,1}^2}{a_{1,1}^2} \right)$$

to the element of index $(m+1, m+1)$. This modifies the diagonal terms of order ϵ. Therefore, we see that the modification of the diagonal terms is recursive. This means that there is no gain from doing the decomposition of C from scratch. Consequently, we decided to bypass the recursion. We apply only the modifications to the initial D. For example, the correction of the element $(3,3)$ will be

$$\epsilon \left(d_3 + d_2 \frac{a_{3,2}^2}{\sigma_{2,2}^2} \right).$$

This will work when ϵ is small but not with larger values of the parameter. In the latter case we have

$$\frac{1}{\epsilon d_1 + a_{1,1}} \simeq \frac{1}{\epsilon d_1}.$$

Therefore, in the formulas for order 1 we replace $\sigma_{i,i}$ by $\sigma_{i,i} + \epsilon d_i$. This does not make too much of a difference when ϵ is small and gives asymptotically the exact answer when it is large. For instance, the final correction for the element $(3,3)$ in the second step will be

$$\epsilon \left(d_3 + d_2 \frac{a_{3,2}^2}{(\sigma_{2,2} + \epsilon d_2)^2} \right).$$

This algorithm is what we call order 1 in the numerical experiments. This has the additional advantage that all the corrections can proceed in parallel. In this way we get rid of the recursion of the incomplete Cholesky factorization which is not easily parallelizable. However, note that there is still a recursion when solving the triangular systems.

In the case of a matrix arising from a diffusion equation with a constant coefficient, we can perform an asymptotic analysis of the diagonal elements of the incomplete factors. If $a_{i,i} = a, a_{i,i-1} = b, a_{i,i-m} = c$, then the elements of the incomplete decomposition $\sigma_{i,i}$ (within a block) converge rapidly to σ:

$$\sigma = \frac{a+s}{2}, \quad s = \sqrt{a^2 - 4(b^2 + c^2)}.$$

If the diagonal coefficients of A are perturbed by ϵ, we obtain a limit $\bar{\sigma}$:

$$\bar{\sigma} \simeq \frac{a+s}{2} + \frac{\epsilon}{s} \frac{a+s}{2}.$$

The fact that the difference between the exact factorization and the approximate one is small is also illustrated in Figure 2.1, where we show the relative differences between the exact and approximate order 0 values of the second block of diagonal coefficients for the perturbed Poisson equation with a 30×30 mesh as a function of the relative index in the block. The solid line is $\epsilon = 10^{-2}$, the dashed line is $\epsilon = 10^{-1}$, the dot-dashed line is $\epsilon = 1$, the dotted line is $\epsilon = 10$, the plus signs are $\epsilon = 100$, and the circles are $\epsilon = 1000$. Over all values of ϵ the maximum relative difference is 4%.

This technique can be generalized to any nonzero structure for A. For order 0 we just add ϵD to the diagonal. For the order 1 we modify only the diagonal, neglecting the recursiveness, and we add an order of ϵ term in the denominator to obtain the correct behavior when ϵ is large. The method can also be applied to matrices arising from finite element methods. In case mass lumping is used we can do exactly the same thing. If the mass matrix is not diagonal, we can do modifications to the nonzero entries but neglecting the recursiveness that is using the initial values of the entries of the mass matrix.

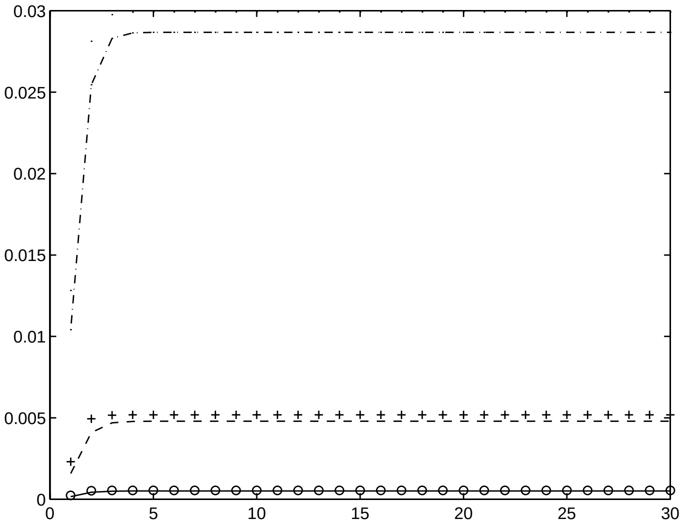


FIG. 2.1. *Relative differences between diagonal coefficients.*

TABLE 1
Poisson problem.

ϵ	IC(C)	Order 0	Order 1	IC(A)	SSOR
320	2	3	3	40	2
80	3	4	4	39	3
20	4	5	5	35	4
5	6	6	6	26	7
1.25	10	10	10	14	11
0.325	17	17	17	15	19
$7.812\ 10^{-2}$	26	26	26	25	29
$1.953\ 10^{-2}$	32	32	32	32	39
$4.882\ 10^{-3}$	34	34	34	34	40

3. Numerical experiments. We denote by IC the incomplete Cholesky decomposition without fill-in. We use several examples to compare the following preconditioners:

1. IC for $C = \epsilon I + A$,
2. approximate IC of C order 0,
3. approximate IC of C order 1,
4. IC for A,
5. SSOR with $\omega = 1$.

We first consider matrices arising from diffusion equations in the unit square with homogeneous Dirichlet boundary conditions. The first example is the Poisson equation we described in section 2. We solve a linear system whose solution is $x = \{1, 1, \ldots, \}^T$. The initial iterate is chosen at random, and we stop the iterations as soon as the relative norm residual is less 10^{-10}. We use a regular 30×30 cartesian mesh. We start with $\epsilon = 320$, and we divide it by 4 several times. The results are given in Table 1.

For this example, the two approximate decompositions give almost the same number of iterations as the "exact" incomplete decomposition. The results of the approximate decompositions are also quite good for large ϵ's. This is linked to the fact that

Table 2
Diffusion problem with discontinuous coefficients.

ϵ	IC(C)	Order 0	Order 1	IC(A)	SSOR
320	16	16	16	148	19
80	24	24	24	171	28
20	29	29	29	129	35
5	32	32	32	79	37
1.25	33	33	33	44	38
0.325	34	34	34	36	40
$7.812\ 10^{-2}$	37	37	37	37	43
$1.953\ 10^{-2}$	39	39	39	39	45
$4.882\ 10^{-3}$	41	41	41	41	47

Table 3
Diffusion problem with anisotropic coefficients.

ϵ	IC(C)	Order 0	Order 1	IC(A)	SSOR
320	4	8	8	143	6
80	5	12	9	92	10
20	7	15	11	53	18
5	10	14	12	28	33
1.25	15	15	15	16	57
0.325	25	26	26	25	97
$7.812\ 10^{-2}$	34	34	34	33	126
$1.953\ 10^{-2}$	38	38	38	38	134
$4.882\ 10^{-3}$	39	39	39	39	136

in this case the matrix is diagonally dominant and that we add a proper correction for the order 1. We also note that in this case the results of the order 1 are the same as those of order 0. Of course, when ϵ is small we obtain the same number of iterations when we only use IC(A).

The second example is a diffusion problem with discontinuous diffusion coefficients and Dirichlet boundary conditions on the unit square. The diffusion coefficient is 1000 in $[1/4, 3/4] \times [1/4, 3/4]$ and 1 elsewhere. We use the same mesh and parameters as before. Results are provided in Table 2. The conclusions are the same as for the first example.

The third example is a diffusion problem with an anisotropic coefficient and Dirichlet boundary conditions on the unit square. The x diffusion coefficient is 100 in $[1/4, 3/4] \times [0, 1]$ and 1 elsewhere. The y diffusion coefficient is 1 everywhere. The parameters are the same as before. Results are given in Table 3.

This is a difficult problem, and we can see there are differences between IC and the approximate decompositions. For this problem we get an improvement when going from order 0 to order 1 for middle range values of ϵ . However, the differences are small, and we can still conclude that it could be useful to use the approximate decomposition.

We now consider some matrices from the Harwell–Boeing collection or from the Boeing collection arising from Tim Davis's Web site (http://www.cise.ufl.edu). We had to normalize some of these matrices in order for the perturbations ϵI to be meaningful. We use the following examples.

1. 1138-bus. An admittance matrix of order 1138 with 4054 nonzeros. It was normalized. Note that the unperturbed matrix is close to being singular.
2. bcsstk01. A stiffness matrix of order 48 with 400 nonzeros. It was normalized.

Table 4
1138-bus.

ϵ	IC(C)	Order 0	Order 1	IC(A)	SSOR
1000	1	2	2	205	1
250	2	3	3	249	2
62.5	2	3	3	249	2
15.625	2	4	4	242	3
3.51	3	5	5	215	4
0.98	6	8	8	163	6
0.24	10	14	13	110	10
0.0610	18	22	21	93	19
0.0015	30	34	33	86	34
0.0038	46	48	48	81	62
$9.54\ 10^{-4}$	65	65	65	80	108
$2.38\ 10^{-4}$	83	83	83	86	175
$5.963\ 10^{-5}$	101	101	101	102	250
$1.490\ 10^{-5}$	114	114	114	114	320

This matrix is not diagonally dominant, nor an M-matrix, but nevertheless positive definite.

3. gr3030. A matrix arising from a nine point approximation to the Laplacian on the unit square with a 30×30 mesh. It has order 900 and 7744 nonzeros.

4. msc00726. A matrix of order 726 with 34518 nonzeros from Nastran. This matrix was normalized.

We ran the different problems and preconditioners with values of ϵ ranging from 1000 to $1.49\ 10^{-5}$. In Tables 4 to 7 we report the number of iterations obtained by using a stopping criterion of 10^{-6} on the relative residual norm in CG.

For 1138-bus there is no gain by using order 1 over order 0. The SSOR preconditioner gives better results than the approximate ICs for large ϵ, but for small ones the SSOR results are much worst. We can see that although this is a different kind of problem the approximate preconditioners still give very good results. Note that ϵ has to be very small for having good results with IC(A). The conclusions for bcsstk01 are the same as for the previous example. For the last two problems gr3030 and msc00726 we can also draw the same conclusions. We can obtain a very good preconditioner by just modifying the diagonal elements in a parallel way. A general remark is that using IC(A) is fine when $\epsilon \to 0$, but with the perturbed factorizations we can also get good results when ϵ is large. Moreover, the results are generally better than using straight SSOR.

4. Application to the heat equation.

We consider the following problem:

$$\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} = f, \quad \text{in } \Omega =]0,1[^2,$$

with Dirichlet boundary conditions and a given initial condition.

We discretize with finite differences and a time implicit scheme as shown in the introduction. We use the conjugate gradient algorithm to solve the linear system we obtain at each time step.

We solved a problem whose exact solution is $u = (1+t^3)\sin(\pi x)\sin(\pi y)$. We use 30 discretization points in each direction and $k = h$. We obtained exactly the same total number of CG iterations using either the "exact" incomplete Cholesky decomposition recomputed at each time step or the approximate one of order 0. Therefore,

Table 5
*bcsstk*01.

ϵ	IC(C)	Order 0	Order 1	IC(A)	SSOR
1000	1	2	2	31	1
250	2	3	3	30	2
62.5	2	3	3	31	2
15.625	2	4	4	29	2
3.51	3	5	5	26	4
0.98	4	7	7	19	6
0.24	6	8	8	13	9
0.0610	7	10	10	12	14
0.0015	10	11	11	12	18
0.0038	12	12	12	13	23
$9.54\ 10^{-4}$	13	13	13	14	24
$2.38\ 10^{-4}$	14	14	14	14	24
$5.963\ 10^{-5}$	14	14	14	14	24
$1.490\ 10^{-5}$	14	14	14	14	24

Table 6
*gr*3030.

ϵ	IC(C)	Order 0	Order 1	IC(A)	SSOR
1000	2	2	2	25	2
250	2	3	3	24	2
62.5	2	3	3	23	2
15.625	3	4	4	19	4
3.51	5	5	5	12	6
0.98	7	7	7	7	10
0.24	11	11	11	14	15
0.0610	14	14	14	16	19
0.0015	16	16	16	17	22
0.0038	17	17	17	17	23
$9.54\ 10^{-4}$	17	17	17	17	23
$2.38\ 10^{-4}$	17	17	17	17	23
$5.963\ 10^{-5}$	17	17	17	17	23
$1.490\ 10^{-5}$	17	17	17	17	23

Table 7
*msc*00726.

ϵ	IC(C)	Order 0	Order 1	IC(A)	SSOR
1000	1	2	2	44	1
250	2	3	3	44	2
62.5	2	3	3	43	2
15.625	2	4	4	41	3
3.51	3	6	6	34	4
0.98	5	8	8	23	6
0.24	8	9	9	13	10
0.0610	12	12	12	12	17
0.0015	18	18	18	18	25
0.0038	26	26	26	26	38
$9.54\ 10^{-4}$	30	30	30	30	40
$2.38\ 10^{-4}$	31	31	31	31	41
$5.963\ 10^{-5}$	31	31	31	31	41
$1.490\ 10^{-5}$	31	31	31	31	41

TABLE 8
Diffusion problem with anisotropic coefficients.

ϵ	IC(C)	Order 1	Order 2	ICd	SSOR
320	40	138	86	40	138
80	39	135	85	39	135
20	35	129	82	36	129
5	29	111	71	29	111
1.25	18	70	43	18	70
0.325	11	40	25	14	40
$7.812\ 10^{-2}$	7	22	13	15	22
$1.953\ 10^{-2}$	5	12	7	13	12
$4.882\ 10^{-3}$	4	7	4	9	7

this shows that in this case it is useless to recompute the decomposition at each time step; we just have to update the diagonal of the decomposition of A which is computed during the initialization phase. Of course, updating the incomplete factorization was much cheaper than recomputing at every time step. We do not give any computer times since this computation was done using Matlab for which computer times depend very much on how the programs are written.

5. Perturbation of a diagonal matrix. For completeness, we now consider the incomplete Cholesky decomposition of a matrix $C = I + \epsilon A$, although this case has much less practical applications than the case we handle in section 2. Let $c_1 = f_1 - r_1$, $l_1 = \epsilon f_1$. For the first step we get

$$C_2 = I + \epsilon A_2 - \frac{1}{1 + \epsilon a_{1,1}} l_1 l_1^T = I + \epsilon \left(A_2 - \frac{\epsilon}{1 + \epsilon a_{1,1}} f_1 f_1^T \right).$$

Now we use an asymptotic expansion of the ratio. The order 1 expansion gives $l_1 = \epsilon f_1$, $C_2 = I + \epsilon A_2$. This is nothing else than the SSOR preconditioner with $\omega = 1$. For the order 2 expansion we obtain

$$C_2 = I + \epsilon A_2 - \epsilon^2 f_1 f_1^T.$$

Looking at this formula, we may already think that this correction cannot give good results for large ϵ. It may even happen that the preconditioner is not positive definite. To obtain something which can work for both small and large ϵ, it makes sense to use a weighting factor. Therefore, we propose to use

$$C_2 = I + \epsilon A_2 - \frac{\epsilon^2}{\epsilon a_{1,1} + 1} f_1 f_1^T.$$

This is what is denoted as order 2 in the results. As in section 2, we neglect the recursion when we compute the diagonal corrections.

Another way is to use the incomplete decomposition of A which computes a diagonal d and to set the diagonal elements of the approximate decomposition to $1 + \epsilon d_{i,i}$. We denote this method as ICd. We consider the third example from section 3 for which results are given in Table 8.

The results in Table 8 show that the order 1 approximation indeed gives the same results as SSOR with $\omega = 1$. The order 2 gives better results than order 1. For small ϵ it gives almost the same results as IC(C). Finally, ICd always gives good results. These are not too far from those of IC(C) for all values of ϵ although order 2 gives better results for very small ϵ. Therefore, ICd seems to be the method of choice for this case.

6. Conclusion. In this paper we have shown that we can efficiently and easily compute incomplete Cholesky-like preconditioners of $\epsilon I + A$ and $I + \epsilon A$ when we know the incomplete decomposition of A. The proposed methods work for a large range of values of the perturbation parameter ϵ. These approximate factorizations can be useful when solving time dependent partial differential equations since we do not anymore have to recompute the factorization at each time step but only have to update it, which is a cheap and parallel operation.

Acknowledgment. Thanks to Henk van der Vorst for helpful comments on a preliminary draft of this paper.

REFERENCES

[1] H. C. Elman, *A stability analysis of incomplete LU factorizations*, Math. Comp., 47 (1986), pp. 191–217.

[2] T. A. Manteuffel, *An incomplete factorization technique for positive definite linear systems*, Math. Comp., 34 (1980), pp. 473–497.

[3] J. A. Meijerink and H. Van der Vorst, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric M–matrix*, Math. Comp., 31 (1977), pp. 148–162.

[4] J. A. Meijerink and H. Van der Vorst, *Guidelines for the usage of incomplete decompositions in solving sets of linear equations as they occur in practical problems*, J. Comput. Phys., 44 (1981), pp. 134–155.

[5] G. Meurant, *Computer Solution of Large Linear Systems*, North–Holland, Amsterdam, 1999.

[6] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing, Boston, 1996.

SIAM J. Sci. Comput.
Vol. 23, No. 2, pp. 430–441

AN AGGREGATION-BASED DOMAIN DECOMPOSITION PRECONDITIONER FOR GROUNDWATER FLOW*

E. W. JENKINS[†], C. E. KEES[‡], C. T. KELLEY[§], and C. T. MILLER[‡]

Abstract. We consider theoretical and computational issues associated with an aggregation-based domain decomposition preconditioner applied to a Bi-CGSTAB iterative solver used to solve both Laplace's equation and an important nonlinear model from hydrology used to simulate unsaturated flow, Richards' equation. Theoretical results for Laplace's equation provide estimates of the condition number and the rate of convergence for a two-level Schwarz domain decomposition preconditioner. Computational results for Laplace's equation and Richards' equation show excellent scalability, although no theory is yet available to support the results for the difficult nonlinear problem.

Key words. domain decomposition, Newton–Krylov–Schwarz methods, Richards' equation, nonlinear equations, aggregation

AMS subject classifications. 65H10, 65N55, 76S05, 76T05, 86A05

PII. S1064827500372274

1. Introduction.

In this paper we report on a scalability study and derive condition number estimates for a two-level additive Schwarz preconditioner. In our previous work we have applied this preconditioner to a set of problems in hydrology [8, 9, 7]. We build the coarse mesh problem with aggregation, an approach used in both algebraic multigrid [2, 21] and domain decomposition methods [11, 12, 13, 3]. In the case of minimal overlap, we implement this as a simple unweighted sum of nodal values. This implementation permits a simple construction of a two-level preconditioner on unstructured grids.

Our methods were developed for use in the adaptive hydrology model (ADH) [18], a production code being developed at the U.S. Army Engineer Research and Development Center. The use of aggregation arose from necessity. In the applications reported in [8, 9, 7] the subdomains were irregular, and a coarse mesh based on "hat functions" over the subdomains was impractical. For the same reason, we needed minimal overlap between subdomains. Unlike the method from [5], we do not need to create a coarse mesh geometry or use geometric information about the subdomains. Neither theoretical analysis of this algorithm applied to any problem nor in-depth scalability studies have been performed to date to our knowledge.

The overall goal of this work is to advance our understanding of this aggregation-based domain decomposition method. The specific objectives are (1) to develop estimates of the condition number as a function of coarse and fine grid size for Laplace's

*Received by the editors May 18, 2000; accepted for publication (in revised form) February 21, 2001; published electronically July 10, 2001. This research was supported in part by Army Research Office contract DAAD19-99-1-0186, U.S. Army contract DACA39-95-K-0098, National Science Foundation grants DMS-9700569 and DMS-0070641, a Cray Research Corporation Fellowship, and a Department of Education GAANN fellowship. Computing activity was partially supported by an allocation from the North Carolina Supercomputing Center.

http://www.siam.org/journals/sisc/23-2/37227.html

[†] TICAM, University of Texas at Austin, ACES 5.340, 201 E. 24th Street, Austin, TX 78712 (lea@ticam.utexas.edu).

[‡] Department of Environmental Sciences and Engineering, University of North Carolina, 104 Rosenau Hall, Chapel Hill, NC 27599-7400 (chris_kees@unc.edu, casey_miller@unc.edu).

[§] Center for Research in Scientific Computation and Department of Mathematics, North Carolina State University, Box 8205, Raleigh, NC 27695-8205 (Tim_Kelley@ncsu.edu).

equation; (2) to evaluate computationally the scalability of this algorithm for Laplace's equation and to compare to theoretical estimates; and (3) to apply this algorithm to a difficult and important problem in hydrology, the flow of water in the unsaturated zone modeled by Richards' equation [16]. The first two objectives will advance and test the current theory of this algorithm for a simple equation, while the latter objective will provide some evidence of the applicability of this algorithm to a challenging class of applications for which improved solution algorithms are badly needed. We consider the theoretical analysis of this algorithm applied to Richards' equation and other nonlinear models as open issues and beyond the scope of this effort.

The use of smoothed aggregation elements in an additive Schwarz scheme was originally proposed in [2, 3]. In these papers this idea was tested on a variety of elliptic problems with complicated geometries. Construction of coarse mesh problems by aggregation was critical to the results reported in [8, 9, 7], where computations were done on unstructured grids in three space dimensions. Our coarse mesh basis functions are nonsmoothed, which means that the square of the H^1 bound on the basis functions is $O(\frac{H^{d-1}}{\delta})$ instead of $O(H^{d-1})$. Note that we do, for generality, include the possibility of overlap $\delta > h$. The method in [2, 3], however, assumes a physical overlap of $O(h)$ and obtains the benefits of overlap (i.e., bounds on the condition number independent of H and h) by smoothing the basis functions. These differences lead to a change in the energy of the coarse mesh operator so that $|Qu|^2_{H^1} \leq C\frac{H}{\delta}|u|^2_{H^1}$. Thus instead of having a condition number bound by a constant we get a condition bound that is $O(\frac{H^2}{\delta^2})$.

In our previous work [8, 9, 7], we used nonsmoothed aggregation elements with minimal overlap and exact subdomain solves. Similarly, in section 3 the overlap is h, where h is the fine mesh length scale, and we solve the subdomain problems exactly with sparse Gaussian elimination. The convergence results in section 2 allow for more flexibility in overlap and subdomain solvers.

The analysis in section 2 uses the standard finite element framework from [17, 23]. The preconditioner also works well in the context of finite differences, however, as the example in section 3.2 illustrates.

Richards' equation [16] is a model of flow through unsaturated porous media. In this paper we consider the head-based form of the equation and for a homogeneous media in two space dimensions,

$$\left[\frac{\partial \theta}{\partial \psi} + \frac{S_s}{\theta_s}\theta\right]\frac{\partial \psi}{\partial t} = \nabla \cdot \left[K_s k_r \nabla\left(\psi + z\right)\right], \tag{1.1}$$

where ψ is the pressure head, θ is the volume fraction of the wetting phase, and k_r is the relative permeability of the wetting phase. The remaining terms are scalar coefficients given in Table 1.1, along with their values for the test problem. θ and k_r are functions of ψ given by

$$\theta = (\theta_s - \theta_r)(1 + |\alpha\hat{\psi}|^n)^{-m} + \theta_r, \tag{1.2}$$

$$k_r = (1 + |\alpha\hat{\psi}|^n)^{-m/2}[1 - |\alpha\hat{\psi}|^{n-1}(1 + |\alpha\hat{\psi}|^n)^{-m}]^2, \text{ and} \tag{1.3}$$

$$\hat{\psi} = \min(\psi, 0), \tag{1.4}$$

where $m = 1 - 1/n$. These functions are derived from the van Genuchten [20] and Mualem [14] empirical relations among pressure, saturation, and relative permeability. We discretize (1.1) in space with finite differences and integrate the resulting system of differential algebraic equations in time with the fixed leading coefficient backward

Table 1.1
Richards' equation parameters.

Description	Symbol	Value
Saturated volume fraction	θ_s	3.01×10^{-1}
Residual volume fraction	θ_r	9.30×10^{-2}
Specific storage	S_s	1.00×10^{-6} (1/m)
Hydraulic conductivity	K_s	5.04×10^{0} (m/day)
Mean pore size	α	5.47×10^{0} (1/m)
Pore size uniformity	n	4.26×10^{0}

difference formulas of orders one to five [4, 10]. Within the implicit temporal integration is a Newton iteration, and we solve the nonsymmetric linear equation for the Newton step with a preconditioned Bi-CGSTAB [19] linear iteration.

2. Theory for an elliptic model problem.

The convergence theory in this paper is for the weak form of an elliptic boundary value problem with Dirichlet boundary conditions on a domain $\Omega \subset R^d$ with boundary Γ, with spatial dimension d. We will restrict our attention to piecewise linear nodal finite element spaces.

The goal is to find $u \in V$ such that

$$a(u, v) = l(v) \text{ for all } v \in V, \tag{2.1}$$

where a is a strongly elliptic bilinear form on V, l is a linear functional on V, and V is an appropriate function space.

We let $V^h \subset V$ be the appropriate space of piecewise linear functions. The approximating problem at level h is to find $u^h \in V^h$ such that

$$a(u^h, v) = l(v) \text{ for all } v \in V^h. \tag{2.2}$$

The problem (2.2) is equivalent to a linear system

$$Au^h = f \tag{2.3}$$

on V^h, where $a(u, v) = (Au, v)$ for all $u, v \in V^h$. Here $(\cdot, \cdot)$ is the l^2 inner product.

Schwarz preconditioners are designed to accelerate Krylov space iterative methods for the solution of (2.3).

2.1. One-level method.

We begin with the one-level additive Schwarz preconditioner. We divide Ω into subdomains $\{\Omega_j\}_{j=1}^J$ with an overlap width of δ and assume that $\bigcup_{j=1}^J \Omega_j = \Omega$.

Let R_j be the restriction map from an element of V^h to the subspace V_j of functions in V^h with support on Ω_j. Let

$$A_j = R_j A R_j^T$$

be the subdomain operator. We assume that A_j is nonsingular on V_j and define

$$B_j = R_j^T \tilde{A}_j^{-1} R_j,$$

where $\tilde{A}_j$ is an approximation of A_j. The one-level additive Schwarz preconditioner is

$$M = \sum_{j=1}^{J} B_j. \tag{2.4}$$

2.2. Two-level method. The two-level additive Schwarz method adds a coarse mesh term

$$B_0 = R_0^T \tilde{A}_0^{-1} R_0$$

to the one-level preconditioner. Here $\tilde{A}_0$ is an approximation of A_0. We let V^H denote the space of coarse mesh basis functions. If the coarse mesh restriction map R_0 and the coarse mesh operator A_0 are well designed, the condition number of MA is significantly reduced.

One way to define a coarse mesh problem is to discretize the continuous problem on a coarser mesh. There are a few difficulties associated with forming the coarse problem this way. First, for unstructured meshes, such as the ones considered in [7, 9, 8], the interpolation operators between the fine mesh and the coarse mesh are difficult to define. Second, a coarse mesh must be generated, stored, and parallelized. Finally, the PDE must be discretized and recomputed on the coarse mesh.

Alternatively, the discretization of the coarse mesh operator may be defined in terms of the existing fine mesh discretization. A Galerkin or variational coarse grid correction uses the fine grid matrix to obtain the coarse grid operator as $A_0 = R_0 A R_0^T$, where R_0^T is the interpolation operator from the coarse mesh function space, and R_0 is the restriction operator. If the coarse mesh basis functions are obtained from the fine mesh basis functions, then the coarse mesh space V^H is contained in the fine grid space V^h.

In this section we use the aggregation-based basis from [2, 3, 7, 8, 9], where one coarse mesh basis function is defined for each subdomain as the sum of the fine mesh basis functions for that subdomain.

To set the notation that we will need in section 2.4, let the expansion of a function $u \in V^h$ in the finite element basis be

$$u = \sum_l u_l \psi_l, \tag{2.5}$$

where the ψ_l's are the nodal basis functions for the fine mesh. A function $u_C \in V^H$ can be represented on the coarse mesh space as

$$u_C = \sum_K u_{C_K} \Psi_K, \tag{2.6}$$

where the Ψ_K's are the basis functions for the coarse mesh space. Since $V^H \subset V^h$, Ψ_K can be written as

$$\Psi_K = \sum_l R_{Kl} \psi_l. \tag{2.7}$$

The index K represents the subdomain number. The value of $R_{Kl} \geq 0$ is constrained by the conditions that $\nabla \Psi_K = O(\delta)$ and the requirement of the theory that the coarse mesh basis functions provide a partition of unity, i.e.,

$$\sum_K \Psi_K = 1.$$

In the case of minimal overlap, where $\delta = O(h)$, $R_{Kl} = 1$ if the support of the fine mesh basis function ψ_l is contained in subdomain K; otherwise, $R_{Kl} = 0$.

Further expanding the representation of u_C gives

$$\begin{aligned} u_C &= \sum_K u_{C_K} \Psi_K, \\ &= \sum_K u_{C_K} \sum_j R_{Kl} \psi_l, \\ &= \sum_l \left(\sum_K u_{C_K} R_{Kl} \right) \psi_l, \\ &= \sum_l \left(R^T u_C \right)_l \psi_l. \end{aligned}$$

Thus R^T is the operator which interpolates from the coarse mesh to the fine mesh. Any function on the coarse mesh can be represented solely in terms of the already existing fine mesh functions, making the formulation of a separate coarse mesh unnecessary.

2.3. Condition number estimate. In Assumption 2.1 we make precise the idea that H is the characteristic diameter of a subdomain. In Assumption 2.2 we make precise the overlap δ between the subdomains and the properties of the coarse mesh basis functions. These assumptions are based on assumptions given in [2, 3], but ours differ in the fact that an overlap parameter of δ is considered and the square of the energy of our coarse mesh basis function is bound by $\frac{H^{d-1}}{\delta}$ instead of H^{d-2}, where d is the dimension of the problem.

ASSUMPTION 2.1.

1. *There is* $C > 0$ *such that* $diam(\Omega_j) \leq CH$ *for all* $j = 1, \ldots, J$.
2. *There is* $c > 0$ *such that for all* $x \in \Omega$ *there exists* $j \geq 1$ *such that* $x \in \Omega_j$ *and*

$$\text{dist}\,(x, \partial\Omega_j \backslash \partial\Omega) \geq c\delta.$$

3. *There are* $C_R, C_1, C_2 > 0$ *such that for all* $x \in \Omega$ *the ball*

$$B\,(x, C_R H) = \{y \in \Omega : \text{dist}\,(y, x) \leq C_R H\}$$

intersects at most $C_1 + C_2^d$ *subdomains* Ω_j *(i.e., an object of diameter* $O\,(H)$ *intersects at most* $O(1)$ *subdomains* Ω_i*).*

4. $\mu\,(\Omega_j) \geq CH^d$, $j = 1, \ldots, J$.

In Assumption 2.1, μ denotes the Lebesgue measure.

ASSUMPTION 2.2. *Assume the basis functions* Ψ_i *of the coarse space satisfy the following.*

1. $|\Psi_i|^2_{H^1(\Omega)} \leq C\frac{H^{d-1}}{\delta}$,
   $\|\Psi_i\|^2_{L^2} \leq CH^d$.
2. *There is a domain* $\Omega^{int} \subset \Omega$ *such that* $\sum_i \Psi_i\,(x) = 1$ *for every* $x \in \Omega^{int}$ *and* $\text{dist}(x, \partial\Omega) \leq C\delta$ *for every* $x \in \Omega \backslash \Omega^{int}$.
3. $\text{supp}\,(\Psi_i) \subset \bar{\Omega}_i$.

In section 2.4 we prove the following theorem.

THEOREM 2.1. *Let* $V^h = \sum_{j=0}^J V_j \subset C(\bar{\Omega})$, *and let Assumptions* 2.1 *and* 2.2 *hold. Assume that there is* $\omega \geq 1$ *such that*

$$(v, v) \leq (\tilde{A}_j^{-1} A_j v, v) \leq \omega(v, v)$$

for all $v \in V^h$ *and* $0 \le j \le J$. *Let*

$$M = \sum_{j=0}^{J} B_j; \tag{2.8}$$

then there is $C > 0$, *independent of* H *and* h *such that*

$$\kappa(MA) \le C\omega(1 + (H/\delta)^2). \tag{2.9}$$

2.4. Convergence theory. The bound on the condition number for the system preconditioned using two-level additive Schwarz methods is given in [23, 22]. We use this theory to calculate our bound.

THEOREM 2.2. *Let* K_0 *be a positive constant so that, for any* $v \in V^h$, *there exists a decomposition* $v = \sum_{i=0}^{J} v_i$ *such that* $v_i \in V_i$ *and*

$$\sum_{i=0}^{J} (A_i v_i, v_i) \le K_0 (Av, v) . \tag{2.10}$$

Let

$$K_1 = \max_{1 \le j \le J} \sum_{i=1}^{J} \varepsilon_{ij}, \tag{2.11}$$

where, for $1 \le i, j \le J$, $\varepsilon_{ij} = 0$ *if* $V_i \perp V_j$; $\varepsilon_{ij} = 1$ *otherwise. Then*

$$\kappa (MA) \le \omega K_0 (1 + K_1) , \tag{2.12}$$

where

$$\omega = \max_{0 \le j \le J} \lambda_{\max} \left(\tilde{A}_j^{-1} A_j \right) .$$

We assume that the energy norm is equivalent to the H^1 seminorm, and we can therefore replace (2.10) by

$$\sum_{i=0}^{J} |u_i|^2_{H^1(\Omega)} \le K_0 |u|^2_{H^1(\Omega)}. \tag{2.13}$$

Our estimate for K_0 will be based on (2.13).

The value of K_1 is an indicator of the number of subdomains which contain any given point in Ω; we assume our partition is such that $K_1 = O(1)$. We solve the subdomain problems exactly in our numerical results, so $\tilde{A}_j = A_j$ for all j and $\omega = 1$ in section 3. Thus our condition number estimate is based on the estimate of K_0, which we obtain using Lemmas 2.3 and 2.4.

We define the coarse mesh projection $Q : V^h \to V^H$ by

$$Qu = \sum_{i=1}^{J} \alpha_i \Psi_i, \quad \alpha_i = \alpha_i (u) = \frac{1}{\mu (\Omega_i)} \int_{\Omega_i} u (x) \, dx,$$

where $u \in V^h$.

The value of K_0 depends on the bound of the energy of Qu and on the L^2 bound of the error in the coarse mesh operator, i.e., $u - Qu$. These bounds are provided in Corollary 2.3. This analysis is a modification of the results in [2, 3] for nonsmoothed aggregates.

Here, as in the remainder of this section, C is a constant that is independent of H and h. C may increase as the analysis progresses.

COROLLARY 2.3. *If Assumptions* 2.1 *and* 2.2 *hold, then*

$$\|u - Qu\|_{L^2}^2 \leq CH^2 |u|_{H^1}^2 , \tag{2.14}$$

$$|Qu|_{H^1}^2 \leq C\frac{H}{\delta} |u|_{H^1}^2 . \tag{2.15}$$

Proof. We give here the main results where our work differs from that in [2, 3], and we refer the reader to the details of the proof in those papers.

Since we have $\|\Psi_i\|_{L^2}^2 \leq CH^d$ by construction of our coarse mesh basis functions, we get

$$\|Qu\|_{L^2(\Omega)}^2 \leq C \|u\|_{L^2(\Omega)}^2 .$$

Therefore, by Poisson's inequality [6],

$$\|u - Qu\|_{L^2(\Omega)} \leq \|u\|_{L^2(\Omega)} + \|Qu\|_{L^2(\Omega)} \leq C \|u\|_{L^2(\Omega)} \leq CH |u|_{H^1(\Omega)} . \tag{2.16}$$

Since by construction we also have $|\Psi_i|_{H^1}^2 \leq C\frac{H^{d-1}}{\delta}$, we obtain

$$|Qu|_{H^1(\Omega)}^2 \leq C\tfrac{H}{\delta} |u|_{H^1(\Omega)}^2 . \qquad \square \tag{2.17}$$

We use these bounds in the following lemma.

LEMMA 2.4. *Under Assumptions* 2.1 *and* 2.2*, for every finite element function $u \in V^h$, there exists a decomposition $\{u_i\}_{i=0}^J$, $u_i \in V_i$, such that*

$$u = \sum_{i=0}^{J} u_i, \tag{2.18}$$

$$\sum_{i=0}^{J} |u_i|_{H^1(\Omega)}^2 \leq C \left(1 + \frac{H}{\delta}\right)^2 |u|_{H^1(\Omega)}^2 . \tag{2.19}$$

Proof. Define the fine mesh pointwise projection $I_h : C(\Omega) \to V^h$ by

$$I_h (u) = \sum_{l=1}^{n} u (x_l) \psi_l,$$

where $\{\psi_l\}_{l=1}^n$ is the finite element basis on the fine mesh, and $\{x_l\}_{l=1}^n$ are the fine mesh nodal points. Let u be partitioned such that

$$u_0 = Qu \text{ and } u_i = I_h (\theta_i(u - Qu)) ,$$

where, as in [17], $\{\theta_i\}$ is a smooth partition of unity such that

$$\begin{aligned} \theta_i &= 1 \text{ if } x \in \Omega_i \cup \Omega^{int},\ x \notin \Omega_j \text{ for } j \neq i, \text{ and} \\ \theta_i &= 0 \text{ if } x \notin \Omega_i. \end{aligned}$$

Hence $|\nabla\theta_i| \le \frac{1}{\delta}$. Clearly, by construction, for all $u \in V^h$,

$$\sum_{i=0}^{J} u_i = u.$$

The standard arguments [17] imply that

$$\sum_{i=0}^{J} |u_i|^2_{H^1(\Omega)} \le C\left(\frac{1}{\delta^2}\|u - Qu\|^2_{L^2(\Omega)} + |u - Qu|^2_{H^1(\Omega)} + |Qu|^2_{H^1(\Omega)}\right). \tag{2.20}$$

We complete the proof by applying Lemma 2.3 to estimate $|Qu|^2_{H^1(\Omega)}$ and $\|u - Qu\|^2_{L^2(\Omega)}$. □

If we let

$$K_0 = C\left(1 + \frac{H}{\delta}\right)^2,$$

we obtain the result in Theorem 2.1.

3. Numerical results.

In the two examples we used minimal overlap. Hence the coarse mesh problem is constructed by simply adding the matrix contributions for the nodes in a subdomain. The matrix representation for the restriction operator R_0 given minimal overlap h is

$$R_0 = \begin{bmatrix} 1 & 1 & \ldots & 1 & 0 & 0 & 0 & 0 & 0 & \ldots & 0 \\ 0 & 0 & \ldots & 0 & 1 & 1 & \ldots & 1 & 0 & \ldots & 0 \\ \vdots & & & & & & & & & & \end{bmatrix},$$

where the length of 1's in row i is determined by the number of unknowns in subdomain i.

Subdomain solves were exact, meaning that $\tilde{A}_j = A_j$ for all j and, therefore, that $\omega = 1$.

3.1. Laplace equation.

In this section we consider the simple test problem

$$\nabla^2 u = 0 \tag{3.1}$$

on the unit square $[0,1] \times [0,1]$ with zero Dirichlet boundary conditions. We use the function identically equal to one on the mesh as the initial iterate for a preconditioned conjugate gradient iteration. We terminated the iterations when the residual had been reduced by a factor of 10^{-4}. This example fully conforms to the theory from section 2.4.

In Table 3.1 we report results for a piecewise linear finite element discretization of (3.1). From the theory, one would expect the condition number κ to increase by a factor of 4 as either H is doubled or h is halved and to remain constant along the diagonals where H/h is constant. If, as is the case with the unpreconditioned problem, the iteration count is $O(\sqrt{\kappa})$, we would expect the iteration count to double if either H is held fixed and h is reduced by a factor of two or h is held fixed and H is increased by a factor of two.

If the iteration count is proportional to the square root of the condition number, as it is in the unpreconditioned case, then Table 3.1 shows that the growth in the number of iterations for fixed H and decreasing h is slower than predicted by the theory and is consistent with an $O(H/h)$ growth in the condition number. The rate of reduction in the iteration count as h is held fixed and as H is decreased is smaller still.

TABLE 3.1
Finite element discretization of Laplace's equation.

H \ h	1/64	1/128	1/256
1/4	37	51	68
1/8	32	44	61
1/16	26	36	49
1/32		26	37

3.2. Richards' equation. The test domain is the unit square $[0, 1m] \times [0, 1m]$ with boundary and initial conditions

$$\begin{array}{rclll} \psi(x,0) & = & 0.0, & x \in [0,1], & t > 0, \\ \psi(x,1) & = & 0.1, & x \in [1/3, 2/3], & t > 0, \\ \dfrac{\partial \psi}{\partial z}(x,1) & = & -1.0, & x \in [0,1/3) \cup (2/3,1], & t > 0, \\ \dfrac{\partial \psi}{\partial x}(x,z) & = & 0.0, & x = 0,1 \quad z \in [0,1], & t > 0, \\ \psi(x,z) & = & -z, & x,z \in [0,1] \times [0,1], & t = 0. \end{array} \tag{3.2}$$

3.2.1. Finite difference discretization with minimal overlap. We discretized (1.1) by applying cell-centered finite differences to the spatial operator, thereby yielding the system of differential-algebraic equations,

$$\begin{aligned} F_{i,j}(t, \psi, \frac{\partial \psi}{\partial t}) = & \left(\frac{d\theta}{d\psi}_{i,j} + \frac{S_s}{\theta_s} \theta_{i,j} \right) \frac{\partial \psi_{i,j}}{\partial t} \\ & - \frac{1}{\Delta z^2} \left[K_{i+\frac{1}{2},j}(\psi_{i+1,j} - \psi_{i,j}) - K_{i-\frac{1}{2},j}(\psi_{i,j} - \psi_{i-1,j}) \right] \\ & - \frac{1}{\Delta z}(K_{i+\frac{1}{2},j} - K_{i-\frac{1}{2},j}) \\ & - \frac{1}{\Delta x^2} \left[K_{i,j+\frac{1}{2}}(\psi_{i,j+1} - \psi_{i,j}) - K_{i,j-\frac{1}{2}}(\psi_{i,j} - \psi_{i,j-1}) \right], \end{aligned} \tag{3.3}$$

where $i = 0, \ldots, N-1$, $j = 0, \ldots, N-1$, $\Delta z = \Delta x = 1/N$, and

$$K_{i\pm\frac{1}{2},j} = \left[(K_s k_r)_{i\pm1,j} + (K_s k_r)_{i,j} \right] /2, \tag{3.4}$$

$$K_{i,j\pm\frac{1}{2}} = \left[(K_s k_r)_{i,j\pm1} + (K_s k_r)_{i,j} \right] /2. \tag{3.5}$$

The semidiscrete system was integrated in time over [0, 0.0149 days]. Order and step-size were selected via local truncation error estimates, and the local truncation error tolerance was set to $10\Delta x^2$, thereby balancing temporal and spatial truncation error. A secondary effect of this choice is that the number of iterations needed for convergence grows more slowly as Δx is reduced than would be the case for a steady-state problem. This effect is clearly visible in all the results reported in Tables 3.2, 3.3, and 3.4.

At a given step t_{n+1}, the application of the integration method yielded a nonlinear system of the form

$$F[t_{n+1}, \psi_{n+1}, g(\psi_{n+1})] = G(\psi_{n+1}) = 0,$$

where $g(\psi)$ is a the backward difference formula for $\partial\psi/\partial t$. We solved the nonlinear system with an inexact Newton iteration that terminated when the 2-norm of the nonlinear residual was reduced by a factor of 10^{-5}.

Table 3.2
Richards' equation iteration statistics, 2-level Schwarz.

$H\backslash h$	1/16	1/32	1/64	1/128	1/256
1/8	7	8	9	12	15
1/16		7	9	11	14
1/32			7	9	11
1/64				7	9
1/128					7

Table 3.3
Richards' equation iteration statistics, 1-level Schwarz.

$H\backslash h$	1/16	1/32	1/64	1/128	1/256
1/8	5	6	6	6	6
1/16	6	7	7	7	7
1/32		10	10	10	10
1/64			15	14	14
1/128				21	20
1/256					29

Table 3.4
Richards' equation iteration statistics, no preconditioner.

h	1/16	1/32	1/64	1/128	1/256
	31	56	84	129	193

At each Newton iteration we obtained the Newton step δ^{m+1}, by solving the linear system

$$\left[\frac{\partial G}{\partial \psi}(\psi^m_{n+1})\right]\delta^{m+1} = -G(\psi^m_{n+1})$$

with scaled, preconditioned BiCGstab. The scaling was obtained from the integration method's weighted root mean squared norm [15, 10, 1, 4]. In real applications, such a scaling would allow termination of the linear iteration according to tolerances specified by the integration scheme. However, for this test we computed the l^2 norm of the true linear residual for each linear iteration and terminated the linear iteration when that norm had been reduced by a factor of 10^{-7}. We did this both to more accurately estimate the effects of the preconditioner and to insure that errors in the Newton step were insignificant with respect to the Newton iteration and integration. The choices of termination criteria for the linear and nonlinear solvers imply that the time steps that the code uses are independent of the choice of preconditioner.

The preconditioner was two-level additive Schwarz using (3.1). The subdomains had the minimal overlap of $\Delta x = 1/N$. Table 3.2 gives the average BiCGstab iterations per Newton iteration for two-level additive Schwarz. The iteration count is constant as H and h are reduced simultaneously, as was the case with the simple example from section 3.1.

The iteration counts in Table 3.2 increase more slowly than the theory would predict if, as would be the case with a discretized elliptic problem and conjugate gradient iteration, the iteration count increased as the square of the condition number. In that case, if the condition number is $O(1 + H^2/h^2)$, then the iteration count would double as either H doubled or as h was halved.

For comparison we include similar statistics for a one-level additive Schwarz preconditioner in Table 3.3 and for no preconditioning in Table 3.4. Point Jacobi preconditioning is the $H = h$ diagonal in Table 3.3. Neither of these scale well as H and h are reduced together.

4. Conclusions. In the context of Richards' equation, a model for subsurface flow through the unsaturated zone, we have demonstrated the effectiveness of the two-level additive Schwarz preconditioner using nonsmoothed aggregates for the coarse space on a temporally dependent, nonlinear problem. Our convergence estimates for a simpler model problem are consistent with the observations for Richards' equation, a problem to which the theory does not apply.

Acknowledgments. We wish to thank Marian Brezina, Jim Jones, Van Henson, David Keyes, Andrea Toselli, Mary Wheeler, Carol Woodward, and Jun Zou for many helpful discussions and Dave Klepacki, Matt Matthews, Mark Reed, Eric Sills, and Bob Walkup for their help with the computations. We are particularly grateful to our collaborators at the U.S. Army Engineer Research and Development Center: Charlie Berger, Jackie Hallberg, Stacy Howington, John Peters, Joe Schmidt, and Alan Stagg. We also wish to thank the referees for their helpful suggestions.

REFERENCES

[1] K. E. Brenan, S. L. Campbell, and L. R. Petzold, *Numerical Solution of Initial Value Problems in Differential-Algebraic Equations*, Classics Appl. Math. 14, SIAM, Philadelphia, 1996.

[2] M. Brezina, *Robust Iterative Methods on Unstructured Meshes*, Ph.D. thesis, University of Colorado at Denver, Denver, CO, 1997.

[3] M. Brezina and P. Vaněk, *A black-box iterative solver based on a two-level Schwarz method*, Computing, 63 (1999), pp. 233–263.

[4] P. N. Brown, A. C. Hindmarsh, and L. R. Petzold, *Using Krylov methods in the solution of large-scale differential-algebraic systems*, SIAM J. Sci. Comput., 15 (1994), pp. 1467–1488.

[5] T. F. Chan, S. Go, and J. Zou, *Boundary treatments for multilevel methods on unstructured methods*, SIAM J. Sci. Comput., 21 (1999), pp. 46–66.

[6] L. C. Evans, *Partial Differential Equations*, AMS, Providence, RI, 1998.

[7] S. E. Howington, R. C. Berger, J. P. Hallberg, J. F. Peters, A. K. Stagg, E. W. Jenkins, and C. T. Kelley, *A Model to Simulate the Interaction Between Groundwater and Surface Water*, Tech. rep. CRSC-TR99-27, North Carolina State University, Center for Research in Scientific Computation, 1999, in Proceedings of the High Performance Computing Users' Group Meeting, Monterrey, CA, 1999, CD-ROM, High Performance Computing Users' Group, 1999.

[8] E. W. Jenkins, R. C. Berger, J. P. Hallberg, S. E. Howington, C. T. Kelley, J. H. Schmidt, A. Stagg, and M. D. Tocci, *A two-level aggregation-based Newton-Krylov-Schwarz method for hydrology*, in Parallel Computational Fluid Dynamics 1999, D. E. Keyes, A. Ecer, J. Periaux, and N. Satofuka, eds., North Holland, Amsterdam, 2000, pp. 257–264.

[9] E. W. Jenkins, R. C. Berger, J. P. Hallberg, S. E. Howington, C. T. Kelley, J. H. Schmidt, A. Stagg, and M. D. Tocci, *Newton-Krylov-Schwarz Methods for Richards' equation*, Tech. rep. CRSC-TR99-32, North Carolina State University, Center for Research in Scientific Computation, 1999, submitted.

[10] C. E. Kees and C. T. Miller, *C++ implementations of numerical methods for solving differential-algebraic equations: Design and optimization considerations*, ACM Trans. Math. Software, 25 (2000), pp. 377–403.

[11] J. Mandel, *Balancing domain decomposition*, Comm. Appl. Numer. Methods Engrg., 9 (1993), pp. 233–241.

[12] J. Mandel, *Hybrid domain decomposition with unstructured subdomains*, in Domain Decomposition Methods in Science and Engineering (Como, 1992), Contemp. Math. 157, AMS, Providence, RI, 1994, pp. 103–112.

[13] J. Mandel and M. Brezina, *Balancing Domain Decomposition: Theory and Performance in Two and Three Dimensions*, University of Colorado at Denver/Center for Computational Mathematics report 2, 1993.

[14] Y. Mualem, *A new model for predicting the hydraulic conductivity of unsaturated porous media*, Water Resources Research, 12 (1976), pp. 513–522.

[15] K. Radhakrishnan and A. C. Hindmarsh, *Description and Use of LSODE, the Livermore Solver for Ordinary Differential Equations*, Tech. rep. URCL-ID-113855, Lawrence Livermore National Laboratory, Livermore, CA, 1993.

[16] L. A. Richards, *Capillary conduction of liquids through porous media*, Physics, 1 (1931), pp. 318–333.

[17] B. Smith, P. Bjørstad, and W. Gropp, *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge University Press, Cambridge, UK, 1996.

[18] K. Staheli, J. H. Schmidt, and S. Swift, *Guidelines for Solving Groundwater Problems with ADH*, manuscript, 1998.

[19] H. A. van der Vorst, *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric systems*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 631–644.

[20] M. T. van Genuchten, *Predicting the hydraulic conductivity of unsaturated soils*, Soil Science Society of America Journal, 44 (1980), pp. 892–898.

[21] P. Vaněk, J. Mandel, and M. Brezina, *Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems*, Computing, 56 (1996), pp. 179–196.

[22] J. Xu, *Iterative methods by space decomposition and subspace correction*, SIAM Rev., 34 (1992), pp. 581–613.

[23] J. Xu and J. Zou, *Some nonoverlapping domain decomposition methods*, SIAM Rev., 40 (1998), pp. 857–914.

SIAM J. SCI. COMPUT.
Vol. 23, No. 2, pp. 442–462

ON THE CONSTRUCTION OF DEFLATION-BASED PRECONDITIONERS*

J. FRANK† AND C. VUIK‡

Abstract. In this article we introduce new bounds on the effective condition number of deflated and preconditioned-deflated symmetric positive definite linear systems. For the case of a subdomain deflation such as that of Nicolaides [*SIAM J. Numer. Anal.*, 24 (1987), pp. 355–365], these theorems can provide direction in choosing a proper decomposition into subdomains. If grid refinement is performed, keeping the subdomain grid resolution fixed, the condition number is insensitive to the grid size. Subdomain deflation is very easy to implement and has been parallelized on a distributed memory system with only a small amount of additional communication. Numerical experiments for a steady-state convection-diffusion problem are included.

Key words. deflation, preconditioners, optimal methods, parallel computing, conjugate gradients

AMS subject classifications. 65F10, 65F50, 65N22

PII. S1064827500373231

1. Background: Preconditioning and deflation.

It is well known that the convergence rate of the conjugate gradient method is bounded as a function of the condition number of the system matrix to which it is applied. Let $A \in \mathbb{R}^{n\times n}$ be symmetric positive definite. We assume that the vector $f \in \mathbb{R}^n$ represents a discrete function on a grid Ω and that we are searching for the vector $u \in \mathbb{R}^n$ on Ω which solves the linear system

$$Au = f.$$

Such systems are encountered, for example, when a finite volume/difference/element method is used to discretize an elliptic partial differential equation (PDE) defined on the continuous analogue of Ω. In particular our goal is to develop efficient serial and parallel methods for applications in incompressible fluid dynamics; see [28, 27].

Let us denote the spectrum of A by $\sigma(A)$ and the ith eigenvalue in nondecreasing order by $\lambda_i(A)$ or simply by λ_i when it is clear to which matrix we are referring. After k iterations of the conjugate gradient method, the error is bounded by (cf. [10, Thm. 10.2.6]

$$\|u - u_k\|_A \leq 2\,\|u - u_0\|_A \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k, \tag{1.1}$$

where $\kappa = \kappa(A) = \lambda_n/\lambda_1$ is the spectral condition number of A and the A-norm of u is given by $\|u\|_A = (u^T A u)^{1/2}$. The error bound (1.1) does not tell the whole story, however, because the convergence may be significantly faster if the eigenvalues of A are clustered [23].

*Received by the editors June 6, 2000; accepted for publication September 27, 2000; published electronically July 10, 2001.

http://www.siam.org/journals/sisc/23-2/37323.html

†CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands (jason@cwi.nl). The research of this author was supported in part by Delft University of Technology.

‡Delft University of Technology, Faculty of Information Technology and Systems, Department of Applied Mathematical Analysis, P.O. Box 5031, 2600 GA Delft, The Netherlands (vuik@math.tudelft.nl).

When A is the discrete approximation of an elliptic PDE, the condition number can become very large as the grid is refined, thus slowing down convergence. In this case it is advisable to solve, instead, a preconditioned system $K^{-1}Au = K^{-1}f$, where the symmetric positive definite preconditioner K is chosen such that $K^{-1}A$ has a more clustered spectrum or a smaller condition number than that of A. Furthermore, K must be cheap to solve relative to the improvement it provides in convergence rate. A final desirable property in a preconditioner is that it should parallelize well, especially on distributed memory computers. Probably the most effective preconditioning strategy in common use is to take $K = LL^T$ to be an incomplete Cholesky (IC) factorization of A [18]. For discretizations of second order PDEs in two dimensions, defined on a grid with spacing h, one finds, with IC factorization, $\kappa \sim h^{-2}$; with a modified IC factorization [11, 1], $\kappa \sim h^{-1}$; and with a multigrid cycle, $\kappa \sim 1$. Preconditioners such as multigrid and some domain decomposition methods, for which the condition number of the preconditioned system is independent of the grid size, are termed *optimal*.

Another preconditioning strategy that has proven successful when there are a few isolated extremal eigenvalues is *deflation* [20, 16, 17]. Let us define the projection P by

$$P = I - AZ(Z^TAZ)^{-1}Z^T, \quad Z \in \mathbb{R}^{n\times m}, \tag{1.2}$$

where Z is the deflation subspace, i.e., the space to be projected out of the residual, and I is the identity matrix of appropriate size. We assume that $m \ll n$ and that Z has rank m. Under this assumption $A_c \equiv Z^TAZ$ may be easily computed and factored and is symmetric positive definite. Since $u = (I - P^T)u + P^Tu$ and because

$$(I - P^T)u = Z(Z^TAZ)^{-1}Z^TAu = ZA_c^{-1}Z^Tf \tag{1.3}$$

can be immediately computed, we need only compute P^Tu. In light of the identity $AP^T = PA$, we can solve the deflated system

$$PA\tilde{u} = Pf \tag{1.4}$$

for $\tilde{u}$ using the conjugate gradient method and premultiply this by P^T. Obviously (1.4) is singular, and this raises a few questions. First, the solution $\tilde{u}$ may contain an arbitrary component in the null space of PA, i.e., in span$\{Z\}$.[1] This is not a problem, however, because the projected solution $P^T\tilde{u}$ is unique. Second, what consequences does the singularity of (1.4) imply for the conjugate gradient method?

Kaasschieter [14] notes that a positive semidefinite system can be solved as long as the right-hand side is consistent (i.e., as long as $f = Au$ for some u). This is certainly true for (1.4), where the same projection is applied to both sides of the nonsingular system. Furthermore, he notes (with reference to [23]) that because the null space never enters the iteration, the corresponding zero-eigenvalues do not influence the convergence. Motivated by this fact, we define the *effective condition number* of a positive semidefinite matrix $C \in \mathbb{R}^{n\times n}$ with corank m to be the ratio of its largest to smallest *positive* eigenvalues:

$$\kappa_{\text{eff}}(C) = \frac{\lambda_n}{\lambda_{m+1}}.$$

[1] We will use the notation span$\{Z\}$ to denote the column space of Z.

Example. To see that the condition number of PA may be better than that of A, consider the case in which Z is the invariant subspace of A corresponding to the smallest eigenvalues. Note that $PAZ = 0$, so that PA has m zero-eigenvalues. Furthermore, since A is symmetric positive definite, we may choose the remaining eigenspace Y in the orthogonal complement of span$\{Z\}$, i.e., $Y^T Z = 0$ so that $PY = Y$. However, $AY = YB$ for some invertible B; therefore, $PAY = PYB = YB$, and span$\{Y\}$ is an invariant subspace of PA. Evidently, when Z is an invariant subspace of A,

$$\kappa_{\text{eff}}(PA) = \frac{\lambda_n(A)}{\lambda_{m+1}(A)}.$$

In summary, deflation of an invariant subspace cancels the corresponding eigenvalues, leaving the rest of the spectrum untouched.

This idea has been exploited by several authors. For nonsymmetric systems, approximate eigenvectors can be extracted from the Krylov subspace produced by GMRES. Morgan [19] uses this approach to improve the convergence after a restart. In this case, deflation is not applied as a preconditioner, but the deflation vectors are augmented with the Krylov subspace, and the minimization property of GMRES ensures that the deflation subspace is projected out of the residual. For more discussion on deflation methods for nonsymmetric systems, see [15, 8, 6, 21, 5, 2]. Other authors have attempted to choose a subspace a priori that effectively represents the slowest modes. In [29] deflation is used to remove a few stubborn but *known* modes from the spectrum. Mansfield [16] shows how Schur-complement-type domain decomposition methods can be seen as a series of deflations. Nicolaides [20] chooses Z to be a piecewise constant interpolation from a set of m subdomains and points out that deflation might be effectively used with a conventional preconditioner. Mansfield [17] uses the same "subdomain deflation" in combination with damped Jacobi smoothing, obtaining a preconditioner which is related to the two-grid method.

In this article we introduce new bounds on the effective condition number of deflated and preconditioned-deflated symmetric positive definite linear systems. For the case of a subdomain deflation such as that of Nicolaides [20], these theorems can provide direction in choosing a proper decomposition into subdomains. If grid refinement is done keeping the subdomain grid resolution fixed, the condition number is insensitive to the grid size. Subdomain deflation is very easy to implement and has been parallelized on a distributed memory system with only a small amount of additional communication. Numerical experiments for a steady-state convection-diffusion problem are included.

2. A condition number bound for deflation.

Nicolaides [20] proves the following bound on the spectrum of PA:

$$\lambda_{m+1}(PA) = \min \frac{v^T v}{v^T A^{-1} v}, \quad \lambda_n(PA) = \max \frac{v^T v}{v^T A^{-1} v},$$

where v is taken in span$\{Z\}^\perp$. In this section we give a bound of a different flavor which will be used in the subsequent sections to construct a preconditioning strategy with an optimal convergence property.

First we need the following result on the preservation of positive semidefiniteness under deflation.

LEMMA 2.1. *Let R be positive semidefinite and P be a projection ($P^2 = P$); then if PR is symmetric, it is positive semidefinite.*

Proof. By hypothesis, $0 \leq u^T R u$ for all u. In particular, $0 \leq (P^T u)^T R (P^T u) = u^T P R P^T u$ so that $PRP^T = P^2 R = PR$ is positive semidefinite. $\square$

The next theorem provides a bound on the condition number of PA and is our main result.

THEOREM 2.2. *Let A be symmetric positive definite, let P be defined by* (1.2), *and suppose there exists a splitting $A = C + R$ such that C and R are symmetric positive semidefinite with $\mathcal{N}(C) = \text{span}\{Z\}$ the null space of C. Then*

$$\lambda_i(C) \leq \lambda_i(PA) \leq \lambda_i(C) + \lambda_{\max}(PR). \tag{2.1}$$

Moreover, the effective condition number of PA is bounded by

$$\kappa_{\text{eff}}(PA) \leq \frac{\lambda_n(A)}{\lambda_{m+1}(C)}. \tag{2.2}$$

Proof. From (1.2) it is obvious that PA is symmetric. Since Z is in the null space of C, we have that $PC = C$ and is therefore also symmetric by hypothesis. Symmetry of $PR = PA - C$ follows immediately; and by assumption R is positive semidefinite, so we can apply Lemma 2.1 to arrive at $\lambda_{\min}(PR) \geq 0$, with equality holding in any case due to singularity of P. The bound (2.1) now follows from Theorem 8.1.5 of [10]:

$$\lambda_i(PC) + \lambda_{\min}(PR) \leq \lambda_i(PA) \leq \lambda_i(PC) + \lambda_{\max}(PR).$$

Furthermore, because $PA = A - AZ(Z^T AZ)^{-1}(AZ)^T$ is the difference of positive (semi-)definite matrices, the same theorem (Theorem 8.1.5 of [10]) gives $\lambda_{\max}(PA) \leq \lambda_{\max}(A)$. This upper bound together with the lower bound in (2.1) proves (2.2). $\square$

There is also a preconditioned version of the previous theorem.

THEOREM 2.3. *Assume the conditions of Theorem* 2.2 *and let K be a symmetric positive definite preconditioner with Cholesky factorization $K = LL^T$. Then*

$$\lambda_i(L^{-1}CL^{-T}) \leq \lambda_i(L^{-1}PAL^{-T}) \leq \lambda_i(L^{-1}CL^{-T}) + \lambda_{\max}(L^{-1}PRL^{-T}), \tag{2.3}$$

and the effective condition number of $L^{-1}PAL^{-T}$ is bounded by

$$\kappa_{\text{eff}}(L^{-1}PAL^{-T}) \leq \frac{\lambda_n(L^{-1}AL^{-T})}{\lambda_{m+1}(L^{-1}CL^{-T})}. \tag{2.4}$$

Proof. Define $\hat{A} = L^{-1}AL^{-T}$, $\hat{C} = L^{-1}CL^{-T}$, $\hat{R} = L^{-1}RL^{-T}$ (all congruence transformations), $\hat{Z} = L^T Z$, and

$$\hat{P} = I - \hat{A}\hat{Z}(\hat{Z}^T \hat{A}\hat{Z})^{-1}\hat{Z}^T = L^{-1}PL.$$

Note that $\hat{P}$ is a projection and $\hat{P}\hat{A}$ is symmetric, and also that $\hat{Z}$ is in the null space of $\hat{C}$ so that $\hat{P}\hat{C} = \hat{C}$. Thus, Theorem 2.2 applies directly to the deflated system matrix $\hat{P}\hat{A}$. The conclusions follow immediately from the definitions of $\hat{A}$ and $\hat{C}$. $\square$

Remark. Experience with discretized PDEs indicates that the greatest improvement in convergence is obtained by removing the smallest eigenvalues from the spectrum. It is therefore the lower bounds of (2.1) and (2.3) which are of most concern. Theorem 2.3 suggests that it might be better to construct a preconditioner for C rather than for A in this case. However, care should be taken that a good preconditioner for C does not increase the upper bound in (2.3) when applied to A. See Kaasschieter [14] for a discussion about preconditioning indefinite systems.

In the next section we consider applications of Theorems 2.2 and 2.3 in lieu of a specific choice of the subspace of deflation Z.

3. Subdomain deflation. The results of the previous section are independent of the choice of deflation subspace Z in (1.2). As mentioned in section 1, deflation of an eigenspace cancels the corresponding eigenvalues without affecting the rest of the spectrum. This has led some authors to try to deflate with "nearly invariant" subspaces obtained during the iteration, and led others to try to choose in advance subspaces which represent the extremal modes.

For the remainder of this article we make a specific choice for the subspace Z in (1.2), based on a decomposition of the domain Ω with index set $\mathcal{I} = \{i \,|\, u_i \in \Omega\}$ into m nonoverlapping subdomains Ω_j, $j = 1, \ldots, m$, with respective index sets $\mathcal{I}_j = \{i \in \mathcal{I} \,|\, u_i \in \Omega_j\}$. We assume that the Ω_j are simply connected graphs covering Ω. Define Z by

$$z_{ij} = \begin{cases} 1, & i \in \mathcal{I}_j, \\ 0, & i \notin \mathcal{I}_j. \end{cases} \tag{3.1}$$

With this choice of Z, the projection (1.2) will be referred to as *subdomain deflation*. Such a deflation subspace has been used by Nicolaides [20] and Mansfield [16, 17].

This choice of deflation subspace is related to domain decomposition and multigrid methods. The projection P can be seen as a subspace correction in which each subdomain is agglomerated into a single cell; see, for example, [13]. Within the multigrid framework, P can be seen as a coarse grid correction using a piecewise constant interpolation operator with very extreme coarsening.

Note that the matrix $A_c = Z^T AZ$, the projection of A onto the deflation subspace Z, has sparsity pattern similar to that of A. We will see that the effective condition number of PA improves as the number of subdomains is increased (for a fixed problem size). However, this implies that the dimension of A_c also increases, making direct solution expensive. By analogy with multigrid, it might be advantageous in some applications to solve A_c recursively.[2] In a parallel implementation this would lead to additional idle processor time, as it does with multigrid.

3.1. Application to Stieltjes matrices. Using subdomain deflation, we can identify matrices C and R needed for application of Theorems 2.2 and 2.3 to the class of irreducibly diagonally dominant Stieltjes matrices (i.e., symmetric M-matrices). Such matrices commonly arise as a result of discretization of symmetric elliptic and parabolic PDEs. For our purposes the following characteristics are important:

- A is symmetric positive definite and irreducible.
- $a_{ii} > 0$, $a_{ij} \leq 0$ for $i \neq j$.
- $a_{ii} + \sum_{j \neq i} a_{ij} \geq 0$ with strict inequality holding for some i.

For a matrix A, define the subdomain block-Jacobi matrix $B(A) \in \mathbb{R}^{n \times n}$ associated to A by

$$b_{ij} = \begin{cases} a_{ij} & \text{if } i, j \in \mathcal{I}_k, \text{ for some } k, \\ 0 & \text{otherwise.} \end{cases} \tag{3.2}$$

Notice that since each block B_{jj} is a principle submatrix of A, it is symmetric positive definite. Also, since B is obtained from A by deleting off-diagonal blocks containing only negative elements, the B_{jj} are at least as diagonally dominant as the corresponding rows of A. Furthermore, the irreducibility of A implies that A itself cannot be

[2] A referee pointed out to us that the two-level method with direct solution of A_c has suboptimal complexity. On the other hand, for the examples considered in this article, A_c is too small for a second coarsening.

written in block diagonal form, so to construct B it is necessary to delete at least one nonzero block from each block-row. As a result, at least one row of each B_{jj} is strictly diagonally dominant. We will further assume that the so-constructed B_{jj} are irreducible.[3] It follows from Corollary 6.4.11 of [12] that the B_{jj} are again Stieltjes matrices.

Additionally, define $\mathbf{1} = (1, \ldots, 1)^T$ with the dimension following from the context, such that $A\mathbf{1}$ is the vector of row sums of A. Let the matrix C be defined by

$$C = B - \operatorname{diag}(B\mathbf{1})\,. \tag{3.3}$$

Each block C_{jj} of C has zero row sums—so $\mathbf{1}$ is in the null space of each block—but is further irreducible and weakly diagonally dominant and has the M-matrix property. According to Theorem 4.16 of [3], a singular M-matrix has a null space of rank exactly 1. It follows that the matrix Z defined by (3.1) is a basis for the null space of C.

Putting these ideas together we formulate the following.

Theorem 3.1. *If A is an irreducibly diagonally dominant Stieltjes matrix and C defined by* (3.3) *has only irreducible blocks, then the hypotheses of Theorem* 2.2 *are met.*

Example. Consider a Poisson equation on the unit square with homogeneous Dirichlet boundary conditions

$$-\Delta u = f, \quad u = 0, u \in \partial\Omega, \quad \Omega = [0,1] \times [0,1]. \tag{3.4}$$

The problem is discretized using central finite differences on a 9×9 grid, and subdomain deflation is applied with a 3×3 decomposition into blocks of resolution 3×3. The system matrix A is pre- and postmultiplied by the inverse square root of its diagonal. Figure 3.1 shows the eigenvalues of A, PA, and C. The extreme positive eigenvalues of these three matrices are

	$\lambda_{\min}$	$\lambda_{\max}$
A	0.06	1.94
PA	0.27	1.91
C	0.25	1.50

Both the table and the figure support the conclusions of Theorem 2.2; namely, that the largest eigenvalue of A and the smallest nonzero eigenvalue of C bound the spectrum of PA. (Note that each eigenvalue of C has multiplicity equal to the number of blocks—9 in this case.) We observe also that the bounds are reasonably sharp.

Each diagonal block C_{jj} of the matrix C as defined by (3.3) can be interpreted as the discretization of a related Neumann problem on the jth subdomain. By Theorem 2.2, the effective condition number of the deflated matrix PA is determined by the smallest nonzero eigenvalue of C—in this case, the smallest nonzero eigenvalue over the set of related Neumann problems on the subdomain grids, i.e.,

$$\lambda_{m+1}(PA) = \min_j \lambda_2(C_{jj}).$$

[3] This is generally the case with matrices arising from discretization of PDEs on simply connected domains. If a block B_{ii} *is* reducible, then it may be possible to decompose B_{ii} into additional subdomains which *are* irreducible.

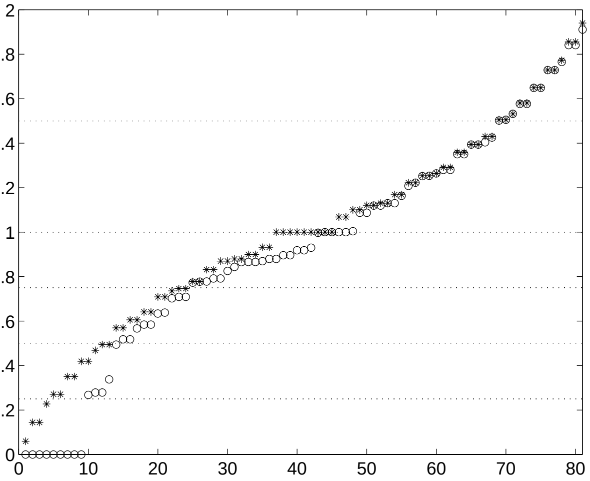


FIG. 3.1. *The eigenvalues of A (*), PA ($\circ$), and C ($\cdots$).*

Theorem 2.2 thus says that *subdomain deflation effectively decouples the original system into a set of independent Neumann problems on the subdomains,* with convergence governed by the "worst-conditioned" Neumann problem. This implies an optimality result, since—if we can somehow refine the grid without affecting the worst-conditioned Neumann problem—the condition number will also remain unchanged. For an isotropic problem on a uniform grid, for example, this can be achieved by simply fixing the subgrid resolutions and performing refinement by adding more subdomains. The numerical experiments of section 6 support this observation.

3.2. Application to finite element stiffness matrices. A result similar to the above discussion on M-matrices holds for finite element stiffness matrices. We briefly describe it here. Suppose we have a domain Ω whose boundary is given by $\partial\Omega = \partial\Omega^D \cup \partial\Omega^N$, with Dirichlet boundary conditions on $\partial\Omega^D$ and Neumann boundary conditions on $\partial\Omega^N$. Let Ω be decomposed into m nonoverlapping subdomains Ω_j, $j = 1, \ldots, m$, and define the finite element decomposition of Ω by

$$\bar{\Omega} = \cup_{i\in\mathcal{I}} \bar{e}_i.$$

Let the index set $\mathcal{I}$ be divided into $m+1$ disjoint subsets $\mathcal{I}_1, \ldots, \mathcal{I}_m$ and $\mathcal{I}_r$, defined by

$$\mathcal{I}_j = \left\{ i \in \mathcal{I} \,|\, e_i \subset \Omega_j \text{ and } \bar{e}_i \cap \partial\Omega^D = \varnothing \right\},$$

and $\mathcal{I}_r = \mathcal{I} \backslash \cup_j \mathcal{I}_j$. Figure 3.2 shows an example of a domain with quadrilateral elements and two subdomains.

The stiffness matrix A is defined as the sum of elemental stiffness matrices A_{e_i}:

$$A = \sum_{i\in\mathcal{I}} A_{e_i},$$

where the elemental matrices are assumed to be positive semidefinite. This is always the case when the integrals in the element matrices are computed analytically. We assume that A is symmetric positive definite. This is normally true if the solution

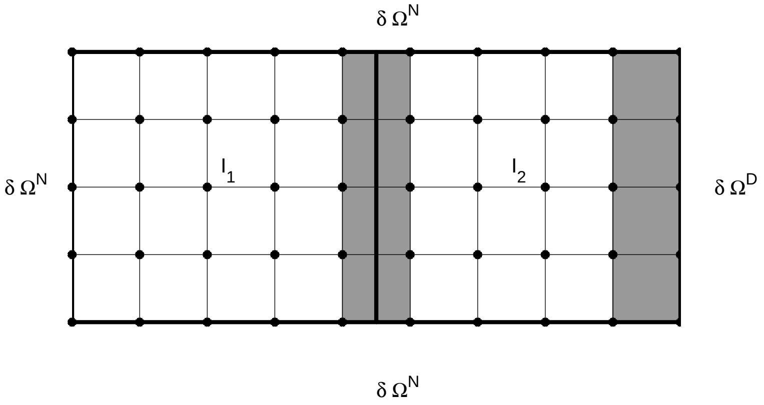


FIG. 3.2. *The domain Ω is decomposed into two subdomains (the shaded region is $\mathcal{I}_r$).*

is prescribed somewhere on the boundary. The matrix C needed for Theorem 2.2 is defined by

$$C = \sum_{i \in \mathcal{I} \backslash \mathcal{I}_r} A_{e_i}.$$

Note that C is block diagonal and the blocks C_{jj} can be interpreted as a finite element discretization of the original system on the subdomain Ω_j with homogeneous Neumann boundary conditions. This implies that $\lambda_1(C_{jj}) = 0$ and that Z is in the null space of C. Clearly C is positive semidefinite, as is

$$R = \sum_{i \in \mathcal{I}_r} A_{e_i}.$$

To ensure that $\lambda_{m+1}(C) \neq 0$, it is necessary that every grid point $x_k \in \bar{\Omega} \backslash \partial\Omega^D$ be contained in a finite element e_i with $i \in \cup_{j=1}^m \mathcal{I}_j$; otherwise the ith row of C contains only zero elements.

4. Guidelines for selecting subdomains.

We can use the results of the previous section to give guidance in choosing a good decomposition of the domain Ω such that the "worst-conditioned related Neumann problem" is as well conditioned as possible. We consider two cases: a Poisson equation on a stretched uniform grid, and a diffusion equation with a discontinuity in the diffusion coefficient.

4.1. Large domain/grid aspect ratios.

Consider the Poisson equation with homogeneous Neumann boundary conditions on a rectangular domain Ω:

$$-\Delta u = f, \quad \partial u / \partial \hat{n} = 0, \; u \in \partial\Omega,$$

where $\hat{n}$ denotes the unit normal vector to the boundary. This equation is discretized using cell-centered, central finite volumes on a uniform $N_x \times N_y$ grid having cell dimensions $h_x \times h_y$:

$$\frac{1}{h_x^2}(-u_{j-1,k} + 2u_{j,k} - u_{j+1,k}) + \frac{1}{h_y^2}(-u_{j,k-1} + 2u_{j,k} - u_{j,k+1}) = f_{j,k}$$

for $j = 0, \ldots, N_x$ and $k = 0, \ldots, N_y$. Assume central discretization of the boundary conditions

$$u_{-1,k} = u_{0,k}, \text{ etc.};$$

then, the eigenvalues of the discretization matrix are given by

$$\lambda_{j,k} = \frac{4}{h_x^2} \sin^2\left(\frac{j\pi}{2(N_x+1)}\right) + \frac{4}{h_y^2} \sin^2\left(\frac{k\pi}{2(N_y+1)}\right). \tag{4.1}$$

The largest eigenvalue is λ_{N_x,N_y} and the smallest nonzero eigenvalue is the minimum of $\lambda_{0,1}$ and $\lambda_{1,0}$. Substituting into (4.1), and assuming $N_x, N_y \gg 1$, we find

$$\begin{aligned}\lambda_{N_x,N_y} &\approx \frac{4}{h_x^2} + \frac{4}{h_y^2},\\ \lambda_{0,1} &\approx \frac{4}{h_y^2}\left(\frac{\pi}{2(N_y+1)}\right)^2 = \frac{\pi^2}{h_y^2(N_y+1)^2},\\ \lambda_{1,0} &\approx \frac{4}{h_x^2}\left(\frac{\pi}{2(N_x+1)}\right)^2 = \frac{\pi^2}{h_x^2(N_x+1)^2}.\end{aligned} \tag{4.2}$$

The decomposition problem can be stated as follows: For a fixed cell aspect ratio $\mathcal{Q}_c \equiv h_x/h_y$ and a fixed total number of cells $\gamma \equiv N_x N_y = \text{const}$, find the grid aspect ratio $\mathcal{Q}_g \equiv N_x/N_y$ minimizing the effective condition number

$$\begin{aligned}\kappa_{\text{eff}} &= \max\left\{\frac{\lambda_{N_x,N_y}}{\lambda_{0,1}}, \frac{\lambda_{N_x,N_y}}{\lambda_{1,0}}\right\}\\ &= 4/\pi^2 \max\left\{(1+\mathcal{Q}_c^{-2})(\gamma/N_x+1)^2, (1+\mathcal{Q}_c^2)(N_x+1)^2\right\}.\end{aligned}$$

Since both arguments of the maximum are monotone functions of positive N_x, one increasing and the other decreasing, the condition number is minimized when these arguments are equal:

$$\begin{gathered}(1+\mathcal{Q}_c^{-2})(\gamma/N_x+1)^2 = (1+\mathcal{Q}_c^2)(N_x+1)^2,\\ \frac{1}{\mathcal{Q}_c^2} = \frac{1+\mathcal{Q}_c^{-2}}{1+\mathcal{Q}_c^2} = \frac{(N_x+1)^2}{(N_y+1)^2} \approx \mathcal{Q}_g^2.\end{gathered}$$

Thus, for constant coefficients and a uniform grid, one should choose a decomposition such that the subdomain grid aspect ratio is the reciprocal of the cell aspect ratio; that is, one should strive for a subdomain aspect ratio $\mathcal{Q}_d \equiv (N_x h_x)/(N_y h_y)$ of 1:

$$\mathcal{Q}_d = \mathcal{Q}_g \mathcal{Q}_c = 1.$$

Example. Again take the Poisson equation on the unit square (3.4), with a grid resolution $N_x = 16$, $N_y = 32$. We compare the condition number of PA for three decompositions into 16 subdomains as shown in Figure 4.1:

	$\lambda_{\min}(C)$	$\lambda_{\min}(PA)$	$\kappa(PA)$
2×8	0.013	0.024	83.0
4×4	0.053	0.062	32.2
8×2	0.014	0.024	81.8

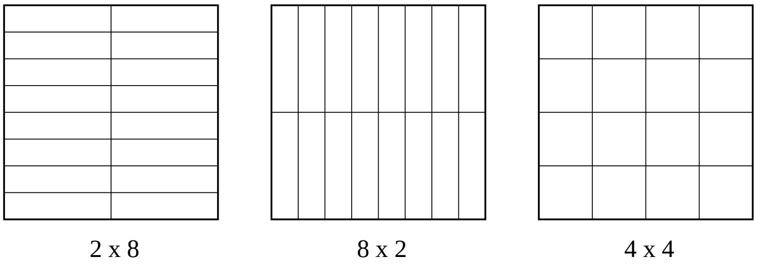


FIG. 4.1. *Three decompositions of the unit square into* 16 *subdomains.*

The 4×4 decomposition yields a subdomain aspect ratio of $\mathcal{Q}_d = 1$, and this is the best-conditioned case, as predicted.

The decomposition problem described above assumes that the grid size and the number of domains is given, and that one would like to choose the decomposition for optimal convergence rate. This would be the case, for example, if a parallel decomposition is desired on a prescribed number of processors. For a serial computation, or if there is an unlimited number of available processors, a better approach would be to ask what number of domains gives the fastest solution. Suppose we decompose into subdomains of unit aspect ratio, as described above. By comparison with (4.2), the smallest positive eigenvalue of C scales as $1/N_x^2$, with N_x the number of grid cells in the x direction for the worst-conditioned Neumann problem. Thus if we split each subdomain horizontally and vertically into four equivalent smaller subdomains, the condition number of C is improved by a factor of 4, roughly speaking. On the other hand, the dimension of the coarse grid matrix A_c will be increased by a factor of 4, causing the direct (or recursive) solution of this system to be relatively more expensive. In the extreme case of one unknown per subdomain, $A_c = A$, so that solving A_c is as expensive as solving A. Clearly, there must be an optimal value for the number of subdomains; however, this will depend on the convergence of the conjugate gradients process, and therefore also on the distribution of eigenvalues.

4.2. Discontinuous coefficients. When a problem has a large jump in coefficients at some location, poor scaling may result in slow convergence. It may be possible to improve the convergence by applying subdomain deflation, choosing the subdomain interface at the discontinuity. Since the related Neumann problems are decoupled, a diagonal scaling preconditioner is sufficient to make the condition number independent of the jump in coefficients. This is best illustrated with an example.

Consider a one-dimensional diffusion problem with Neumann and Dirichlet boundary conditions

$$-\frac{d}{dx}\alpha(x)\frac{du}{dx} = f(x), \quad x \in (0,1), \quad \frac{du}{dx}(0) = 0, \ u(1) = 1,$$

and a jump discontinuity in the coefficient

$$\alpha(x) = \begin{cases} 1, & x \leq 0.5, \\ \epsilon, & x > 0.5 \end{cases}$$

for some $\epsilon > 0$. Choose an even number n and define $h = 1/n$. The grid points are given by $x_i = ih, i = 0, \ldots, n$ and u_i is the numerical approximation for $u(x_i)$. For all $i \in \{0, 1, \ldots, n-1\} \setminus \{n/2\}$ we use the standard central difference scheme.

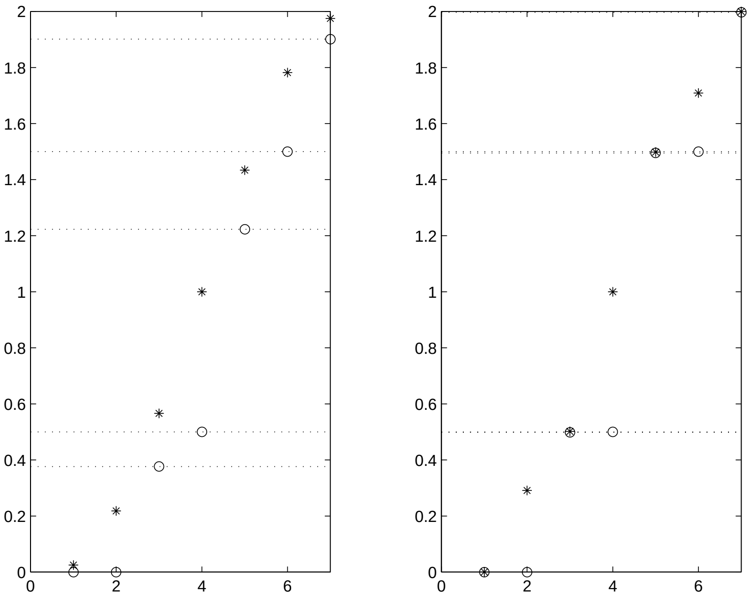


FIG. 4.2. *Eigenvalues of $D^{-1}A$ ($*$) and $D^{-1}PA$ ($\circ$) for $\epsilon = 1$ (left) and $\epsilon = 0.01$ (right). The spectrum of $D^{-1}C$ is indicated by the dotted lines.*

The unknown u_n is eliminated from the system of equations by using the Dirichlet boundary condition. For $i = 0$ the value u_{-1} is eliminated by a central discretization of the Neumann boundary condition. The resulting equation is multiplied by 1/2 to make the coefficient matrix symmetric. Finally for $i = n/2$ the discrete equation is

$$\frac{\frac{u_{n/2}-u_{n/2-1}}{h} - \epsilon\frac{u_{n/2+1}-u_{n/2}}{h}}{h} = f(x_{n/2}).$$

The domain $\Omega = [0, 1]$ is subdivided into two subdomains $\Omega_1 = [0, 0.5]$ and $\Omega_2 = (0.5, 1]$. Note that grid point $x_{n/2} = 0.5$ belongs to Ω_1. The subdomain deflation space Z is defined by (3.1).

To construct C from A we decouple the matrix A according to the subdomains, so

$$c_{n/2+1,n/2} = c_{n/2,n/2+1} = 0.$$

The other off-diagonal elements of A and C are identical. Finally the diagonal elements of C are made equal to minus the sum of the off-diagonal elements, so

$$\sum_{j=1}^{n} c_{ij} = 0.$$

Let D be the diagonal of A. The eigenvalues of $D^{-1}A$ and $D^{-1}PA$ (equivalent to the eigenvalues of the symmetrically preconditioned case $D^{-1/2}AD^{-1/2}$, etc.) with $n = 8$ are shown in Figure 4.2 for $\epsilon = 1$ and $\epsilon = 0.01$ with the eigenvalues of $D^{-1}C$ appearing as dotted lines. Note that the smallest positive eigenvalue of $D^{-1}C$ bounds from below the smallest positive eigenvalue of $D^{-1}PA$, as predicted by Theorem 2.3.

In the following table we give the effective condition numbers relevant for convergence of the preconditioned conjugate gradient method.

ϵ	$\lambda_1(D^{-1}A)$	$\kappa(D^{-1}A)$	$\lambda_3(D^{-1}PA)$	$\kappa_{\text{eff}}(D^{-1}PA)$
1	$2.5 \cdot 10^{-2}$	$7.9 \cdot 10^{1}$	$3.8 \cdot 10^{-1}$	5.0
10^{-2}	$4.1 \cdot 10^{-4}$	$4.8 \cdot 10^{3}$	$5.0 \cdot 10^{-1}$	4.0
10^{-4}	$4.2 \cdot 10^{-6}$	$4.8 \cdot 10^{5}$	$5.0 \cdot 10^{-1}$	4.0

Due to diagonal preconditioning, the smallest nonzero eigenvalue of $D^{-1}C$ is independent of ϵ. As predicted by Theorem 2.3, the same holds for $D^{-1}PA$. The smallest eigenvalue of $D^{-1}A$, however, decreases proportionally to ϵ, leading to a large condition number and slow convergence of the conjugate gradient method applied to $D^{-1}Au = D^{-1}f$.

5. Additional considerations.

In this section we discuss extension of deflation methods to the nonsymmetric case and describe an efficient parallel implementation of the subdomain deflation method.

5.1. The nonsymmetric case.

A generalization of the projection P for a nonsymmetric matrix $A \in \mathbb{R}^{n\times n}$ is used in [29]. In this case there is somewhat more freedom in selecting the projection subspaces. Let P and Q be given by

$$P = I - AZ(Y^TAZ)^{-1}Y^T, \quad Q = I - Z(Y^TAZ)^{-1}Y^TA,$$

where Z and Y are suitable subspaces of dimension $n \times m$. The operator A_c on the projection subspace is given by $A_c = Y^TAZ$.[4] We have the following properties for P and Q:

- $P^2 = P$, $Q^2 = Q$.
- $PAZ = Y^TP = 0$, $Y^TAQ = QZ = 0$.
- $PA = AQ$.

To solve the system $Au = f$ using deflation, note that u can be written as

$$u = (I - Q)u + Qu$$

and that $(I-Q)u = Z(Y^TAZ)^{-1}Y^TAu = Z(Y^TAZ)^{-1}Y^Tf$ can be computed immediately (cf. (1.3)). Furthermore Qu can be obtained by solving the deflated system

$$PA\tilde{u} = Pf \tag{5.1}$$

for $\tilde{u}$ (cf. (1.4)) and premultiplying the result with Q.

Also in the nonsymmetric case, deflation can be combined with preconditioning. Suppose K is a suitable preconditioner of A, then (5.1) can be replaced by the following: solve $\tilde{u}$ from

$$K^{-1}PA\tilde{u} = K^{-1}Pf \tag{5.2}$$

and form $Q\tilde{u}$, or solve $\tilde{v}$ from

$$PAK^{-1}\tilde{v} = Pf \tag{5.3}$$

and form $QK^{-1}\tilde{v}$. Both systems can be solved by one's favorite Krylov subspace solver, such as GMRES [22], GCR [7, 25], Bi-CGSTAB [24], etc.

The question remains how to choose Y. We consider two possibilities:

[4] In multigrid terminology, Z is the projection or interpolation operator, and Y^T is the restriction operator.

1. Suppose Z consists of eigenvectors of A. Choose Y as the corresponding eigenvectors of A^T.
2. Choose $Y = Z$.

For both choices we can prove some results about the spectrum of PA.

ASSUMPTION 5.1. *We assume that A has real eigenvalues and is nondefective.*

Whenever A satisfies Assumption 5.1 there exists a matrix $X \in \mathbb{R}^{n\times n}$ such that $X^{-1}AX = \mathrm{diag}(\lambda_1, \dots, \lambda_n)$. For the first choice, which is related to Hotelling deflation (see [30, p. 585]), we have the following result.

LEMMA 5.1. *If A satisfies Assumption 5.1, $Z = [x_1 \cdots x_m]$, and Y is the matrix composed of the first m columns of X^{-T}, then*

$$X^{-1}PAX = \mathrm{diag}(0, \dots, 0, \lambda_{m+1}, \dots, \lambda_n).$$

Proof. From the definition of P we obtain $PAZ = 0$, so $PAx_i = 0,\ i = 1, \dots, m$. For the other vectors $x_i,\ i = m + 1, \dots, n$, we note that

$$PAx_i = Ax_i - AZ(Y^TAZ)^{-1}Y^TAx_i = \lambda_ix_i - AZ(Y^TAZ)^{-1}\lambda_iY^Tx_i = \lambda_ix_i. \qquad \square$$

The second choice $Y = Z$ has the following properties.

LEMMA 5.2. *For $Y = Z$ one has the following:*

(i) *If A is positive definite and Z has full rank, $A_c = Z^TAZ$ is nonsingular.*

(ii) *If A satisfies Assumption 5.1 and $Z = [x_1 \cdots x_m]$, the eigenvalues of PA are $\{0, \lambda_{m+1}, \dots, \lambda_n\}$, where the zero-eigenvalue has multiplicity m.*

Proof. (i) For $Y = Z$ the matrix $A_c = Z^TAZ$ is nonsingular since $s^TA_cs > 0$ for all $s \in \mathbb{R}^m$ and $s \neq 0$.

(ii) Again $PAx_i = 0$ for $i = 1, \dots, m$. For the other eigenvalues we define the vectors

$$v_i = x_i - AZA_c^{-1}Z^Tx_i = x_i - ZD_mA_c^{-1}Z^Tx_i, \quad i = m + 1, \dots, n,$$

where $D_m = \mathrm{diag}(\lambda_1, \dots, \lambda_m)$. These vectors are nonzero, because $x_1, \dots, x_n$ form an independent set. Multiplication of v_i by PA yields

$$PAv_i = PA(x_i - ZD_mA_c^{-1}Z^Tx_i) = PAx_i = Ax_i - AZA_c^{-1}Z^TAx_i = \lambda_iv_i,$$

which proves the lemma. $\square$

From these lemmas we conclude that both choices of Y lead to the same spectrum of PA. The second choice has the following advantages: when A is positive definite we have proven that A_c is nonsingular; it is not necessary to determine (or approximate) the eigenvectors of A^T; and finally only one set of vectors $z_1, \dots, z_m$ has to be stored in memory. This motivates us to use the choice $Y = Z$. In our applications Z is not an approximation of an invariant subspace of A but is defined as in (3.1).

Theorems 2.2 and 2.3 do not apply to the nonsymmetric case. However, our experience has shown that the convergence of (5.1) is similar to that of (1.4) as long as the asymmetric part of A is not too dominant.

5.2. Parallel implementation. In this section we describe an efficient parallel implementation of the subdomain deflation method with Z defined by (3.1). We distribute the unknowns according to subdomain across available processors. For the discussion we will assume one subdomain per processor. The coupling with neighboring domains is realized through the use of virtual cells added to the local grids. In this way, a block-row of $Au = f$ corresponding to the subdomain ordering

$$A = \begin{bmatrix} A_{11} & \cdots & A_{1m} \\ \vdots & \vdots & \vdots \\ A_{m1} & \cdots & A_{mm} \end{bmatrix} \tag{5.4}$$

can be represented locally on one processor: the diagonal block A_{ii} represents coupling between local unknowns of subdomain i, and the off-diagonal blocks of block-row i represent coupling between local unknowns and the virtual cells.

Computation of element $A_{c_{ij}}$ of $A_c = Z^TAZ$ can be done locally on processor i by summing the coefficients corresponding to block A_{ij} of (5.4): $A_{c_{ij}} = \mathbf{1}^T A_{ij}\mathbf{1}$.

Use of the deflation P within a Krylov subspace method involves premultiplying a vector v by PA:

$$PAv = (I - AZ(Z^TAZ)^{-1}Z^T)Av.$$

Assuming A_c^{-1} has been stored in factored form, this operation requires two multiplications with A. However, the special form of Z given by (3.1) allows some simplification. Since Z is piecewise constant, we can efficiently compute and store the vectors

$$w_j = Az_j = \begin{bmatrix} A_{1j} \\ \vdots \\ A_{mj} \end{bmatrix} \mathbf{1} \tag{5.5}$$

corresponding to row sums of the jth block-column of A. Note that for the ith block system the local block of w_j is nonzero only if there is coupling between subdomains i and j, and only the nonzero blocks of w_j need be stored. Thus, for a five-point stencil the number of nonzero vectors w_j which have to be stored per block is five. Furthermore, for many applications, the row sums are zero, and w_j is only nonzero on subdomain boundaries.

With the w_j stored, local computation of $AZ\tilde{e}$ for a given (m-dimensional) vector $\tilde{e}$ consists of scaling the nonzero w_j by the corresponding $\tilde{e}_j$ and summing them up: $AZ\tilde{e} = \sum_j \tilde{e}_j w_j$. The number of vector updates is five for a five-point stencil.

In parallel, we first compute and store the (nonzero parts of the) w_j and $(Z^TAZ)^{-1}$ (factored) on each processor. In particular, on processor i we store the local part $w_j = A_{ij}\mathbf{1}$ for all nonzero A_{ij}. Then to compute PAv we first perform the matrix-vector multiplication $\tilde{q} = Av$, requiring nearest neighbor communications. Next we compute the local contribution to the restriction $q = Z^T\tilde{q}$ (local summation over all grid points) and distribute the result to all processes. With this done, we solve for $\tilde{e}$ from $A_c\tilde{e} = q$ and finally compute $AZ\tilde{e} = \sum_j \tilde{e}_j w_j$ locally.

The total parallel communication involved in the matrix-vector multiplication and deflation are a nearest neighbor communication of the length of the interfaces and a global gather-broadcast of dimension m.

The computational and communication costs plus storage requirements of subdomain deflation are summarized in Table 5.1, assuming a five-point discretization stencil on an $N_x \times N_y$ grid with $M_x \times M_y$ decomposition into blocks of revolution $n_x \times n_y$ ($N_x = n_xM_x$, $N_y = n_yM_y$). The abbreviation *GaBr* (m) refers to a gather-broadcast operation in which a set of m distributed floating point numbers is gathered from the participating processors and then the whole set is returned to each processor. The construction costs are incurred only once, whereas the iteration costs are in each conjugate gradient iteration. Also included in the table are the costs of an (in the parallel case, blockwise) incomplete factorization preconditioner with zero fill-in, ILU(0).

TABLE 5.1
Work, storage, and communication costs for deflation-based preconditioning.

	Sequential		Parallel		
	Work	Storage	Work	Storage	Comms.
Construction:					
ILU(0)	$6N_xN_y$	N_xN_y	$6n_xn_y$	n_xn_y	0
A_c	$5N_xN_y$	$5M_xM_y$	$5n_xn_y$	$5M_xM_y$	GaBr $(5M_xM_y)$
Band-factor A_c	$2M_x^3My$	$2M_x^2M_y$	$2M_x^3M_y$	$2M_x^2M_y$	0
AZ	$9N_xN_y$	$5N_xN_y$	$9n_xn_y$	$9n_xn_y$	0
Iteration:					
Backsolve IC(0):	$10N_xN_y$		$10n_xn_y$		0
Restriction: $q = Z^TAv$	N_xN_y		n_xn_y		0
Backsolve: $A_c\tilde{e} = q$	$4M_x^2M_y$		$4M_x^2M_y$		GaBr (M_xM_y)
Prolongation: $AZ\tilde{e}$	$5N_xN_y$		$5n_xn_y$		0
Vector update: $Av - AZ\tilde{e}$	N_xN_y		n_xn_y		0

Besides the items tabulated above, there are computation and communication costs associated with the matrix-vector multiplication and inner products as well as computational costs of vector updates, associated with the CG method. Based on this table, we expect the added iteration expense of deflation to be less expensive than an ILU(0) factorization, and that the method will parallelize very efficiently on a distributed memory computer.

6. Numerical experiments. All experiments in this section are conducted with PDEs discretized using cell-centered, central finite volumes on Cartesian grids in rectangular regions. The theory discussed until now makes no such assumptions, however, and should hold in a more general, unstructured setting.

In conducting numerical experiments, we are interested in the following issues: (i) verification of the theoretical results of this article, (ii) the properties of subdomain deflation for nonsymmetric systems, and (iii) the parallel performance of the method. To this end we consider three test cases:

I. Poisson equation: $-\Delta u(x,y) = f$.

II. Diffusion equation: $-\nabla \cdot \nu(x,y)\nabla u(x,y) = f$.

III. Steady-state convection-diffusion equation: $\nabla\cdot(\mathbf{a}(x,y)u(x,y)) - \Delta u(x,y) = f$.

In most examples we take $f \equiv 1$, having checked that similar results are observed for a random right-hand side function. We use a global grid resolution $N_x \times N_y$, with decomposition into $M_x \times M_y$ subdomains, each of resolution $n_x \times n_y$ (thus, $N_x = n_xM_x$ and $N_y = n_yM_y$).

We solve the resulting discrete (symmetric) system using the CG method and subdomain deflation. The initial iterate is chosen to be $u^{(0)} = 0$, and convergence is declared when, in the Jth iteration, $\|r_J\| \leq tol \cdot \|r_0\|$ for $tol = 10^{-6}$.

When classical preconditioning is included, we solve $K^{-1}PAu = K^{-1}Pf$, where the preconditioner K used on the blocks is the relaxed incomplete Cholesky (RIC) factorization of [1], with relaxation parameter $\omega = 0.975$. We choose this preconditioner because it is simple to implement (for a five-point stencil, modifications occur only on the diagonal) and is reasonably effective. Certainly, more advanced preconditioners could be employed on the blocks of C.

6.1. Convergence results. In this section we give convergence results with problems I, II, and III to illustrate the insensitivity of the convergence to the number of subdomains, the optimal decomposition on stretched grids, the effectiveness of the method for problems with discontinuous coefficients, and the convergence behavior for nonsymmetric problems.

6.1.1. Near grid independence. First we illustrate the sense in which subdomain deflation can lead to nearly grid-independent convergence. The symmetric discretization matrix of problem I on $(0,1)\times(0,1)$ with homogeneous Dirichlet boundary conditions is used without preconditioning. Keeping the resolution of each subdomain fixed, the number of subdomains is increased. In so doing, the blocks of C remain roughly the same as the grid is refined, and the bound in (2.1) becomes insensitive to the number of blocks m for large enough m.

Assume $M_x = M_y$ and $n_x = n_y$. Figure 6.1 shows the scaled number of CG iterations J/n_x (note that n_x is constant along each line in the figure) for problem I as the grid is refined keeping the subdomain resolution n_x fixed at values of 10, 50, and 200. The lines are almost indistinguishable from one another. It is apparent from the figure that—using only subdomain deflation—the number of iterations required for convergence is bounded independent of the number of subdomains. The same qualitative behavior is observed with preconditioning.

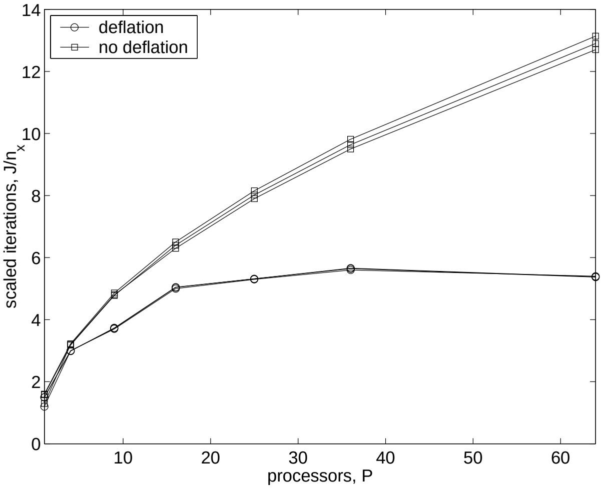


FIG. 6.1. *Number of iterations J divided by the subdomain resolution $n_x \equiv n_y \in \{10, 50, 200\}$ with and without deflation.*

6.1.2. Stretched grid. We consider problem I on $(0,3)\times(0,1)$ with homogeneous Dirichlet boundary conditions, and $N_x = 36$ and $N_y = 72$. The cell aspect ratio is $\mathcal{Q}_c = h_x/h_y = (3/36)/(1/72) = 6$. Based on the discussion of section 4.1, the best condition number is expected for a subdomain aspect ratio $\mathcal{Q}_d = 1$, associated with a subdomain grid aspect ratio of $\mathcal{Q}_g = \mathcal{Q}_d/\mathcal{Q}_c = 1/6$. Table 6.1 gives the number of iterations required for convergence for 5 different decompositions into 12 equally sized subdomains. The solution tolerance of the nonpreconditioned CG algorithm was set to $tol = 10^{-2}$, prior to the onset of superlinear convergence, to obtain these results. The 6×2 decomposition with $\mathcal{Q}_d = 1$ gives the minimum number of iterations, in keeping with the discussion. We note that if iteration is continued to high tolerance, the superlinear convergence effect may give quite different results than shown here. This domain decomposition selection strategy is most useful when the condition number governs the convergence rate.

Table 6.1
Iterations required for problem I *for different decompositions.*

$M_x \times M_y$	$n_x \times n_y$	Q_d	J
2×6	18×12	9	73
3×4	12×18	4	63
4×3	9×24	9/4	56
6×2	6×36	1	48
12×1	3×72	1/4	50

6.1.3. Discontinuous coefficients. To further illustrate the discussion of section 4.2 we give results for problem II on $(0,1) \times (0,1)$ with boundary conditions $u_x(0,y) \equiv u_y(x,0) \equiv u_y(x,1) \equiv 0$, $u(1,y) \equiv 0$. We define the diffusion coefficient to have value $\nu(x,y) = 1$ on the lower left subdomain, including its interfaces, and $\nu(x,y) = \epsilon$ elsewhere. Table 6.2 lists the iterations for the CG method with diagonal preconditioning for $M_x = M_y = 3$ and $n_x = n_y = 30$, as ϵ is decreased.

One observes that this is a very effective strategy for eliminating the effect of the jump in coefficients.

Table 6.2
Iterations for problem II *with discontinuous coefficients.*

ϵ	No deflation	Deflation
1	295	151
10^{-2}	460	183
10^{-4}	521	189
10^{-6}	628	189

6.1.4. A nonsymmetric example. We also illustrate the convergence of the deflation method for a convection dominated problem III on $(0,1) \times (0,1)$ with recirculating wind field $a_1(x,y) = -80xy(1-x)$, $a_2(x,y) = 80xy(1-y)$ and boundary conditions $u(x,0) \equiv u(y,0) \equiv u(x,1) \equiv 0$, $u_x(1,y) = 0$. The grid parameters are $N_x = N_y$, $M_x = M_y$, $n_x = n_y$ with grid spacing given by

$$x_i = (i/N_x)^2(3 - 2(i/N_x)).$$

The resulting system is solved using GCR truncated to a subspace of 20 vectors by dropping the vector most nearly orthogonal to the current search direction [26]. Classical preconditioning in the form of RILU(0.975) is incorporated. The restriction matrix for deflation is chosen to be $Y = Z$.

Table 6.3 compares the required number of GCR iterations as the number of subdomains is increased keeping the subdomain resolution fixed at $n_x = 50$. Although the number of iterations is not bounded in the deflated case, it grows much slower than the nondeflated case.

6.2. Parallel performance. For the results in this section, problem I will be solved on $(0,1) \times (0,1)$ with homogeneous Dirichlet boundary conditions everywhere.

The resulting equations are solved with CG preconditioned with RIC(0.975). Our implementation does not take advantage of the fact that some of the row sums may

Table 6.3
Scalability for a nonsymmetric problem, subdomain grid 50 × 50.

M_x	No deflation	Deflation
1	42	42
2	122	122
3	224	191
4	314	235
5	369	250
6	518	283
7	1007	377

Table 6.4
Speedup for problem I *on a* 120 × 120 *grid.*

p	J	t_{const}	t_{iter}	s	*eff*
1	38	$8.7 \cdot 10^{-3}$	1.3	–	–
4	58	$1.2 \cdot 10^{-2}$	0.57	2.3	0.58
9	68	$5.0 \cdot 10^{-3}$	0.33	4.0	0.44
16	64	$5.3 \cdot 10^{-3}$	0.18	7.2	0.45
25	57	$4.3 \cdot 10^{-3}$	0.15	9.0	0.36
36	50	$7.6 \cdot 10^{-3}$	0.11	11.7	0.33
64	41	$1.1 \cdot 10^{-2}$	0.11	12.3	0.19

Table 6.5
Speedup for problem I *on a* 480 × 480 *grid.*

p	J	t_{const}	t_{iter}	s	*eff*
1	120	$1.4 \cdot 10^{-1}$	67.3	–	–
4	137	$1.3 \cdot 10^{-1}$	21.8	3.1	0.77
9	138	$6.3 \cdot 10^{-2}$	9.65	7.0	0.78
16	139	$3.6 \cdot 10^{-2}$	5.60	12.0	0.75
25	121	$2.5 \cdot 10^{-2}$	3.21	21.0	0.84
36	118	$2.2 \cdot 10^{-2}$	2.27	29.7	0.82
64	100	$1.3 \cdot 10^{-2}$	1.19	56.6	0.88

be zero in (5.5). Each processor is responsible for exactly one subdomain. Parallel communications were performed with MPI, using simple point-to-point and collective communications. No exploitation of the network topology was used. Parallel results were obtained from a Cray T3E. Wall-clock times in seconds were measured using the MPI timing routine.

6.2.1. Speedup for fixed problem size. To measure the speedup, we choose $p = M_x^2$ processors for $M_x \in \{1, 2, 3, 4, 5, 6, 8\}$. The results are given in Tables 6.4 and 6.5 for $N_x = 120$ and $N_x = 480$, respectively. The total number of iterations is denoted by J; the time to construct the incomplete factorization and deflation operator is denoted by t_{const}; and the time spent in iterations is denoted by t_{iter}. The speedup is determined from $s = (t_{\text{iter}}|_{p=1}) / (t_{\text{iter}}|_{p=M_x^2})$ and parallel efficiency by $\mathit{eff} = s/p$.

In Table 6.4 the parallel efficiency decreases from 58% on 4 processors to only 19% on 64 processors, whereas in Table 6.5 efficiency increases slightly from 77% to 88%. We expect that the poorer performance in the first table is due to both a relatively large cost of solving the coarse operator A_c and a large communication-to-computation ratio for small subdomains. The following factors contribute to the parallel performance:

- As more subdomains are added, the relative size of the deflation system A_c

increases, making it more expensive to solve, but at the same time, its solution becomes a better approximation of the global solution.
- As the size of the subdomain grids decreases, the *RILU* preconditioner becomes a better approximation of the exact solution of the subdomain problems.
- Global communications become more expensive for many subdomains.
- Additionally there may be architecture-dependent effects in play.

6.2.2. Scaled performance for fixed subdomain size. Table 6.6 gives the computation times in seconds obtained with and without deflation, keeping the subdomain size fixed at $n_x \in \{5, 10, 20, 50, 100, 200\}$ as the number of processors is increased. It is clear that the effect of deflation is to make the parallel computation time less sensitive to the number of processors.

We have already seen that the number of iterations levels off as a function of the number of subdomains. The results of this table show that also the parallel iteration time becomes relatively insensitive to an increase in the number of blocks. Some overhead is incurred in the form of global communications and in solving the deflation subsystem. As a result, the computation times are not bounded independent of the number of subdomains.

Comparing the iteration counts for this problem, we note that the ratio of iterations with and without deflation is very similar to that of Figure 6.1 without preconditioning. Furthermore, the cost per iteration scales with n_x^2 for $n_x \geq 20$ (for smaller n_x, the cost of deflation offsets the advantage gained). The effect of preconditioning is to reduce the necessary number of iterations in both the deflated and undeflated cases such that the ratio of iterations remains fixed. We therefore expect that the ratio of computation times with and without deflation should reflect the ratios of Figure 6.1 as well.

TABLE 6.6
Scaled performance for problem I *with fixed subdomain size* n_x.

n_x		$p = 1$	$p = 4$	$p = 9$	$p = 16$	$p = 25$	$p = 36$	$p = 64$
5	no defl.	$4 \cdot 10^{-4}$	$4 \cdot 10^{-3}$	$1 \cdot 10^{-2}$	$2 \cdot 10^{-2}$	$3 \cdot 10^{-2}$	$4 \cdot 10^{-2}$	$4 \cdot 10^{-2}$
	defl.	—	$5 \cdot 10^{-3}$	$1 \cdot 10^{-2}$	$1 \cdot 10^{-2}$	$2 \cdot 10^{-2}$	$3 \cdot 10^{-2}$	$4 \cdot 10^{-2}$
10	no defl.	$1 \cdot 10^{-3}$	$9 \cdot 10^{-3}$	$3 \cdot 10^{-2}$	$3 \cdot 10^{-2}$	$5 \cdot 10^{-2}$	$6 \cdot 10^{-2}$	$7 \cdot 10^{-2}$
	defl.	—	$1 \cdot 10^{-2}$	$3 \cdot 10^{-2}$	$4 \cdot 10^{-2}$	$5 \cdot 10^{-2}$	$6 \cdot 10^{-2}$	$6 \cdot 10^{-2}$
20	no defl.	$6 \cdot 10^{-3}$	$3 \cdot 10^{-2}$	$6 \cdot 10^{-2}$	$8 \cdot 10^{-2}$	0.12	0.15	0.18
	defl.	—	$3 \cdot 10^{-2}$	$7 \cdot 10^{-2}$	$8 \cdot 10^{-2}$	0.10	0.11	0.13
50	no defl.	0.11	0.34	0.51	0.69	0.94	1.10	1.37
	defl.	—	0.35	0.57	0.64	0.71	0.75	0.77
100	no defl.	0.78	2.11	2.98	4.10	5.29	6.23	8.00
	defl.	—	2.10	3.27	3.46	3.58	3.89	3.97
200	no defl.	4.96	13.3	18.6	25.3	32.8	38.6	49.7
	defl.	—	12.9	17.6	20.4	20.8	22.5	23.3

7. Conclusions. In this paper we have given new effective condition number bounds for deflated systems, both with and without conventional preconditioning. Specifically, we show that choosing the deflation subspace to be piecewise constant on subdomains effectively decouples the problem into a set of related Neumann problems, with the convergence governed by the "worst-conditioned" Neumann problem. This knowledge can help to choose an effective decomposition of the domain and is especially useful for problems with large discontinuities in the coefficients. Numerical experiments illustrate that the convergence rate is nearly independent of the number of subdomains for some problems, and that the method can be very efficiently implemented on distributed memory parallel computers.

We see the deflation approach presented here as offering improved convergence rate at a small additional cost for parallel computations using blockwise application of conventional preconditioners. The reader is referred to [9] for a comparison of blockwise incomplete factorization in the framework of nonoverlapping domain decomposition. In that reference is also a comparison of blockwise incomplete factorization with single-block incomplete factorization. In turn, to put these results in perspective, Botta et al. [4] compare a number of modern strategies including ICCG and multigrid methods.

Acknowledgments. We thank HPαC for the use of the Cray T3E, and Pieter Wesseling, Guus Segal, Jos van Kan, and a referee for helpful discussions and suggestions.

REFERENCES

[1] O. Axelsson and G. Lindskog, *On the eigenvalue distribution of a class of preconditioning methods*, Numer. Math., 48 (1986), pp. 479–498.

[2] J. Baglama, D. Calvetti, G. H. Golub, and L. Reichel, *Adaptively preconditioned GMRES algorithms*, SIAM J. Sci. Comput., 20 (1999), pp. 243–269.

[3] A. Berman and R. J. Plemmons, *Nonnegative matrices in the mathematical sciences*, Classics Appl. Math. 9, SIAM, Philadelphia, 1994.

[4] E. F. F. Botta, K. Dekker, Y. Notay, A. van der Ploeg, C. Vuik, F. W. Wubs, and P. M. de Zeeuw, *How fast the Laplace equation was solved in* 1995, Appl. Numer. Math., 24 (1997), pp. 439–455.

[5] K. Burrage, J. Erhel, B. Pohl, and A. Williams, *A deflation technique for linear systems of equations*, SIAM J. Sci. Comput., 19 (1998), pp. 1245–1260.

[6] A. Chapman and Y. Saad, *Deflated and augmented Krylov subspace techniques*, Numer. Linear Algebra Appl., 4 (1997), pp. 43–66.

[7] S. C. Eisenstat, H. C. Elman, and M. H. Schultz, *Variational iterative methods for nonsymmetric systems of linear equations*, SIAM J. Numer. Anal., 20 (1983), pp. 345–357.

[8] J. Erhel, K. Burrage, and B. Pohl, *Restarted GMRES preconditioned by deflation*, J. Comput. Appl. Math., 69 (1996), pp. 303–318.

[9] J. Frank and C. Vuik, *Parallel implementation of a multiblock method with approximate subdomain solution*, Appl. Numer. Math., 30 (1999), pp. 403–423.

[10] G. H. Golub and C. F. van Loan, *Matrix Computations*, 3rd ed., Johns Hopkins University Press, Baltimore, MD, 1996.

[11] I. Gustafsson, *A class of first order factorization methods*, BIT, 18 (1978), pp. 142–156.

[12] W. Hackbusch, *Iterative Solution of Large Sparse Systems of Equations*, Springer-Verlag, New York, 1993.

[13] C. B. Jenssen and P. A. Weinerfelt, *Coarse grid correction scheme for implicit multiblock Euler calculations*, AIAA J., 33 (1995), pp. 1816–1821.

[14] E. F. Kaasschieter, *Preconditioned conjugate gradients for solving singular systems*, J. Comput. Appl. Math., 24 (1988), pp. 265–275.

[15] S. A. Kharchenko and A. Y. Yeremin, *Eigenvalue translation based preconditioners for the GMRES(k) method*, Numer. Linear Algebra Appl., 2 (1995), pp. 51–77.

[16] L. Mansfield, *On the conjugate gradient solution of the Schur complement system obtained from domain decomposition*, SIAM J. Numer. Anal., 27 (1990), pp. 1612–1620.

[17] L. Mansfield, *Damped Jacobi preconditioning and coarse grid deflation for conjugate gradient iteration on parallel computers*, SIAM J. Sci. Statist. Comput., 12 (1991), pp. 1314–1323.

[18] J. A. Meijerink and H. A. van der Vorst, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix*, Math. Comput., 31 (1977), pp. 148–162.

[19] R. B. Morgan, *A restarted GMRES method augmented with eigenvectors*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 1154–1171.

[20] R. A. Nicolaides, *Deflation of conjugate gradients with applications to boundary value problems*, SIAM J. Numer. Anal., 24 (1987), pp. 355–365.

[21] Y. Saad, *Analysis of augmented Krylov subspace methods*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 435–449.

[22] Y. Saad and M. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[23] A. van der Sluis and H. A. van der Vorst, *The rate of convergence of conjugate gradients*, Numer. Math., 48 (1986), pp. 543–560.

[24] H. A. van der Vorst, *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for solution of nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 631–644.

[25] H. A. van der Vorst and C. Vuik, *GMRESR: A family of nested GMRES methods*, Numer. Linear Algebra Appl., 1 (1994), pp. 369–386.

[26] C. Vuik, *Further experiences with GMRESR*, Supercomputer, 55 (1993), pp. 13–27.

[27] C. Vuik and J. Frank, *A parallel block preconditioner accelerated by coarse grid correction*, in High-Performance Computing and Networking, Proceedings of the 8th International Conference, HPCN Europe 2000, M. Bubak, H. Afsarmanesh, R. Williams, and B. Hertzberger, eds., Lecture Notes in Comput. Sci. 1823, Springer-Verlag, Berlin, 2000, pp. 99–108.

[28] C. Vuik, J. Frank, and A. Segal, *A parallel block-preconditioned GCR method for incompressible flow problems*, Future Generation Computer Systems, to appear.

[29] C. Vuik, A. Segal, and J. A. Meijerink, *An efficient preconditioned CG method for the solution of a class of layered problems with extreme contrasts in the coefficients*, J. Comput. Phys., 152 (1999), pp. 1–19.

[30] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, 1992.

SIAM J. Sci. Comput.
Vol. 23, No. 2, pp. 463–479

MULTIRESOLUTION APPROXIMATE INVERSE PRECONDITIONERS[*]

ROBERT BRIDSON[†] AND WEI-PAI TANG[‡]

Abstract. We introduce a new preconditioner for elliptic PDEs on unstructured meshes. Using a wavelet-inspired basis we compress the inverse of the matrix, allowing an effective sparse approximate inverse by solving the sparsity vs. accuracy conflict. The key issue in this compression is to use *second generation* wavelets which can be adapted to the unstructured mesh, the true boundary conditions, and even the PDE coefficients. We also show how this gives a new perspective on multiresolution algorithms such as multigrid, interpreting the new preconditioner as a variation on node-nested multigrid. In particular, we hope the new preconditioner will combine the best of both worlds: fast convergence when multilevel methods can succeed but with robust performance for more difficult problems.

The rest of the paper discusses the core issues for the preconditioner: ordering and construction of a factored approximate inverse in the multiresolution basis, robust interpolation on unstructured meshes, automatic mesh coarsening, and purely algebraic alternatives. Some exploratory numerical experiments suggest the superiority of the new basis over the standard basis for several tough problems, including discontinuous anisotropic coefficients, strong convection, and indefinite reaction problems on unstructured meshes, with scalability like hierarchical basis methods achieved.

Key words. preconditioner, multilevel, multigrid, hierarchical basis, unstructured mesh, elliptic PDEs

AMS subject classifications. 65F10, 65F50

PII. S1064827500373784

1. Motivation.

Approximate inverses are becoming increasingly popular preconditioners for the iterative solution of large sparse linear systems. The main reason is that they can be efficiently applied (with just matrix-vector products) on high-performance hardware; they are also a valuable general-purpose alternative to ILU for tough problems where ILU breaks down from instabilities.

Several algorithms for computing sparse approximations to $\mathbf{A}^{-1}$, or to its inverse triangular factors $\mathbf{L}^{-1}$ and $\mathbf{U}^{-1}$, have been proposed; see, e.g., [5, 6, 7, 19, 27, 30, 36]. Unfortunately, for linear systems arising from elliptic PDEs, there appears to be an inherent problem in the explicit nature of these preconditioners, a fundamental conflict between accuracy and sparsity. As problem sizes increase, their performance (either in terms of convergence rate at a fixed number of nonzeros per row, or storage required for a fixed convergence rate) quickly decreases.[1]

For a simple heuristic analysis of this problem, ignore boundary conditions. Suppose the elliptic PDE $\mathcal{L}u = f$ on domain Ω is discretized to $\mathbf{Au} = \mathbf{f}$ on points x_1, …, x_n in Ω, where the matrix $\mathbf{A}$ is the discrete form of the elliptic operator $\mathcal{L}$,

[*] Received by the editors June 15, 2000; accepted for publication (in revised form) January 5, 2001; published electronically July 10, 2001. This work was supported by the Natural Sciences and Engineering Council of Canada, and Communications and Information Technology Ontario (CITO), funded by the province of Ontario.

http://www.siam.org/journals/sisc/23-2/37378.html

[†] SCCM, Gates 2B, Stanford University, Stanford, CA 94305 (rbridson@stanford.edu).

[‡] Department of Computer Science, University of Waterloo, Waterloo, Ontario, N2L 3G1, Canada (wptang@elora.uwaterloo.ca).

[1] Of course, switching to an implicit preconditioner isn't guaranteed to solve the scalability problem; for example, standard ILU does not scale any better than no preconditioner, and the question of how best to use matrix orderings, dropping strategies, and numerical modifications to improve the scalability of ILU is still open [3, 4, 9, 14, 32, 33, 34].

and $u_i \approx u(x_i)$. (The discretization may use finite elements, finite volumes, or any other reasonable scheme.) The continuous solution may be written with the Green's function $\mathcal{G}(x,y)$ on $\Omega \times \Omega$ satisfying $\mathcal{L}\mathcal{G}(x,y) = \delta(x-y)$:

$$u(x) = \int_\Omega \mathcal{G}(x,y) f(y)\, dy$$

(assuming this exists, as one would expect for an elliptic problem well-posed enough to permit numerical solution). The discrete solution is similarly found with the matrix $\mathbf{A}^{-1}$ satisfying $\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$:

$$u_i = \sum_{j=1}^{n} A^{-1}_{ij} f_j.$$

Through this analogy, it is clear that $\mathbf{A}^{-1}$ is the discrete approximation to the Green's function.

Unfortunately, it is well known that though the Green's function decays away from its diagonal singularity, the decay may be slow especially for convective or indefinite problems (in fact, for problems with Neumann boundaries, $\mathcal{G}$ might not even decay to zero at all). The decay of the true Green's function is independent of the mesh size in the discretization, and so as the mesh is refined the number of large nonzeros in $\mathbf{A}^{-1}$ also increases, roughly like $O(n^{1+1/d})$, where n is the number of mesh nodes and d is the dimensionality of the problem.[2] This means an approximate inverse preconditioner cannot scale effectively with the problem size.[3]

To be more precise, one cannot scale in the standard basis, where u_i approximates $u(x_i)$. In [11, 18], the realization that the Green's function is smooth away from the diagonal suggested wavelets as alternate bases: they can compress smooth functions into high-quality sparse approximations, handle nonsmooth points (e.g., at the diagonal singularity, or arising from discontinuous coefficients), and provide fast and parallelizable transforms to and from the standard basis. The finer the mesh, the better the smoothness of $\mathcal{G}(x,y)$ can be exploited for compression, so the preconditioner may scale much more effectively.

The original paper [18] considered only classical wavelets, treating the discrete Green's function as a two-dimensional, regularly sampled, periodic image. For problems of dimensions greater than one (so the Green's function is of dimension greater than two) on irregular meshes with nonperiodic boundaries, this leads to significant problems. In particular, these "first generation" wavelets are constructed on regularly sampled one-dimensional periodic domains and so cannot hope to perform well on data coming from more complicated situations. This motivated the use of second generation wavelets [35] in [11] that naturally match such domains, while retaining the attractive properties of compression, tolerance of singularities, and fast transforms. The present article describes the multiresolution approximate inverse preconditioner of [11] and more recent developments.

[2] There are n columns in $\mathbf{A}^{-1}$, with column i containing a discrete approximation to $G(\cdot, x_i)$. There is a d-dimensional subregion $\Omega_L \subset \Omega$ such that $G(x, x_i)$ is large for $x \in \Omega_L$. There are $O(n^{1/d})$ mesh nodes contained in Ω_L, so column i of $\mathbf{A}^{-1}$ has roughly $O(n^{1/d})$ large entries, for a total of $O(n^{1+1/d})$ in $\mathbf{A}^{-1}$.

[3] Of course, this reasoning applies directly only to fully explicit approximate inverses: preconditioners in factored form perhaps may be more effective, since their product may be dense, though at the time of writing this has yet to be demonstrated.

The initialization phase

- Determine multiresolution bases α and β.
- Use a standard operator-based approximate inverse algorithm to find $\widetilde{\mathbf{Q}} \approx (\mathbf{M}_\beta^{-T}\mathbf{A}\mathbf{M}_\alpha^{-1})^{-1}$.

The application phase (called by a Krylov solver)

- Apply preconditioner to x by $y = \mathbf{M}_\alpha^{-1}(\widetilde{\mathbf{Q}}(\mathbf{M}_\beta^{-T}x))$.

FIG. 1. *The multiresolution approximate inverse method.*

Other authors have proved that discretizing elliptic PDEs with a wavelet basis in the finite element method (and using point or block diagonal preconditioners) can give optimal scalability, primarily working with classical wavelets and generalizations to handle boundaries and even dyadically refined unstructured meshes (see, e.g., [15, 16, 21, 22, 23, 24]) but also, e.g., with approximate wavelets derived from an existing hierarchical basis [37, 38]. The work in [21] actually worked with a full approximate inverse rather than just diagonal preconditioners and found significant improvements by taking into account interactions between different levels. The present paper differs by combining second generation wavelets with a more sophisticated approximate inverse, with an emphasis on heuristically finding good multilevel algorithm components (regrettably without theoretical results to confirm the promising first few numerical experiments).

2. Outline of the method. Since the Green's function is defined on the product space $\Omega \times \Omega$, it is natural to look for a wavelet basis that is a tensor product $\alpha \otimes \beta$ of wavelet bases α and β on Ω. In the discrete case, this means representing $\mathbf{A}^{-1}$ as

$$A_{ij}^{-1} = \sum_{k=1}^{n}\sum_{l=1}^{n} Q_{kl} a_i^k b_j^l,$$

where the separable basis functions are the product of elements $a^k \in \alpha$ and $b^l \in \beta$, and the coefficients are stored in matrix $\mathbf{Q}$. Equivalently,

$$\mathbf{A}^{-1} = \mathbf{M}_\alpha^{-1}\mathbf{Q}\mathbf{M}_\beta^{-T},$$

where $\mathbf{M}_\alpha^{-1}$ and $\mathbf{M}_\beta^{-1}$ have the basis functions of α and β, respectively, as their columns. Applying these operators to the standard basis vectors shows that $\mathbf{M}_\alpha$ and $\mathbf{M}_\beta$ are the transforms from the wavelet bases to the standard basis.

The transformed $\mathbf{A}^{-1}$, ready for compression, is $\mathbf{Q} = \mathbf{M}_\alpha\mathbf{A}^{-1}\mathbf{M}_\beta^T$. For the preconditioner, a highly sparse approximation $\widetilde{\mathbf{Q}}$ will be used. Rewriting $\mathbf{Q} = (\mathbf{M}_\beta^{-T}\mathbf{A}\mathbf{M}_\alpha^{-1})^{-1}$ shows that $\widetilde{\mathbf{Q}}$ can be obtained by applying a standard sparse approximate inverse algorithm to the transformed operator $\mathbf{M}_\beta^{-T}\mathbf{A}\mathbf{M}_\alpha^{-1}$—in particular, without knowledge of the true inverse $\mathbf{A}^{-1}$. Note that to avoid forming $(\mathbf{M}_\beta^{-T}\mathbf{A}\mathbf{M}_\alpha^{-1})$ explicitly, which may incur significant fill-in, an approximate inverse algorithm that works on a linear operator (not necessarily a matrix) is required. One example, used in this research, is SAINV [7].

To summarize, an overview of the multiresolution approximate inverse method is given in Figure 1.

The forward transform (standard basis → multiresolution basis)

- Start with the function values $f_1, \ldots, f_n$ at sample points $x_1, \ldots, x_n$.
- Let $\lambda_i^0 = f_i$ for all i, $\mathcal{C}^0 = \{x_1, \ldots, x_n\}$, and $j = 0$.
- Begin loop:
  - Split up the sample points $\mathcal{C}^j$ into two disjoint subsets, the fine nodes $\mathcal{F}^{j+1}$ and the coarse nodes $\mathcal{C}^{j+1}$.
  - Predict λ_F^j, the values at the fine nodes, from λ_C^j, the values at the coarse nodes, with some linear prediction operator $\mathbf{P}^j$: $\lambda_F^j \approx \mathbf{P}^j \lambda_C^j$.
  - Store the wavelet coefficient $\gamma_i^{j+1} = \lambda_i^j - (\mathbf{P}^j \lambda_C^j)_i$ for each fine node $x_i \in \mathcal{F}^{j+1}$.
  - Update the value at each coarse node by $\lambda_i^{j+1} = \lambda_i^j + (\mathbf{U}^j \gamma^{j+1})_i$ for each $x_i \in \mathcal{C}^{j+1}$ so that the required moments will be preserved. This update operator $\mathbf{U}^j$ must also be linear.
  - If $|\mathcal{C}^{j+1}|$ is small enough, below some constant, break out of the loop. Otherwise, set $j \leftarrow j + 1$ and continue.
- Return λ^j from the coarsest level along with the wavelet coefficients $\gamma^1, \ldots, \gamma^j$ from each level.

The inverse transform (multiresolution basis → standard basis)

- Start with λ^j and the wavelet coefficients $\gamma^1, \ldots, \gamma^j$.
- Begin loop:
  - Reconstruct λ_C^{j-1} at the coarse nodes by $\lambda_i^{j-1} = \lambda_i^j - (\mathbf{U}^{j-1}\gamma^j)_i$ for each $x_i \in \mathcal{C}^j$.
  - Reconstruct λ_F^{j-1} at the fine nodes by $\lambda_i^{j-1} = \gamma_i^j + (\mathbf{P}^{j-1}\lambda_C^{j-1})_i$ for each $x_i \in \mathcal{F}^j$.
  - Continue with $j \leftarrow j - 1$ until $j = 1$.
- Return $f_i = \lambda_i^0$ for all i.

FIG. 2. *The transform algorithms of the lifting scheme.*

3. The general algorithm.

3.1. The basis construction. The goal of the new basis $\alpha \otimes \beta$ is to convert "smoothness" in the standard basis to small coefficients that can be accurately approximated by zero. For irregular domains, the lifting scheme [35] for second generation wavelets is a natural choice. In this scheme, the basis is not constructed explicitly, but rather the forward transform algorithms (from the standard basis to the multiresolution basis, called $\mathbf{M}_\alpha$ and $\mathbf{M}_\beta$ above) are designed to directly achieve good compression along with easy invertibility. Figure 2 gives summaries of the transform algorithms.

The essential idea is that where a function is smooth, its values can be accurately predicted from nearby neighbors, and so storing the prediction error results in small coefficients except near "rough" regions. Doing this in a hierarchy of levels gives rise to a multiresolution representation: wavelet coefficients at level j correspond to features on the scale of the grid resolution of the set of nodes $\mathcal{C}^j$ at level j. It should be noted that on unstructured meshes (and possibly for other reasons mentioned below) the prediction operator for each fine node may have different weights—unlike classical wavelets, where one set of convolution weights is used throughout the domain.

The update step is required in signal processing to prevent aliasing, where for example a high-resolution singularity is propagated unchanged to lower resolutions; the update step is an averaging designed to make sure the signal is smooth enough to be faithfully represented at the next coarser level. However, in this context of compressing discrete Green's functions, this seems to be a liability, as demonstrated in [11]. In the ideal case, all wavelet coefficients in $\mathbf{Q}$ will be very small except on the diagonal, where the discrete Green's function must have a singularity: finding a sparse approximate inverse for a nearly diagonal matrix is simple. However, the update step smooths out this inherent singularity, smearing the large diagonal entries to off-diagonals, resulting in a harder task for a sparse approximate inverse algorithm.

On the other hand, without the update step the basis can be viewed as just a generalization of the standard hierarchical basis [1, 40] (if regular refinement and linear interpolation for prediction are used, it is exactly the hierarchical basis; see section 4 for more details). While optimal scalability has been demonstrated with classical wavelets in [15, 16, 21, 22, 23, 24], it is well known that with just diagonal or block diagonal preconditioning the hierarchical basis is not optimal: the condition number of the preconditioned system slowly grows with the number of unknowns, particularly in higher dimensions. To get the optimal performance of methods such as multigrid, the basis must be stabilized [37, 38], making the basis functions from different levels at least approximately orthogonal. This is essentially what the update step does, smoothing the function at each level so that coarser levels don't see the high-resolution features picked up at finer levels. Thus perhaps a theoretical analysis, beyond the scope of this paper, will show that the update step can be of value for multiple dimensions. However, the issue is further complicated by the fact that approximate inverses can be more effective in higher dimensions, where the Green's function decays faster and a sparse approximation is more feasible. For the rest of this paper we will not use the update step, leaving these questions for future research.

Without the update step, the forward transform is simplified, and in fact all wavelet coefficients at all levels can be computed simultaneously. The forward transform $\mathbf{M}$ can easily be written explicitly as a triangular matrix multiply and the inverse transform as a triangular solve, as in Figure 3. As long as each prediction operation can be done in constant time, i.e., each $\mathbf{P}^j$ has a bounded number of nonzeros per row, it is clear that the forward and inverse transforms can both be done in $O(n)$ time in serial. In terms of parallel computation, the forward transform is as good as a single sparse matrix multiply, whereas the inverse transform can naturally be done in $O(\log n)$ steps (with smaller and smaller matrix multiplies) assuming a geometric decline in the size of the $\mathcal{C}^j$.

There are two big issues that need to be resolved when constructing the basis. The first is how to coarsen; how to select the sets $\mathcal{C}^j$ of coarse nodes at each level. So far, we have concentrated on independent set heuristics similar to those used in unstructured multigrid [17] and algebraic multigrid [31]. The second issue is how to define the prediction operators $\mathbf{P}^j$ at each level. Standard linear interpolation or higher order polynomial interpolation is a possibility, but for robustness in difficult problems we have found more sophisticated techniques are necessary [11].

Looking at the compressed inverse $\mathbf{Q} = \mathbf{M}_\alpha \mathbf{A}^{-1} \mathbf{M}_\beta^T$, notice that the transform $\mathbf{M}_\alpha$ is being applied to each of the columns of $\mathbf{A}^{-1}$. From the equation $\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$, observe that column i of $\mathbf{A}^{-1}$ is actually the discrete solution to the PDE $\mathcal{L}u(x) = \delta(x - x_i)$. Thus the functions compressed by α satisfy the homogeneous PDE $\mathcal{L}u = 0$ almost

The forward transform (standard basis → multiresolution basis)

$$\begin{pmatrix}\gamma^1\\ \gamma^2\\ \vdots\\ \gamma^j\\ \lambda^j\end{pmatrix} = \mathbf{Mf} = \left(\begin{array}{c|c} \mathbf{I} & -\mathbf{P}^1 \\ \hline & \begin{array}{c|c} \mathbf{I} & -\mathbf{P}^2 \\ \hline & \begin{array}{c|c} \ddots & \vdots \\ \hline & \begin{array}{cc}\mathbf{I} & -\mathbf{P}^j\\ & \mathbf{I}\end{array}\end{array}\end{array}\end{array}\right) \begin{pmatrix}\mathbf{f}_{F1}\\ \mathbf{f}_{F2}\\ \vdots\\ \mathbf{f}_{Fj}\\ \mathbf{f}_{Cj}\end{pmatrix}$$

The inverse transform (multiresolution basis → standard basis)

$$\mathbf{f} = \mathbf{M}^{-1}\gamma = \left(\begin{array}{c|c} \mathbf{I} & -\mathbf{P}^1 \\ \hline & \begin{array}{c|c} \ddots & \vdots \\ \hline & \begin{array}{cc}\mathbf{I} & -\mathbf{P}^j\\ & \mathbf{I}\end{array}\end{array}\end{array}\right)^{-1} \begin{pmatrix}\gamma^1\\ \vdots\\ \gamma^j\\ \lambda^j\end{pmatrix}$$

FIG. 3. *The multiresolution basis transforms without update steps expressed in terms of triangular matrices.*

everywhere, which we can take as our definition of "smooth." Predicting the value at a fine node i from nearby coarse nodes can be done by solving a discrete form of $\mathcal{L}u(x_i) = 0$ using the coarse nodes as specified "boundary" data.

Similarly, the β transform is applied to the rows of $\mathbf{A}^{-1}$, which are discrete solutions to the adjoint problems, so β may be constructed with the adjoint operator $\mathcal{L}^*$ in mind. In particular, for self-adjoint problems it makes sense to take $\alpha = \beta$; for highly non-self-adjoint problems it will be important to have $\alpha \neq \beta$.

Finally it should be noted that for some problems—e.g., with oscillatory coefficients, strong indefiniteness, or complicated convection streamlines—it may be too difficult to construct very coarse yet useful representations of the Green's function. Though ideas from homogenization theory may help, it's likely that there will be a lower limit to the resolutions that are useful to consider. In this case, it is probably wisest to limit the multiresolution bases to a few levels and instead concentrate resources on the approximate inverse.

3.2. The approximate inverse. The transformed operator $\mathbf{M}_\beta^{-T}\mathbf{A}\mathbf{M}_\alpha^{-1}$ may be multiplied out explicitly at which point any approximate inverse algorithm may be used. However, in doing so substantial extra fill-in is incurred, increasing the cost of the preconditioner construction and application as well as storage requirements. A more attractive route is to use an approximate inverse algorithm that doesn't require explicit knowledge of the matrix and thus can precondition an operator known only in this factored form.

Many popular algorithms can, in their simplest form, be adapted to this context. One example is the minimal residual (MR) method of [19] which requires no modification. However, other approximate inverses require more thought. For example,

```
• Take as input matrix $\mathbf{A}$ of dimension $n$ and a drop tolerance $\delta$, generally
  around 0.1 or 0.01.
• Set $\mathbf{W} \leftarrow \mathbf{I}$ and $\mathbf{Z} \leftarrow \mathbf{I}$.
• for $j = 1$ to $n$ do
    • Compute $l = \mathbf{A}Z_{:,j}$ and $u = \mathbf{A}^T W_{:,j}$.
    • Set $D_{jj} \leftarrow (u^T Z_{:,j})^{-1}$, the inverse pivot.
    • Rescale $L = lD_{jj}$ and $U = uD_{jj}$.
    • Compute the biconjugation coefficients
      $C_W = L^T W_{:,j+1:n}$ and $C_Z = U^T Z_{:,j+1:n}$.
    • Update the remaining columns of $\mathbf{W}$ and $\mathbf{Z}$ with sparsified outer-
      products (i.e., only updating with the entries of magnitude greater
      than $\delta$): $W_{:,j+1:n} \leftarrow W_{:,j+1:n} - \mathit{sparsify}(W_{:,j}C_W)$
      and $Z_{:,j+1:n} \leftarrow Z_{:,j+1:n} - \mathit{sparsify}(Z_{:,j}C_Z)$.
```

FIG. 4. *The SAINV algorithm, using MATLAB colon notation for submatrices.*

a sparsity pattern must be specified a priori for some methods, and it is not clear how to do so for efficient and robust performance here. Avoiding this issue, we have chosen to adopt SAINV [5, 6, 7] for this first study, which constructs the sparsity pattern during construction. In particular we use an outer-product-based version of the algorithm [13] that doesn't require any knowledge of the sparsity pattern of the operator for efficient performance. We note that for efficiency during the construction phase, the basis transforms and matrix multiplies must be done in fully sparse mode. Figure 4 gives the algorithm, which by biconjugation (along the lines of modified Gram–Schmidt) applied to two copies of the identity matrix constructs upper triangular matrices $\mathbf{W}$ and $\mathbf{Z}$ and a diagonal matrix $\mathbf{D}$ such that $\mathbf{A}^{-1} \approx \mathbf{ZDW}^T$. It can be simplified in the case of symmetric matrices to construct just $\mathbf{Z}$ (which is equal to $\mathbf{W}$) with half the work and storage. It has the advantage that for positive definite matrices it is guaranteed to produce a positive definite preconditioner, though breakdown is possible in the general case, and has generally been shown to be very robust [7].

One important issue for factored approximate inverses is the ordering of the rows and columns of the matrix. As demonstrated in [8, 12, 13], performance can be significantly improved by an appropriate reordering—e.g., nested dissection (we use the Metis routine [29]). On the other hand, one might argue that if the multiresolution bases here are constructed correctly, the transformed $\mathbf{A}$ will be well enough conditioned that ordering isn't needed. However, it seems doubtful that the multiresolution framework will be robust enough to handle all problems on its own. What we desire for tough problems is a multiresolution basis construction algorithm which "fails gracefully," i.e., never makes $\mathbf{A}$ worse conditioned even though it may not provide adequate improvement. In this case, the power of the approximate inverse will hopefully show through, provided we have taken care of the ordering.

3.2.1. Ordering. Unfortunately, typical ordering algorithms require the explicit structure of the matrix, so this is a nontrivial step in this context; some analysis is required.

Before going further, recall the graph theory notation often used in sparse matrix ordering. With a given $n \times n$ matrix $\mathbf{B}$, associate the graph $G_{\mathbf{B}}$, or simply G if the context makes it clear, defined on nodes $\{1,\dots,n\}$ with a directed edge $i \to j$ if and only if $B_{ij} \neq 0$ (for $i \neq j$). Thus the nonzero structure of $\mathbf{B}$ and the graph $G_{\mathbf{B}}$ may be identified. As an abbreviation, write $i \to j$ to mean the statement that the directed edge $i \to j$ exists in G. The neighborhood of a node i is the set of nodes j such that $i \to j$. A path is a sequence of distinct nodes $i_1, \dots, i_k$ such that $i_1 \to i_2$, $i_2 \to i_3, \dots,$ and $i_{k-1} \to i_k$, often written $i_1 \to \cdots \to i_k$, or simply $i_1 \rightsquigarrow i_k$. The transitive closure G^* of a graph G is one constructed on the same nodes but having $i \to j$ whenever $i \rightsquigarrow j$ in G. For a fuller treatment, see [25, 26].

As is shown in [26], assuming here and for the rest of this section that there is no felicitous cancellation, the structure of $\mathbf{B}^{-1}$ is given by the transitive closure of the graph of $G_{\mathbf{B}}$. As shown before, the forward transform $\mathbf{M}_\alpha$ can be simply expressed as a triangular matrix (when there are no update steps). Then the graph of $\mathbf{M}_\alpha$ satisfies $i \to j$ if and only if at some level i there is a fine node whose prediction uses coarse node j. Therefore the graph of $\mathbf{M}_\alpha^{-1}$ has $i \to j$ if and only if there is a chain of prediction dependencies $i \rightsquigarrow j$.

Define the support of a node j to be the set $\operatorname{supp}(j)$ of nodes i such that $(\mathbf{M}_\alpha^{-1})_{ij} \neq 0$—this is actually the support of the jth multiresolution basis function. From the transitive closure characterization of inverses, observe that the supports have a nested structure: if $i \in \operatorname{supp}(j)$, then $\operatorname{supp}(i) \subset \operatorname{supp}(j)$. Notice that if j is a fine node at the highest resolution level, $\operatorname{supp}(j) = \{j\}$, but that if j is at the lowest resolution level, its support may be very dense—showing that it is important to not multiply out the inverse transform explicitly.

Now examine the structure of $\mathbf{M}_\beta^{-T}\mathbf{A}\mathbf{M}_\alpha^{-1}$. Assume that $\mathbf{A}$ has symmetric structure ($A_{ij} \neq 0$ if and only if $A_{ji} \neq 0$) and $\mathbf{M}_\beta$ and $\mathbf{M}_\alpha$ have the same structure, i.e., that the two bases have the same hierarchy of levels and the same prediction dependencies.[4] Then the product has symmetric structure, and one can speak unambiguously about coarse/fine nodes and the support of a node. Observe

$$\begin{aligned}(\mathbf{M}_\beta^{-T}\mathbf{A}\mathbf{M}_\alpha^{-1})_{ij} &= \sum_{k=1}^{n}\sum_{l=1}^{n}(\mathbf{M}_\beta^{-T})_{ik}A_{kl}(\mathbf{M}_\alpha^{-1})_{lj}\\ &= \sum_{k=1}^{n}\sum_{l=1}^{n}(\mathbf{M}_\beta^{-1})_{ki}A_{kl}(\mathbf{M}_\alpha^{-1})_{lj}.\end{aligned}$$

Then $(\mathbf{M}_\beta^{-T}\mathbf{A}\mathbf{M}_\alpha^{-1})_{ij} \neq 0$ if and only if there exist nodes k and l with $k \in \operatorname{supp}(i)$, $l \in \operatorname{supp}(j)$, and $k \to l$ in $\mathbf{A}$. In other words, $i \to j$ in the product if and only if their supports are adjacent in $\mathbf{A}$. Using the nested structure of the supports, it is then clear that the neighborhood of any node j contains the neighborhoods of all nodes in $\operatorname{supp}(j)$.

Now, the location of nonzeros in column i of the upper inverse triangular factor $\mathbf{Z}$ of a symmetrically structured matrix $\mathbf{B}$ can be characterized as follows [12]: $\mathbf{Z}$ has a nonzero for each node before i and reachable from i via paths in $\mathbf{B}$ using nodes before i.

Consider the effect of swapping the positions of $i \neq j$ in some ordering, when $i \in \operatorname{supp}(j)$. Clearly the number of nonzeros in columns in $\mathbf{Z}$ ordered before both

[4] So far, relatively good results have been obtained under this assumption, which makes the following analysis much easier. However, it may prove useful to relax this requirement for convection-dominated problems, where predicting from the upwind nodes suggests that the structure of M_α and M_β should be different: convection for the adjoint problem (handled by β) is in the opposite direction from the original problem (handled by α).

```
• Take as input the structure of $\mathbf{M}_\alpha$ or $\mathbf{M}_\beta$ (multiplied out).
• For $i = 1, \ldots, n$
    • Set $numdep(i)$ = number of nodes $j$ with $j \to i$, not including $i$ itself.
    • Set $waiting(i)$ to false.
• Initialize a queue with room for $n$ entries, empty at first.
• Set $i = 1$, the first node to attempt to order.
• Set $j = 1$, the first index into the modified ordering $p$.
• While $j \leq n$
    • If the queue is not empty then
        • Remove the first node $k$ from the front of the queue.
        • Set $p_j = k$ and $j \leftarrow j + 1$.
        • Consider, in order, each $l \neq k$ with $k \to l$ and $waiting(l)$ true;
          decrement $numdep(l)$, and if this is 0 set $waiting(l)$ to false and
          append $l$ to the queue.
    • Else if $numdep(i) = 0$ then
        • Set $p_j = i$, $j \leftarrow j + 1$, and $i \leftarrow i + 1$.
        • Consider, in order, each $l \neq i$ with $i \to l$ and $waiting(l)$ true;
          decrement $numdep(l)$, and if this is 0 set $waiting(l)$ to false and
          append $l$ to the queue.
    • Else $(numdep(i) > 0)$
        • Set $waiting(i)$ to true, and $i \leftarrow i + 1$.
• Return the modified ordering $p$.
```

FIG. 5. *Modifying an ordering to respect the multiresolution basis.*

i and j or after both will not be changed. However, the columns in between may be altered. Since the neighborhood of j contains the neighborhood of i, any nodes reachable on paths through i are reachable through j, but not necessarily the other way around. Therefore ordering i before j can't result in more nonzeros in $\mathbf{Z}$, but putting j before i might.

Thus any ordering of the nodes should respect j ordered after all other nodes in $\operatorname{supp}(j)$. Since $\operatorname{supp}(j)$ is the set of i such that $(\mathbf{M}_\alpha^{-1})_{ij} \neq 0$, this is equivalent to requiring that i be ordered before j whenever $i \rightsquigarrow j$ in $\mathbf{M}_\alpha$. This is clearly equivalent to ordering i before j whenever $i \to j$ in $\mathbf{M}_\alpha$, which can be enforced by the algorithm in Figure 5.

Essentially the algorithm outputs the nodes in the existing order except when a coarse node comes before any of its fine dependents. Then the coarse node is made to wait until all the fine dependents have been ordered, at which point it's put on a queue to be ordered as soon as possible. The value $numdep(i)$ serves as a counter of how many fine nodes dependent on i have yet to be ordered—since i is put into p only when this reaches zero, the ordering must be consistent.

The initialization loop, assuming sparse storage of the matrix, takes time on the order of the number of nonzeros in the matrix, which should be $O(n)$. The complexity of the main loop is a little more difficult to prove.

First note that both i and j begin at 1 and are never decremented. Let $d = \sum_{i=1}^{n} numdep(i)$, so before the main loop begins $d = nnz(\mathbf{M}_\alpha) - n$, the number of off-diagonal nonzeros in $\mathbf{M}_\alpha$. Values in *numdep* are never incremented, so d never increases.

A node can only be marked as waiting in the final else clause, and since i is incremented there it can never be marked as waiting again. The only way an entry in *numdep* can be decremented to zero is if it had been marked as waiting, and when it hits 0 its marked as not waiting, so it can never be decremented past 0. Therefore d is always nonnegative.

Suppose i is incremented past $n+1$—this can only happen if $i = n+1$ at the start of an iteration with the queue empty. There must be some unordered nodes left, as otherwise j would have been incremented past n and the loop would have stopped. If any of the unordered nodes had *numdep* equal to zero, they either would have started at zero, in which case the first else clause would have been executed for that value of i, or would have been decremented to zero and added to the queue—in either case implying that they must now be ordered, a contradiction. Thus all the unordered nodes have positive *numdep* counters. However, some unordered node v must be from the finest resolution level of all unordered nodes and so cannot have any unordered dependent fine nodes—and so must have $\mathit{numdep}(v) = 0$, a contradiction. Therefore i never is incremented past $n + 1$.

Clearly j can never be incremented past $n + 1$ thanks to the loop condition. Therefore, since in each iteration either j is incremented, i is incremented, or at least one of the values in *numdep* is decremented, there can be at most $n + \mathit{nnz}(\mathbf{M}_\alpha)$ iterations. In fact, assuming constant time queue operations (e.g., as in a simple array implementation) the time spent in the main loop is $O(n) + O(\mathit{nnz}(\mathbf{M}_\alpha))$, which again should be $O(n)$. Thus the entire algorithm runs in $O(n)$ time.

Now consider the following simple scheme: order $\mathbf{A}$ with nested dissection and then run the above algorithm to make the ordering consistent with the multiresolution basis. The only worry is that the modification will destroy the good fill-reducing qualities of the original ordering. However, the bulk of the nodes should be at the finest level and thus have trivial supports, so the modification can't change their relative order. The only nodes that can be greatly affected by the ordering modification are the very coarse nodes, which are in a very small minority. Thus the potential damage is very limited. Experiments have confirmed that this isn't much worse (but far cheaper) than applying nested dissection to the multiplied out $\mathbf{M}_\beta^{-T}\mathbf{A}\mathbf{M}_\alpha^{-1}$.

4. Relationships with other methods. Before proceeding to our actual implementation and testing for unstructured two-dimensional problems, it is instructive to compare the new algorithm with some other multiresolution methods.

As mentioned before, the basis transforms can be expressed as triangular matrices with unit block diagonals, so the algorithm could be viewed as a highly parallel variant of multilevel ILU (e.g., [4, 3, 9, 32, 33, 34]) with an approximate inverse replacing $\mathbf{D}^{-1}$ for the approximate LDU factorization.

Another viewpoint comes from noting that the operators $\binom{\mathbf{P}_\alpha}{\mathbf{I}}$ and $(\mathbf{P}_\beta^T\ \mathbf{I})$ within the transforms for α and β correspond to node-nested multigrid's prolongation and restriction, respectively. The application of the preconditioner can then be thought of as the multigrid-like algorithm in Figure 6. The key difference between this and multigrid is that the smoothing is performed in one step, and only at the coarsest level for each variable, instead of being interleaved with restriction and prolongation. (See [36] for an example of approximate inverses used as smoothers in multigrid.) This is similar to but not exactly the same as additive multigrid, i.e., BPX [10].

The hierarchical basis preconditioners (see, e.g., [1, 2, 40]) are very similar to the

- Successively restrict the function to coarser and coarser levels by $\mathbf{M}_\beta^{-T}$.
- Smooth all variables at their coarsest level only—including couplings between variables at different levels—by $\widetilde{\mathbf{Q}}$, possibly doing an exact solve on the coarsest level if the approximate inverse is dense enough there.
- Prolong the smoothed multiresolution representation back to the original variables by $\mathbf{M}_\alpha^{-1}$.

FIG. 6. *A multigrid-like interpretation of the multiresolution approximate inverse algorithm.*

new preconditioner. In these, the original system is transformed into a new multiresolution basis, and a simple preconditioner such as block Jacobi is applied. Extending this, the multiresolution approximate inverse naturally works with unsymmetric bases ($\alpha \neq \beta$) better adapted to the problem and also allows for coupling between variables at different levels in the preconditioner $\widetilde{\mathbf{Q}}$.

More sophisticated multiresolution bases (but otherwise essentially the same algorithm as the hierarchical basis method) are used in wavelet methods; see, e.g., [15, 16, 21, 22, 23, 24, 37, 38]. In particular, these bases are more stable than simple hierarchical bases, in the sense that the multiresolution norm is equivalent to the standard norm, which results in optimal scalability. As we mentioned before when discussing the update step in the present method, this issue hasn't been resolved here, and it appears that we currently achieve only the suboptimal scalability of the hierarchical basis method.

Finally, there have already been proposed wavelet–approximate inverse combinations in [18] and [21]. Both of these works used classical wavelets, though the latter featured a generalization which correctly treats nonperiodic boundary conditions.

5. Implementation.

This section illustrates two ways to generate the multiresolution basis, one geometric and one algebraic. The important thing to keep in mind here is not the exact heuristics used but rather that exactly the same techniques used for other node-nested unstructured multilevel methods are used here. The new viewpoint of compressing the discrete Green's function provides additional insight, but this part of the problem is the same.

5.1. Geometric implementation in two dimensions.

In this section, we describe a geometric-oriented implementation for scalar second-order elliptic problems on unstructured triangular meshes. We restrict ourselves to two dimensions since the geometric complexity of remeshing in three dimensions is daunting.

The first issue to be dealt with is discretization. In this geometric implementation, the PDE is rediscretized on coarser and coarser meshes, so it is imperative to have a discretization which is stable and reasonably accurate even for large meshes. In this paper upwinding is used for convection and harmonic averaging for diffusion; more accurate schemes such as [39] could be used instead. Obviously there could be aliasing problems with highly oscillatory coefficients with the simple rediscretization used here—without resorting to algebraic methods, the only solution would be an analytic homogenization, but we have not investigated this.

The second issue is how to choose coarse nodes. Of course, in some applications an appropriate hierarchy of nested meshes is already available, but in general an automatic procedure for generating the hierarchy is needed. The simplest approach, used for multigrid in [17], is to consider the graph of a triangulation of the current set of nodes and to select a greedily chosen maximal independent subset as the next coarser level. These nodes can then be retriangulated for the next coarsening. Under the assumption that the edges of the triangulation represent strong couplings between unknowns, the maximality condition ensures that every fine node has at least one strongly coupled coarse node from which it can be predicted, and the independence condition ensures that there won't be too many coarse nodes.

This assumption breaks down for anisotropic PDEs or anisotropic meshes; "semicoarsening" is needed here, where coarsening takes place only in the directions of strong coupling. Heuristically this can be implemented by rediscretizing the PDE on the coarser mesh and then disregarding the edges corresponding to small off-diagonal nonzeros when constructing the maximal independent set. Then every fine node is guaranteed to have a strongly coupled coarse node, where the strength of coupling is measured by the size of the nonzero in the discretization. A reasonable measure of coupling strength is, for example,

$$|A_{ij}| + |A_{ji}| > \epsilon||(|A_i| + |A_j|)||$$

for some norm of the matrix columns, and with $\epsilon = 0.1$, say. All reasonable heuristics appear to work equally well after a little tuning of ϵ on small test problems.

The other problem with anisotropic PDEs lies in retriangulation of the coarse nodes. At coarse levels with anisotropically distributed nodes, Delaunay triangulation (which ignores the PDE, of course) may produce very poor meshes which don't reflect the anisotropy. Some form of coefficient-adapted triangulation is needed, such as breaking the region into subregions with more-or-less constant coefficients, changing coordinates in each subregion to make the PDE isotropic, Delaunay triangulating in the new coordinates, and then stitching the triangulated subregions back together.

The final issue concerns how to do prediction. The most robust technique is PDE prediction, where as discussed earlier the value at a fine node is taken from the approximate solution of the homogeneous PDE or its adjoint with neighboring coarse nodes as Dirichlet boundary data. With unstructured triangular meshes this is easily done by triangulating the fine node together with the few surrounding coarse nodes, then rediscretizing the PDE at just the fine node to give a single linear equation for the value there. To guarantee sparsity in the prediction operator, the coarse nodes are selected as the vertices of the coarse triangle containing the fine node, possibly with any of the three vertices on the other sides of the triangle's edges according to the Delaunay criterion for edge-swapping. If the fine node is on a convex boundary, the coarse triangle that comes nearest to containing it is used. Note that these neighboring coarse nodes can be found in $O(1)$ time by doing a breadth-first search from the fine node in the fine mesh. Also we restrict the retriangulation so that at most six coarse nodes are used to predict the fine node, thus guaranteeing a fast (linear time) construction, sparse basis transforms, and agreement with stretched meshes (see Figure 7 for what could go wrong with unrestricted retriangulation).

These same issues appear in any unstructured multilevel algorithm and, in particular, algorithms developed for coarsening and interpolation in multigrid, etc., can be used here, just as the algorithms given above could be used for other multilevel methods.

5.1.1. Sample results. In [11] several example problems were given, showing that for a variety of two-dimensional problems the new method is superior to a plain approximate inverse. For the same storage (including the basis transforms as well as the approximate inverse) and similar flop counts per iteration, the multiresolution

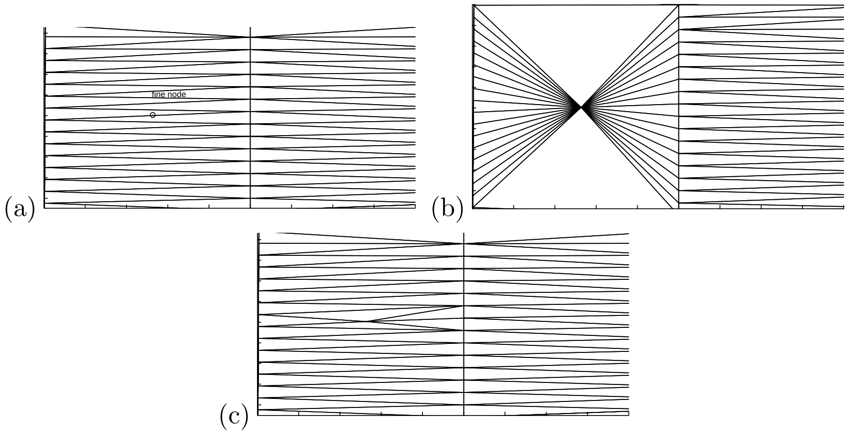


Fig. 7. *When retriangulating around a fine node in a stretched mesh* (a)*, problems can arise if no limits are put on the Delaunay insertion algorithm since many distant coarse nodes are connected to the fine node* (b)*, but a restricted retriangulation solves this issue and is also more efficient* (c)*.*

algorithm provided convergence that was several times faster, even on the smallest problems. As problem sizes increased, the number of iterations for the multiresolution method grew much more slowly than for the plain approximate inverse. (Regrettably the convergence still wasn't mesh-independent but appeared to grow like the number of levels squared, similar to hierarchical basis methods. As noted in the first part, perhaps further analysis of the update step will produce an optimal preconditioner.)

We first present results for three problems from [11] in Table 1. HEAT is a large (10 units) backward-Euler timestep of the heat equation $u_t = \nabla^2 u$ on the unit disc with Neumann boundary conditions $\nabla u \cdot \hat{n} = \mathrm{sign}(\cos(20\theta))$ and a previous timestep of $u = 0$ for $x < 0$ and $u = 1$ for $x > 0$ on an exponentially stretched mesh. ANISO is an anisotropic discontinuous problem from [20], with $1000u_{xx} + u_{yy} = f$ on the southwest and northeast quarters of the unit square, $u_{xx} + 1000u_{yy} = f$ on the others, $f = \sin(10\pi y)$, homogenous Neumann boundaries for $y > 0.25$, and the Dirichlet boundary condition $u = x$ for $y < 0.25$. REACTOR is a discontinuous indefinite problem on the unit disc of the form $\nabla \cdot K\nabla u + cu = f$, with $K = 1$, $c = 0.3$, $f = -1$ in 21 small interior discs, $K = 0.005$, $c = -0.2$, $f = -1$ for the rest of the inner disc $r < 0.9$, and $K = 10^{-6}$, $c = 0$, $f = 0$ for $r > 0.9$.

The multiresolution bases included enough levels so that the coarsest had about 100 nodes. Drop tolerances in AINV were selected to give approximately the same storage (including prediction operators) for each preconditioner: $\approx 7n$ nonzeros for a problem with n nodes. CG was used for the SPD problems and Bi-CGSTAB for the rest, with convergence flagged when the 2-norm of the residual was reduced by a factor of 10^{-6} from a starting guess of all zeros, giving up at 1000 iterations.

However, two examples of difficulties for the geometric approach arose in ANISO without the special coefficient-adapted retriangulation and in ROTATE. The latter is a non-self-adjoint, convection-dominated problem based on a solid-body rotation of a disc (circular streamlines); see Table 2. ROTATE couldn't be effectively solved with a complete hierarchy; the best results were obtained with only two coarse levels, precluding any improvement in the asymptotic rate of convergence.

5.2. An algebraic alternative. Some of the difficulties encountered in solving ANISO and ROTATE are really just artifacts of trying to rediscretize the problem

Table 1

Iteration counts for example problems of varying sizes, with the standard basis and a problem-adapted multiresolution basis for AINV. The number of unknowns starts at $n = 4939$ for HEAT, $n = 900$ for ANISO, and $n = 4195$ for REACTOR.

Problem	Method	n	$\approx 4n$	$\approx 16n$
HEAT	Standard	125	215	432
	Multiresolution	23	25	28
ANISO	Standard	37	67	111
	Multiresolution	12	14	18
REACTOR	Standard	181	355	744
	Multiresolution	89	141	132

Table 2

Iteration counts for examples of difficulties in geometric approach. Delaunay retriangulation ignoring anisotropy causes problems for ANISO; attempting to coarsely discretize curved streamlines causes problems for ROTATE. The number of unknowns starts at $n = 900$ for ANISO and at $n = 1195$ for ROTATE.

Problem	n	$\approx 4n$	$\approx 16n$	$\approx 64n$
ANISO	350	545	*	*
ROTATE	73	135	297	879

on very coarse triangular meshes. A simple triangular mesh cannot easily match the changing anisotropies of ANISO or curved streamlines of ROTATE, yet both of these problems would appear to permit very coarse representations. This motivates the use of algebraic methods for basis construction, where no auxiliary meshes are used; everything is generated from the original matrix alone, hopefully avoiding geometric pitfalls in doing so. This is also an advantage in three dimensions, where unstructured remeshing can be difficult.

For the prediction operators, we first decide which coarse nodes will be used to predict each fine node: we select the strongly coupled coarse nodes that are adjacent to either the fine node or one of its fine neighbors. Next we determine the weights for each coarse node in the prediction.

The simplest method we try, labeled M1, is to predict the fine node value as a weighted mean of the coarse node values, with (positive) weights proportional to the magnitude of the appropriate off-diagonal entries in the matrix $\mathbf{A}$.

A potentially more accurate method, M2, is based on solving the homogeneous PDE at the fine node with boundary values specified at the surrounding coarse nodes, as in the geometric approach. In fact, the matrix gives us an equation for each node involving it and its neighbors. Unfortunately, the equation at the fine node in general involves neighboring fine nodes as well as coarse nodes, and so we cannot stop here. Including the equations at those fine nodes would again, in general, involve more fine nodes or coarse nodes we're not using in the prediction, so the system still won't be closed. However, we can use method M1 to interpolate these unknown values from the coarse neighbors instead and then solve the closed system for the desired fine value. This is similar to element-free AMGe [28], where an additional layer of nodes is used.

The next issue is how to generate the discretized matrix at coarser levels than the original. While these matrices are not used in the preconditioner, they are required to generate the prediction operators using the above schemes and allow us to use the maximal independent set algorithm for coarse node selection from the previous section. The most natural choice for these coarser versions of $\mathbf{A}$ is the Petrov–Galerkin approximation to the Schur complement, as in multigrid:

$$\mathbf{A}_{coarse} = (\mathbf{P}_\beta^T\ \mathbf{I})\mathbf{A}\begin{pmatrix}\mathbf{P}_\alpha \\ \mathbf{I}\end{pmatrix}.$$

Unfortunately we encountered a difficulty with this approach: for unstructured problems with a reasonable number of coarse nodes used to predict each fine node, the coarse versions of **A** quickly become dense. From a finite element perspective, the supports of the coarse basis functions have too much overlap. Perhaps with more tuning of the strong connection heuristic in the coarse node selection this could have been averted, but we looked for a more automatic approach instead.

The bulk of the extra nonzeros in the coarse versions of **A** are very small, and thus a viable approach is to simply filter out the small nonzeros at each level (perhaps with diagonal compensation) as is done in multilevel ILU [4, 3, 9, 32, 33, 32]. However, for anisotropic problems such a filter may be unreliable, destroying essential topology in the problem—it cannot distinguish between the small, negligible entries resulting from excessive overlap of coarse basis functions and the small but nonnegligible entries representing weak couplings in the original PDE (that increase in relative strength as semicoarsening proceeds).

Our solution is to use two sets of prediction operators. One is stored for the basis transform, and the other is used temporarily just to generate the coarsened matrices. The prediction operators in the second set are much sparser, with structures chosen so that excessive fill-in is impossible (e.g., if the initial matrix has a planar graph, so do the coarsened matrices). They are poorer quality than the operators in the first set, but since they are better adapted to the problem than simple polynomial interpolation they should be superior to a standard rediscretization on the coarse nodes.

The nonzero structure of the second set of prediction operators is determined in two stages. In the first stage, each fine node is assigned the coarse node to which it is most strongly coupled, giving one nonzero per row in the prediction operator. In this way, the fine nodes are disjointly partitioned into clusters around the coarse nodes. From the finite element method perspective this guarantees at most unit element overlap between coarse basis functions. From the graph theory perspective this guarantees that the graph of the coarse matrix is the result of a sequence of edge contractions (the edges coupling each fine node to its chosen coarse node) from the original matrix—and this means that graph properties such as planarity or being a triangulation are preserved. In the second stage, additional nonzeros are added to improve the quality of prediction but only when they don't incur any extra fill in the coarse matrix. A greedy algorithm is used, considering the coarse nodes in order of how few fine dependencies they have, adding as many connections to the neighboring fine nodes (in order of connection strength) as possible.

5.2.1. Sample results. We now present the results of using the algebraic approach in solving ANISO and ROTATE. Table 3 gives iteration counts for these problems with both prediction methods. Clearly there is more research to be done as M2 is better than M1 for ANISO but unexpectedly worse for ROTATE.

6. Conclusions.

We have presented a new preconditioner designed for the high-performance solution of linear systems derived from elliptic PDEs. We combine the scalability of multiresolution methods with the robustness of approximate inverses to give something useful for large problems on unstructured meshes with anisotropies, strong convection, or even indefinite reaction terms. The key idea is to create a

Table 3

Iteration counts for example problems of varying sizes, solved with algebraic methods. The number of unknowns starts at $n = 961$ for ANISO and at $n = 307$ for ROTATE.

Problem	Method	n	$\approx 2n$	$\approx 4n$	$\approx 8n$
ANISO	M1	12	15	17	19
	M2	10	10	13	13
ROTATE	M1	10	14	19	23
	M2	24	32	41	63

sparse approximate inverse expressed in a multiresolution basis which compresses the discrete Green's function.

In implementing the method we have worked with a factored approximate inverse algorithm, solving the problem of ordering the unknowns in the new basis. We have also investigated both geometrical and algebraic methods for constructing the basis for unstructured problems.

Unfortunately it appears that the method doesn't scale any better than hierarchical basis methods; while this is much better than simple approximate inverses, it is suboptimal. However, the power of the approximate inverse in addition to problem-adapted interpolation means convergence is generally better than hierarchical basis methods. Robustness is particularly gained for problems, where an effective complete multilevel decomposition cannot be found: the new method can truncate the hierarchy at an appropriate level and the approximate inverse can take care of the rest.

As a result, we don't expect the method to be competitive with a well-tuned multigrid algorithm for well-behaved problems, but it may be of use for more difficult problems where robustness is critical.

REFERENCES

[1] R. E. Bank and T. Dupont, *Analysis of a Two-Level Scheme for Solving Finite Element Equations*, report CNA-159, Center for Numerical Analysis, University of Texas, Austin, TX, 1980.

[2] R. E. Bank, T. Dupont, and H. Yserentant, *The hierarchical basis multigrid method*, Numer. Math., 52 (1988), pp. 427–458.

[3] R. E. Bank and R. K. Smith, *The incomplete factorization multigraph algorithm*, SIAM J. Sci. Comput., 20 (1999), pp. 1349–1364.

[4] R. Bank and C. Wagner, *Multilevel ILU decomposition*, Numer. Math., 82 (1999), pp. 543–576.

[5] M. Benzi, C. D. Meyer, and M. Tůma, *A sparse approximate inverse preconditioner for the conjugate gradient method*, SIAM J. Sci. Comput., 17 (1996), pp. 1135–1149.

[6] M. Benzi and M. Tůma, *A sparse approximate inverse preconditioner for nonsymmetric linear systems*, SIAM J. Sci. Comput., 19 (1998), pp. 968–994.

[7] M. Benzi, J. K. Cullum, and M. Tůma, *Robust approximate inverse preconditioning for the conjugate gradient method*, SIAM J. Sci. Comput., 22 (2000), pp. 1318–1332.

[8] M. Benzi and M. Tůma, *Orderings for factorized sparse approximate inverse preconditioners*, SIAM J. Sci. Comput., 21 (2000), pp. 1851–1868.

[9] E. F. F. Botta and F. W. Wubs, *Matrix renumbering ILU: An effective algebraic multilevel ILU-preconditioner for sparse matrices*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 1007–1026.

[10] J. Bramble, J. Pasciak, and J. Xu, *Parallel multilevel preconditioners*, Math. Comp., 55 (1990), pp. 1–22.

[11] R. Bridson, *Multi-Resolution Approximate Inverses*, Master's thesis, University of Waterloo, Waterloo, Ontario, Canada, 1999. Available from http://www.lib.uwaterloo.ca/ETD/etheses.html.

[12] R. Bridson and W.-P. Tang, *Ordering, anisotropy, and factored sparse approximate inverses*, SIAM J. Sci. Comput., 21 (1999), pp. 867–882.

[13] R. Bridson and W.-P. Tang, *Refining an approximate inverse*, J. Comput. Appl. Math, 123 (2000), pp. 293–306.

[14] R. Bridson and W.-P. Tang, *A structural diagnosis of some IC orderings*, SIAM J. Sci. Comput., 22 (2000), pp. 1527–1532.

[15] C. Canuto, A. Tabacco, and K. Urban, *The wavelet element method. Part* I: *Construction and analysis*, Appl. Comput. Harmon. Anal., 6 (1999), pp. 1–52.

[16] C. Canuto, A. Tabacco, and K. Urban, *The wavelet element method. Part* II: *Realization and additional features in* 2D, Appl. Comput. Harmon. Anal., 8 (2000), pp. 123–165.

[17] T. Chan and B. Smith, *Domain decomposition and multigrid algorithms for elliptic problems on unstructured meshes*, Contemp. Math., 180 (1994), pp. 175–189.

[18] T. Chan, W.-P. Tang, and W. L. Wan, *Wavelet sparse approximate inverse preconditioners*, BIT, 37 (1997), pp. 644–660.

[19] E. Chow and Y. Saad, *Approximate inverse preconditioners via sparse-sparse iterations*, SIAM J. Sci. Comput., 19 (1998), pp. 995–1023.

[20] S. Clift, H. Simon, and W.-P. Tang, *Spectral Ordering Techniques for Incomplete LU Preconditioners for CG Methods*, manuscript.

[21] A. Cohen and R. Masson, *Wavelet methods for second-order elliptic problems, preconditioning, and adaptivity*, SIAM J. Sci. Comput., 21 (1999), pp. 1006–1026.

[22] W. Dahmen, *Wavelet Methods for PDEs—Some Recent Developments*, IGPM report 183, RWTH, Aachen, Germany, 1999.

[23] W. Dahmen and A. Kunoth, *Multilevel preconditioning*, Numer. Math., 63 (1992), pp. 315–344.

[24] W. Dahmen and R. Stevenson, *Element-by-element construction of wavelets satisfying stability and moment conditions*, SIAM J. Numer. Anal., 37 (1999), pp. 319–352.

[25] A. George and J. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1981.

[26] J. Gilbert, *Predicting structures in sparse matrix computations*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 62–79.

[27] M. J. Grote and T. Huckle, *Parallel preconditioning with sparse approximate inverses*, SIAM J. Sci. Comput., 18 (1997), pp. 838–853.

[28] V. E. Henson and P. Vassilevski, *Element-free AMGe: General algorithms for computing interpolation weights in AMG*, SIAM J. Sci. Comput., 23 (2001), pp. 629–650.

[29] G. Karypis and V. Kumar, *A fast and high quality multilevel scheme for partitioning irregular graphs*, SIAM J. Sci. Comput., 20 (1998), pp. 359–392.

[30] L. Yu. Kolotilina and A. Yu. Yeremin, *Factorized sparse approximate inverse preconditionings.* I. *Theory*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 45–58.

[31] J. W. Ruge and K. Stuben, *Algebraic multigrid*, in Multigrid Methods, S. McCormick, ed., Frontiers Appl. Math. 3, SIAM, Philadelphia, 1987, pp. 73–130.

[32] Y. Saad, *ILUM: A multi-elimination ILU preconditioner for general sparse matrices*, SIAM J. Sci. Comput., 17 (1996), pp. 830–847.

[33] Y. Saad and B. Suchomel, *ARMS: An Alegraic Recursive Multilevel Solver for General Sparse Linear Systems*, technical report umsi-99-107, University of Minnesota, 1999.

[34] Y. Saad and J. Zhang, *BILUM: Block versions of multielimination and multilevel ILU preconditioner for general sparse linear systems*, SIAM J. Sci. Comput., 20 (1999), pp. 2103–2121.

[35] W. Sweldens, *The lifting scheme: A construction of second generation wavelets*, SIAM J. Math. Anal., 29 (1998), pp. 511–546.

[36] W.-P. Tang and W. L. Wan, *Sparse approximate inverse smoother for multigrid*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1236–1252.

[37] P. S. Vassilevski and J. Wang, *Stabilizing the hierarchical basis by approximate wavelets,* I: *Theory*, Numer. Linear Algebra Appl., 4 (1997), pp. 103–126.

[38] P. S. Vassilevski and J. Wang, *Stabilizing the hierarchical basis by approximate wavelets,* II: *Implementation and numerical results*, SIAM J. Sci. Comput., 20 (1998), pp. 490–514.

[39] J. Xu and L. Zikatanov, *A monotone finite element scheme for convection-diffusion equations*, Math. Comp., 68 (1999), pp. 1429–1446.

[40] H. Yserentant, *On the multilevel splitting of finite element spaces*, Numer. Math., 49 (1986), pp. 379–412.

SIAM J. SCI. COMPUT.
Vol. 23, No. 2, pp. 480–498

SOLVING COMPLEX-VALUED LINEAR SYSTEMS VIA EQUIVALENT REAL FORMULATIONS*

DAVID DAY† AND MICHAEL A. HEROUX†

Abstract. Most preconditioned iterative methods apply to both real- and complex-valued linear systems. At the same time, most iterative linear solver packages available today focus exclusively on real-valued systems or deal with complex-valued systems as an afterthought. By recasting the complex problem in a real formulation, a real-valued solver can be applied to the equivalent real system.

On one hand, real formulations have been dismissed due to their unfavorable spectral properties. On the other hand, using an equivalent preconditioned real formulation can be very effective. We give theoretical and experimental evidence that an equivalent real formulation is useful in a number of practical situations. Furthermore, we show how to use the advanced features of modern solver packages to formulate equivalent real preconditioners that are computationally efficient and mathematically identical to their complex counterparts.

The effectiveness of equivalent real formulations is demonstrated by solving ill-conditioned complex-valued linear systems for a variety of large scale applications. Moreover, the circumstances under which certain equivalent real formulations are competitive is more clearly delineated.

Key words. complex linear systems, iterative methods, sparse matrices, preconditioning

AMS subject classifications. 65F10, 65F50, 65N12, 65Y05, 65Y15

PII. S1064827500372262

1. Introduction.

We describe a simple yet effective extension of a real-valued preconditioned iterative solver package to solve complex-valued linear systems such as

$$Cw = d, \tag{1.1}$$

where C is an m-by-n known complex matrix, d is a known right-hand side, and w is unknown. Although most preconditioners and iterative methods apply to either complex-valued or real linear systems [4], most preconditioned iterative solver packages treat only real-valued systems. Packages that deal with complex-valued linear systems include QMRPACK [8], PETSc [3, 2, 1], and work done at CERFACS [6]. However, even in these cases, the breadth, depth, and performance capabilities of the complex-valued solvers are not as complete as the collective set of real-valued solvers. Because of these apparent deficiencies, we are compelled to consider using the vast body of real-valued solver packages to solve complex-valued systems. This work explains when and how to leverage the existing real-valued solver packages for use with complex-valued systems.

1.1. Potential equivalent real formulations.

We begin our study of equivalent real formulations by writing (1.1) in its real and imaginary terms:

$$(A + iB)(x + iy) = b + ic. \tag{1.2}$$

*Received by the editors May 18, 2000; accepted for publication (in revised form) September 20, 2000; published electronically July 10, 2001.

http://www.siam.org/journals/sisc/23-2/37226.html

†Applied Mathematics Department, Massively Parallel Computing Research Lab, Sandia National Laboratories, P.O. Box 5800, Albuquerque, NM 87185-1110 (dday@cs.sandia.gov, mheroux@cs.sandia.gov).

Table 1.1
Spectral properties of the K formulations. $\sigma(K)$ denotes the spectrum of K and $i = \sqrt{-1}$.

Matrix	Spectral properties
K_1	(i) If $\lambda \in \sigma(C)$, then $\lambda, \bar{\lambda} \in \sigma(K_1)$. (ii) If C is Hermitian (positive definite), then K_1 is symmetric (positive definite).
K_2	(i) If $\lambda \in \sigma(K_2)$, then $-\lambda, \bar{\lambda}, -\bar{\lambda} \in \sigma(K_2)$. (ii) If C is symmetric, then K_2 is symmetric.
K_3	(i) If $\lambda \in \sigma(K_3)$, then $-\lambda, \bar{\lambda}, -\bar{\lambda} \in \sigma(K_3)$. (ii) If C is symmetric, then K_3 is symmetric. (iii) $\sigma(K_3) = \sigma(K_2)$.
K_4	(i) If $\lambda \in \sigma(C)$, then $-i\lambda, i\bar{\lambda} \in \sigma(K_4)$. (ii) If C is Hermitian (positive definite), then K_4 is skew symmetric (with eigenvalues that have positive imaginary parts).

Equating the real and imaginary parts of the expanded equation, respectively, gives rise to four possible 2-by-2 block formulations, listed in (1.3)–(1.6). We call these K1 to K4, respectively.

K1 *formulation.*

$$\begin{pmatrix} A & -B \\ B & A \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} b \\ c \end{pmatrix}. \tag{1.3}$$

K2 *formulation.*

$$\begin{pmatrix} A & B \\ B & -A \end{pmatrix} \begin{pmatrix} x \\ -y \end{pmatrix} = \begin{pmatrix} b \\ c \end{pmatrix}. \tag{1.4}$$

K3 *formulation.*

$$\begin{pmatrix} B & A \\ A & -B \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} c \\ b \end{pmatrix}. \tag{1.5}$$

K4 *formulation.*

$$\begin{pmatrix} B & -A \\ A & B \end{pmatrix} \begin{pmatrix} x \\ -y \end{pmatrix} = \begin{pmatrix} c \\ b \end{pmatrix}. \tag{1.6}$$

For future reference, we denote the matrix associated with the K1 to K4 formulations by K_1 to K_4, respectively. For any pair of bases for R^{2m} and R^{2n}, there is a different equivalent real formulation of (1.1). K1 to K4 are representative of the many different formulations.

The convergence rate of an iterative method applied directly to an equivalent real formulation is often substantially worse than for the corresponding complex iterative method (see [7] and section 2). The slower convergence rate is due to the spectral properties of the equivalent real formulation. Table 1.1 summarizes the spectral properties of K_1 to K_4.

We digress briefly to justify certain relations in Table 1.1 that are not discussed in the following sections. The K_2 and K_3 matrices satisfy $K_2J = -JK_2$ and $K_3J = -JK_3$, where

$$J = \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix}. \tag{1.7}$$

In fact, $-JK_2$ has the same eigenvalues as K_2 and is similar to K_3. Thus $\sigma(K_3) = \sigma(K_2)$ is symmetric with respect to the origin. ($\sigma(K)$ denotes the spectrum of K.)

The configuration of the eigenvalues of K_2 and K_3 is problematic for Krylov methods. If a matrix has positive eigenvalues, then augmenting that matrix in a way that reflects the spectrum through the origin degrades the convergence rate of an iterative method such as the generalized minimum residual (GMRES) method. The convergence rate for a matrix whose spectrum is symmetric with respect to the origin is the square root of the rate for the same method applied to a linear system whose spectrum lies in a half plane (see section 3.2, Example 2). Similarly, convergence rates in the K2 or K3 formulations tend to be the square root of the rates in the K1 or K4 formulations.

For the K1 formulation, if all the eigenvalues of C are on one side of the imaginary axis, then the spectrum of K_1 should not present a major problem to an iterative method such as GMRES. However, if C has eigenvalues on both sides of the imaginary axis, a property that degrades the GMRES convergence rate, then K_1 will have twice as many problematic eigenvalues. Moreover, the convex hull containing the eigenvalues of K_1 will also contain the origin. For the K4 formulation, if all the eigenvalues of C are in the upper half plane, then the eigenvalues of K_4 will be in the right half plane.

It is noteworthy that the K4 formulation of (1.1) is the same as the K1 formulation of

$$i\,\overline{C}\overline{w} = i\,\overline{d}. \tag{1.8}$$

Similarly, the K3 formulation of (1.1) is the same as the K2 formulation of (1.8). Because of these relationships, we need to consider only the K1 and K2 formulations if we are interested in the convergence properties of *unpreconditioned* Krylov methods. In fact, most of the remaining discussion is focused on the K1 and K2 formulations with K3 and K4 being special cases. However, if we want to consider the preconditioned case, especially preconditioners such as the incomplete lower/upper factorization (ILU), then K4 deserves more attention than we present in section 2.2.

If C is Hermitian, then K_1 is symmetric, and, as we will demonstrate below, the convergence rate with an equivalent real formulation is identical to the convergence rate with the original complex formulation. If C is complex symmetric, then K_2 and K_3 are also complex symmetric. Nevertheless, we still recommend K_1 for complex symmetric linear systems (see section 3.2, particularly Examples 1 and 2) because of the problem mentioned above. The best way known to the authors to precondition K_2 or K_3 is to permute back to K_1.

In view of their spectral properties, the K1 to K4 formulations have been justly criticized. Particularly, the K2 and K3 formulations appear to be unusable. However, in spite of these properties, we have found that a variation of the K1 formulation has merit. Success with the K1 formulation depends on the quality of the preconditioner to the *same* extent that success with the original complex formulation depends on the quality of the complex preconditioner. In fact, our experience shows that for the classes of problems we are solving, in particular, eigenvalue problems for computational fluid dynamics using complex valued shift parameters, if a good preconditioner is used, then the iteration count of the K formulation (discussed below) is generally comparable (within 50%) to that of solving the original complex problem with a true complex preconditioned iterative solver and has the same robustness as a true complex solver. Given the wide availability of excellent real-valued solver packages, we view our results as noteworthy.

2. Preconditioning equivalent real systems.

In trying to solve the original complex system in (1.1) via the K1 formulation in (1.3), the most interesting question is how to precondition K_1. Standard real-valued preconditioners such as Jacobi, Gauss–Seidel, or ILU applied directly to K_1 are not robust enough for our needs. Furthermore, the ordering of the unknowns is unsuitable for sparse matrix operations related to factorization, particularly ILU preconditioning.

Another preconditioner we experimented with is

$$M = \begin{pmatrix} M_A & 0 \\ 0 & M_A \end{pmatrix}, \tag{2.1}$$

where M_A is a real-valued preconditioner determined from the real part of $C = A+iB$. This approach can be viewed as a special form of block Jacobi preconditioning that may be effective if terms in A are generally larger in magnitude than those in B. However, for the problems we tested, the preconditioner in (2.1) was also not robust enough. We observed large increases in iteration count and outright failure to converge, while the solution of the complex system via a complex solver succeeded. Preconditioning by the imaginary part of C (by using the K4 formulation in (1.6)) also failed.

2.1. The K formulation.

The approach that consistently gives us good results is based on preserving the sparsity pattern of C. A *block entry* matrix is a sparse matrix whose entries are all (small) dense (sub)matrices. An alternative formulation, which we call the K formulation, preserves the nonzero pattern of the block entries at the expense of doubling the size of each dense submatrix.

In the K formulation, $c_{pq} = a_{pq} + ib_{pq}$ corresponds, via the scalar K1 formulation, to the 2-by-2 block entry of the $2m$-by-$2n$ real matrix K given by

$$\begin{pmatrix} a_{pq} & -b_{pq} \\ b_{pq} & a_{pq} \end{pmatrix}. \tag{2.2}$$

For example, if

$$C = \begin{pmatrix} c_{11} & 0 & c_{13} & 0 & c_{15} \\ 0 & c_{22} & c_{23} & 0 & 0 \\ c_{31} & 0 & c_{33} & c_{34} & 0 \\ 0 & 0 & c_{43} & c_{44} & 0 \\ c_{51} & 0 & 0 & 0 & c_{55} \end{pmatrix}, \tag{2.3}$$

then

$$K = \left(\begin{array}{cc|cc|cc|cc|cc} a_{11} & -b_{11} & 0 & 0 & a_{13} & -b_{13} & 0 & 0 & a_{15} & -b_{15} \\ b_{11} & a_{11} & 0 & 0 & b_{13} & a_{13} & 0 & 0 & b_{15} & a_{15} \\ \hline 0 & 0 & a_{22} & -b_{22} & a_{23} & -b_{23} & 0 & 0 & 0 & 0 \\ 0 & 0 & b_{22} & a_{22} & b_{23} & a_{23} & 0 & 0 & 0 & 0 \\ \hline a_{31} & -b_{31} & 0 & 0 & a_{33} & -b_{33} & a_{34} & -b_{34} & 0 & 0 \\ b_{31} & a_{31} & 0 & 0 & b_{33} & a_{33} & a_{34} & -b_{34} & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & a_{43} & -b_{43} & a_{44} & -b_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & b_{43} & a_{43} & a_{44} & -b_{44} & 0 & 0 \\ \hline a_{51} & -b_{51} & 0 & 0 & 0 & 0 & 0 & 0 & a_{55} & -b_{55} \\ b_{51} & a_{51} & 0 & 0 & 0 & 0 & 0 & 0 & b_{55} & a_{55} \end{array}\right). \tag{2.4}$$

In a related real formulation, also preserving the block entry structure, the K2 formulation of c_{pq} forms the corresponding 2-by-2 block entry

$$\begin{pmatrix} b_{pq} & a_{pq} \\ a_{pq} & -b_{pq} \end{pmatrix}. \tag{2.5}$$

The K3 and K4 versions are generated similarly.

2.2. Implementation within existing software packages. The properties of the K formulation defined in section 2.1 enable us to implement efficient and robust preconditioned iterative solvers for complex linear systems. We can efficiently compute and apply the exact equivalent of a complex-valued preconditioner. If the complex preconditioned linear system has nice spectral properties, then the K formulation leads to convergence that is competitive with the true complex solver.

Solvers with block entry support. There are a number of general-purpose parallel sparse iterative solver packages that do not require special matrix properties, e.g., structured grids, and thus can be applied to a wide range of problems. These packages are usually freely available and have been used in a variety of applications. Several of these packages, including Aztec [19] and PETSc [2], support block entry matrices. The matrix K in the K formulation has a natural 2-by-2 block structure that can be exploited by using block entry data structures. Using the block entry features of these packages has the following benefits.

1. Applying 2-by-2 block Jacobi scaling to K corresponds exactly to applying point Jacobi scaling to C.
2. The block sparsity pattern of K exactly matches the point sparsity pattern of C. Thus any pattern-based preconditioners such as block ILU(l) applied to K correspond exactly to ILU(l) applied to C. See section 4 for definitions of block ILU(l) and ILU(l).
3. Any drop tolerance-based complex preconditioner has a straightforward K formulation since the absolute value of a complex entry equals the scaled Frobenius norm of the corresponding block entry in K.

Other solver packages. For existing solvers that do not have block entry support, it is still possible to exploit many of the benefits of the K formulation with the following considerations.

1. It is still possible to form the K matrix, even though its underlying block structure will not be explicitly available. Many preconditioners, especially incomplete factorizations, will closely approximate the block preconditioners listed above. Ideally, a real or imaginary nonzero in C corresponds to four entries in K, two of which are zero and may fill in during an incomplete factorization.
2. If any diagonal entries of the complex matrix C have zero or very small real parts, then pointwise incomplete factorization preconditioners may fail. (Note that this is not a problem for block entry factorization if pivoting is done within the diagonal block entries.) In this situation, one might consider using the K4 equivalent of the K formulation, or one may interleave the K1 and K4 formulations at the equation level depending on the relative magnitude of the real and imaginary parts of the complex diagonal entries. In certain situations, this leads to better ILU factors. However, we have not fully explored this approach, and its overall effectiveness remains an open question to us.

As noted above, by using block entry features, we can easily and efficiently construct preconditioners for K that are equivalent to those we would form for C using a true complex preconditioner formulation. As a result, the equivalent real preconditioned matrix operation is *identical* to the true complex preconditioned operator up to a permutation. Thus solving the real equivalent form via the K formulation using a preconditioned iterative method is identical to solving the original complex system using a corresponding preconditioned complex solver, except that the two approaches use different inner product spaces. Section 3 expands upon these comments.

3. Properties of the K formulation.

The K formulation of a preconditioned iterative method is comparable to the original complex formulation. Section 3.1 presents an equivalence between a preconditioned complex linear system and the corresponding K formulation. Next, the properties of K and C that influence convergence of iterative linear solvers are contrasted. The critical difference between K and C is that the eigenvalues of K are the eigenvalues of C together with their conjugates; $\sigma(K) = \sigma(C) \cup \overline{\sigma(C)}$. Section 3.2 contains a detailed survey of the influence of the spectrum of the convergence rate. The theoretical results of sections 3.1 and 3.2 are summarized in section 3.3 and illustrated using an example from computational chemistry in section 3.4.

3.1. Homomorphic properties of the K formulation.

The K formulation of a coefficient matrix C results in a symmetric permutation $K = PK_1P^T$ that preserves the block entry structure of C. Note that K and K_1 are orthogonally similar and share many properties.

We use the function $f()$ to denote the matrix K that corresponds to C in the K formulation, $f(C) = K$. The *key to understanding the K formulation is that f is a homomorphism*:

$$\begin{cases} f(I) = I, \\ f(XY) = f(X)f(Y). \end{cases}$$

This observation appears not to have been made before in this context. In the K formulation of an iterative method for solving a linear system with coefficient matrix C and preconditioner M, the linear system $f(C)$ is preconditioned with preconditioner $f(M)$. The K formulation of a preconditioned iterative method inherits from the preconditioned complex iterative method through

$$f(M^{-1}C) = f(M)^{-1}f(C)$$

all the properties that K_1 inherits from C.

We use the following framework to compare preconditioned iterative linear solvers. For clarity, a zero initial guess is assumed. A left-preconditioned Krylov solver is an algorithm that determines a sequence of approximations from the corresponding expanding Krylov subspaces

$$\mathcal{K}^k(M^{-1}C, M^{-1}d) = \text{span}(M^{-1}d, \ldots, (M^{-1}C)^{k-1}M^{-1}d). \tag{3.1}$$

Right preconditioning is similar. A preconditioned Krylov solver succeeds to the extent that the algorithm converges in a small number of iterations.

Characterizing successful preconditioners is difficult. The convergence rate of conjugate gradient methods applied to the left or right normal equations is bounded in terms of the square of the condition number

$$\text{cond}(M^{-1}C). \tag{3.2}$$

The convergence rate of GMRES is bounded in terms of the eigenvalues and condition number of the matrix of eigenvectors of $M^{-1}C$ [15]. For many problems, the maximum absolute value of the ratio of eigenvalues, the *spectral* condition number, best correlates with preconditioner effectiveness. The Krylov subspace, (3.1), the condition number of C, (3.2), and the condition number of the matrix of eigenvectors of C are all invariant under the K formulation (see below).

The K formulation preserves Krylov subspaces in the sense that

$$f((M^{-1}C)^k M^{-1}) = (f(M)^{-1}f(C))^k f(M)^{-1}$$

for left preconditioning and a similar equation applies with a right preconditioner. A prerequisite to discussing condition numbers is to relate the singular value decomposition (SVD) of C,

$$C = U\Sigma V^*,$$

to the SVD of K_1. Problem 8.6.4 in [9] is to show that if $U = U_r + iU_u$ and $V = V_r + iV_u$, where U_r,U_u,V_r,V_u are real, then K has the SVD

$$K = \begin{bmatrix} U_r & -U_u \\ U_u & U_r \end{bmatrix} \begin{bmatrix} \Sigma & \mathbf{0} \\ \mathbf{0} & \Sigma \end{bmatrix} \begin{bmatrix} V_r & -V_u \\ V_u & V_r \end{bmatrix}^T.$$

In particular, f preserves condition numbers. This implies the second property, that conditioning is preserved from the complex case:

$$\begin{aligned} \text{cond}(M^{-1}C) &= \text{cond}(f(M^{-1}C)) \\ &= \text{cond}(f(M)^{-1}f(C)). \end{aligned}$$

For clarity, we discuss eigenvalues in the unpreconditioned case. However, as above, these results to extend to the preconditioned case. As mentioned in section 3, eigenvalues of K are the eigenvalues of C together with their conjugates. Thus, for

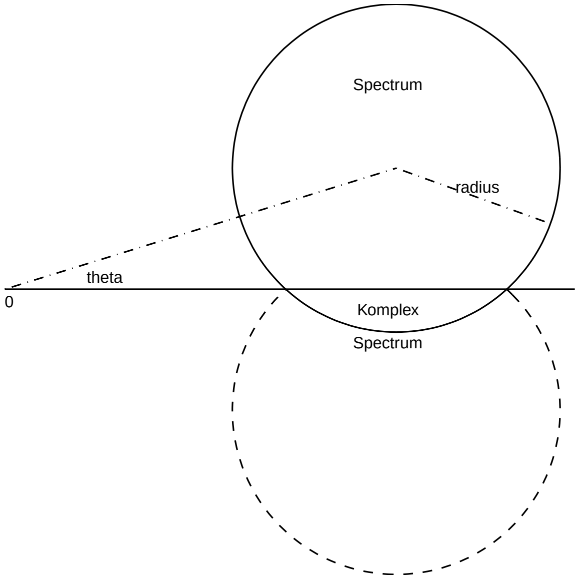


FIG. 3.1. *Asymmetric preconditioned spectrum.*

example, $M^{-1}C$ and $f(M)^{-1}f(C)$ also have the same *spectral* condition number. More precisely, Proposition 5.1 of [7] states that if C has Jordan normal form,

$$C = XJX^{-1},$$

then $K_1 = W\text{diag}(J, \overline{J})W^{-1}$ for

$$W = \left[\begin{array}{cc} X & \overline{X} \\ -iX & i\overline{X} \end{array} \right].$$

An observation not made explicitly in [7] is that it immediately follows that

$$\text{cond}(W) = \text{cond}(X).$$

As mentioned earlier, eigenvalue-based bounds on the convergence rate involve the condition number of the matrix of eigenvectors. (See also (3.3) below.)

3.2. Convergence of the K formulation. Augmenting the spectrum of C to $\sigma(K) = \sigma(C) \cup \overline{\sigma(C)}$ may or may not change the convergence rate of an iterative method. Examples 1–4 below compare the convergence rates of iterative methods in the K formulation and the original complex formulation applied to representative classes of model problems. Based on these examples, we conclude the following.

- For Hermitian linear systems, the K formulation preserves the convergence rate of the original complex formulation.
- For linear systems with an asymmetric preconditioned spectrum (see Figure 3.1), the convergence rate degrades mildly in a K formulation.

We review the prerequisite mathematical tools to keep this work self-contained.

Preliminary remarks. We selected four examples to illustrate the influence of recasting a complex linear system in an equivalent real form on the convergence rate of the linear solver. Examples 1 and 3 are paraphrased from [5]. Example 2 is classical and reviewed here in detail. Example 4 appears to be new. Examples 1 and 2 illustrate the best and the worst properties of the K1 formulation. In Example 1 the spectral condition number is preserved, and in Example 2 the spectral condition number is squared. Fortunately, the K formulation of a successfully preconditioned system is more like Example 1 than Example 2.

Examples 3 and 4 relate the convergence rates of the K formulation of a preconditioned iterative method to the corresponding complex preconditioned iterative method if the spectrum is nearly symmetric with respect to the real axis. Successful preconditioning transforms the spectrum into a disk far from 0. Example 3 shows that the convergence rate for the complex formulation is the ratio of the radius of the disk to the distance from the center of the disk to the origin. In the K formulation the convergence rate also depends on the angle between the disk and the real axis. Example 4 shows how the convergence rate gently increases as the disk rotates away from the positive real axis.

The asymptotic convergence factor for polynomial-based methods, including preconditioned Krylov subspace methods, is κ if the nth residual norm is proportional to κ^n for some constant κ. A sharp upper bound for the asymptotic convergence factor can be determined using the complex Green's function for the convex hull of the spectrum. Following [5, pp. 90–93], an iterative method for $Cw = d$ determines $\{w_k\}$ that (hopefully) converges to w and residuals, $r_k = d - Cw_k$, such that $\{r_k\}$ converges to zero. Consider the polynomial-based iterative solution method

$$r_n = p_n(C)r_0, \quad p_n(0) = 1,$$

where each $p_n \in \Pi_n$, the space of nth degree polynomials. Next let Ω be a set containing $\sigma(C)$, and define

$$\|p_n\|_\Omega = \max_{\omega\in\Omega} |p_n(\omega)|.$$

Common choices for Ω are the convex hull of $\sigma(C)$, a disk, or an ellipse with a major axis along a ray through the origin. For clarity, we assume that C is diagonalizable: $CV = V\Lambda$. The reduction in the residual norm

$$\|r_n\|_2 = \|p_n(C)r_0\|_2 \leq \text{cond}(V)\|p_n\|_{\sigma(C)}\|r_0\|_2 \leq \text{cond}(V)\|p_n\|_\Omega\|r_0\|_2 \tag{3.3}$$

is bounded above by the spectral condition number of V and $\|p_n\|_\Omega$. A residual polynomial p_n that minimizes $\|p_n\|_\Omega$ is an *optimal polynomial* $\mathcal{P}_n(t;\Omega,0)$ and solves

$$\|\mathcal{P}_n(t;\Omega,0)\|_\Omega = \min\{\|p\|_\Omega : p \in \Pi_n, p(0) = 1\}, \quad 0 \notin \Omega.$$

The convergence of an iteration $r_n = p_n(C)r_0$ is related to the asymptotic convergence factor for the polynomial iterative method induced by $\{p_n\}$,

$$\kappa(C;p_n) = \limsup_{n\to\infty}\left(\sup_{r_0\neq 0}\frac{\|r_n\|_2}{\|r_0\|_2}\right)^{1/n}.$$

The asymptotic convergence factor for Ω is defined by

$$\kappa(\Omega) = \inf_{p_n}\sup_{\sigma(C)\subset\Omega}\kappa(C;p_n).$$

The asymptotic convergence factor corresponding to the K formulation is $\kappa(\Omega\cup\overline{\Omega})$.

Next we determine the asymptotic convergence factor for C and K in several characteristic cases. $\kappa(\Omega)$ is determined from Green's function $G(z;\Omega^c)$ for the complement Ω^c of Ω with pole at infinity

$$\kappa(\Omega) = |G(0;\Omega^c)|.$$

If Ω is connected, then G is a conformal mapping from Ω^c to the open unit disk such that $G(\infty) = 0$. In general, G has the following properties.

- $G(z;\Omega^c)$ is an analytic function on Ω^c with a single-valued modulus $|G(z;\Omega^c)| < 1$ in Ω^c.
- $G(z;\Omega^c)$ has precisely one zero at ∞.
- If $z \in \partial\Omega$, then $|G(z;\Omega^c)| = 1$.

Example 1. If C is Hermitian positive definite, then the convex hull of $\sigma(C) = \sigma(K)$ is $\Omega = [\alpha, \beta]$ for $0 < \alpha < \beta$. The first step is to derive Green's function for the complement of the interval $G(z; I^c)$, an inverse of $\phi(z) = (z + z^{-1})/2$, where I denotes the interval $[-1, 1]$. We select the square root function with a branch cut along the negative real axis and

$$\phi^{-1}(z) = \begin{cases} z + \sqrt{z^2 - 1}, & 0 \le \arg(z) \le \pi, \\ z - \sqrt{z^2 - 1}, & \pi < \arg(z) < 2\pi. \end{cases}$$

Note that the branch cut for the argument function is along the positive real axis, and for $\rho > 1$,

$$\lim_{z \to \rho, \Im(z) < 0} \arg(z^2 - 1) = 2\pi.$$

Here $\Im(z)$ denotes the imaginary part of the complex number z. In this case, the singularity in ϕ^{-1} along $[1, \infty]$ has been removed, and because ϕ^{-1} is odd, ϕ^{-1} is analytic on I^c. To show that ϕ^{-1} maps I^c to the open unit disk, note that the equation $\phi(z) = w$ is a quadratic polynomial in z and $\phi(z) = \phi(z^{-1})$.

Now for $\Omega = [\alpha, \beta]$ such that $0 < \alpha < \beta$, $G(z;\Omega^c) = \phi^{-1}(\ell(z))$ for $\ell(t) = (2t - \alpha - \beta)/(\beta - \alpha)$, and

$$\kappa([\alpha, \beta]) = |G(0)| = \frac{1}{-\ell(0) + \sqrt{\ell^2(0) - 1}} = \frac{\beta - \alpha}{\beta + 2\sqrt{\beta\alpha} + \alpha}.$$

Example 2. If iC is Hermitian positive definite, then the convex hull of $\sigma(C)$ is $[i\alpha, i\beta]$ for $0 < \alpha < \beta$ and $\kappa(i[\alpha, \beta]) = \kappa([\alpha, \beta])$. However, $\sigma(K) \subset [-i\beta, -i\alpha] \cup [i\alpha, i\beta]$, and $\kappa([-i\beta, -i\alpha] \cup [i\alpha, i\beta]) = \kappa([-\beta, -\alpha] \cup [\alpha, \beta])$. The first step is to derive Green's function for the complement of symmetric intervals $\Omega = [-1, -\eta] \cup [\eta, 1]$ for $\eta = \alpha/\beta < 1$. Green's function is a branch of the solution of

$$\psi^2 + 2\frac{2z^2 - (1 + \eta^2)}{1 - \eta^2}\psi + 1 = 0.$$

The two branches

$$\psi_\pm = \frac{2}{1 - \eta^2}\left(\pm\sqrt{(z^2 - 1)(z^2 - \eta^2)} - z^2 + \frac{1 + \eta^2}{2}\right)$$

satisfy $\psi_+\psi_- = 1$, and we seek a branch whose range is the open unit disk. We choose the branch of $\pm\sqrt{(z^2 - 1)(z^2 - \eta^2)}$ that is nearest to z^2. The branch cut for $\sqrt{}$ is along the negative real axis, and

$$\arg\left((z^2 - 1)(z^2 - \eta^2)\right) = \pm\pi \leftrightarrow z = x + iy \;\text{ and }\; x^2 - y^2 = \frac{1 + \eta^2}{2}.$$

This gives us Green's function

$$G(z) = \begin{cases} \frac{2}{1-\eta^2}\left(\sqrt{(z^2-1)(z^2-\eta^2)} - z^2 + \frac{1+\eta^2}{2}\right), & x^2 - y^2 \geq \frac{1+\eta^2}{2}, \\ \frac{2}{1-\eta^2}\left(-\sqrt{(z^2-1)(z^2-\eta^2)} - z^2 + \frac{1+\eta^2}{2}\right), & x^2 - y^2 < \frac{1+\eta^2}{2}. \end{cases}$$

To show that the range of G is the open unit disk, note that ψ is a quadratic polynomial and $|\psi_\pm| = 1$ if and only if $\eta^2 \leq z^2 \leq 1$. In this case, $\kappa(\Omega) = (1-\eta)/(1+\eta) = (\beta-\alpha)/(\beta+\alpha)$.

To contrast Examples 1 and 2 as in the preliminary remarks, note that β/α is the spectral condition number of C. In Example 1 the asymptotic reduction factor depends on the square root of the spectral condition number,

$$\kappa_1 \approx 1 - 2\sqrt{\frac{\alpha}{\beta}}.$$

In Example 2 the asymptotic reduction factor depends on the spectral condition number and is much larger:

$$\kappa_2 \approx 1 - 2\,\frac{\alpha}{\beta} \gg \kappa_1.$$

Examples 3 and 4 quantify the penalty for using the K formulation instead of the true complex formulation if the convex hull of the preconditioned spectrum lies inside a disk in the left half plane rotated by θ from the positive real axis. Our analysis applies in the case in which the disk intersects its conjugate. Example 3 shows that in complex arithmetic, the asymptotic convergence factor is independent of θ. Comparing (3.4) with $|w| = 1$ to (3.6) shows that, in the K formulation, the asymptotic convergence factor increases mildly with θ.

Example 3. Here $\Omega = \{z : |z - w| < r\}$ is the disk in the complex plane of radius r about a point w that is the distance $\rho = |w|$ from the origin. The asymptotic convergence factor is $\eta = r/\rho$. A conformal mapping of Ω onto the open unit disk is $G(z;\Omega) = \frac{r}{z-w}$ and

$$\kappa(\Omega) = \frac{r}{|w|} = \eta. \tag{3.4}$$

Example 4. Here Ω comes from rotating the disk $\{z : |z - \rho| < r\}$ with center $\rho > 0$ by θ and then from reflecting though the real axis as illustrated in Figure 3.1. The circles intersect: $\sin\theta < \eta = r/\rho$.

Figure 3.2 illustrates the conformal mapping from two intersecting disks to a disk. The upper left graph corresponds to Figure 3.1. In the upper right figure the two disks are transformed to a wedge by the fractional linear transformation $w(z) = (z-\mu)/(z-\nu)$ for μ and ν defined in (3.5). Next, in the lower left figure, the wedge is opened up into the imaginary axis via $v(w) = w^{\pi/\alpha}$ for α defined in (3.5). In the lower right figure, the half plane is transformed to a disk by another fractional linear transformation $u(v) = (v-1)/(v+1)$.

To sum up, the conformal mapping of the exterior of two intersecting circles to the open unit disk is given by

$$G(z) = \frac{\left(\frac{z-\mu}{z-\nu}\right)^{\pi/\alpha} - 1}{\left(\frac{z-\mu}{z-\nu}\right)^{\pi/\alpha} + 1},$$

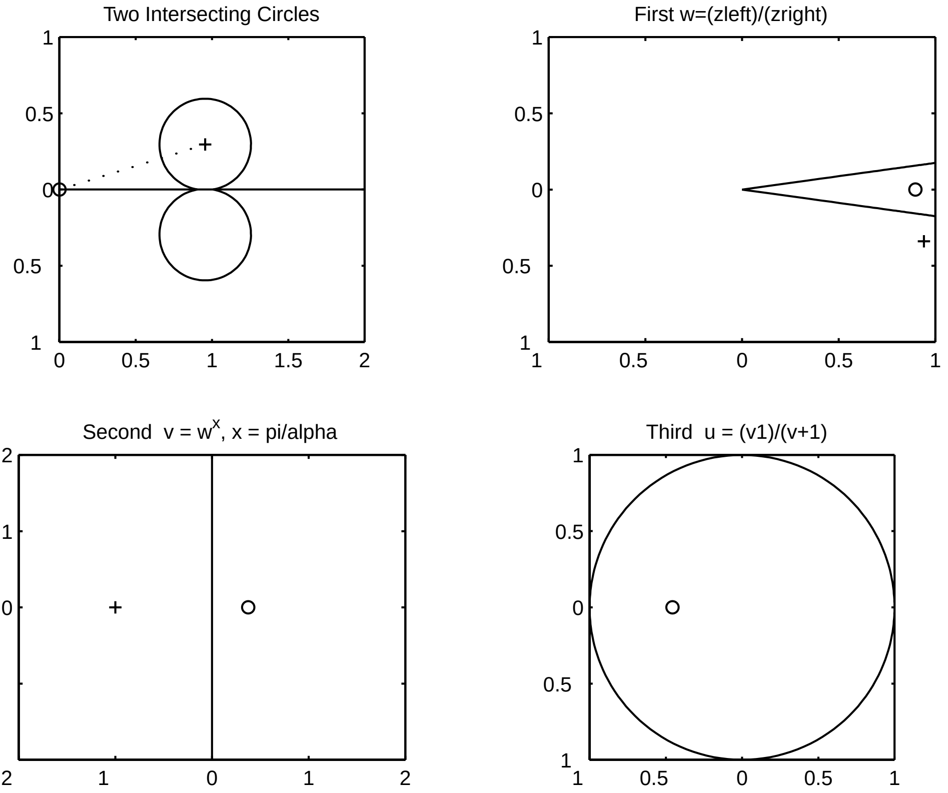


Fig. 3.2. *A conformal mapping between two intersecting disks (upper left) and a disk (lower right).*

where

$$\mu = \cos\theta - \sqrt{\eta^2 - \sin^2\theta}, \quad \nu = \cos\theta + \sqrt{\eta^2 - \sin^2\theta}, \quad \alpha = \pi - 2\tan^{-1}\left(\frac{\sin\theta}{\sqrt{\eta^2 - \sin^2\theta}}\right). \tag{3.5}$$

Also $w^{\pi/\alpha} = \exp(\frac{\pi}{\alpha}\log w)$ for $\log w = \log|w| + i\arg w$, and the branch for the argument function is placed along the negative real axis, $-i\pi < \arg w \leq i\pi$. In this case,

$$\kappa(\Omega) = (1 - (\mu/\nu)^{\pi/\alpha})/(1 + (\mu/\nu)^{\pi/\alpha}) = \eta + \frac{|\sin\theta|}{\eta\pi}(1-\eta)^2 + O(\theta^2). \tag{3.6}$$

3.3. Summary of K formulation properties. Based on the results of this section, we see that the K formulation differs from a true complex iterative solver only in the inner product space used by the iterative method. In other words, by utilizing the block entry data structures mentioned in section 2.2, we are able to provide the identical preconditioned matrix-multiply computations using the K formulation as we would for a true complex solver.

Furthermore, for complex Hermitian problems, there is no difference in asymptotic convergence rates. In fact, as is well known (e.g., see Problem 8.3.6 in [9]), if a complex matrix C is Hermitian (positive definite), the corresponding K matrix has the same eigenvalues, each with doubled multiplicity. Given the ability of the conjugate gradient

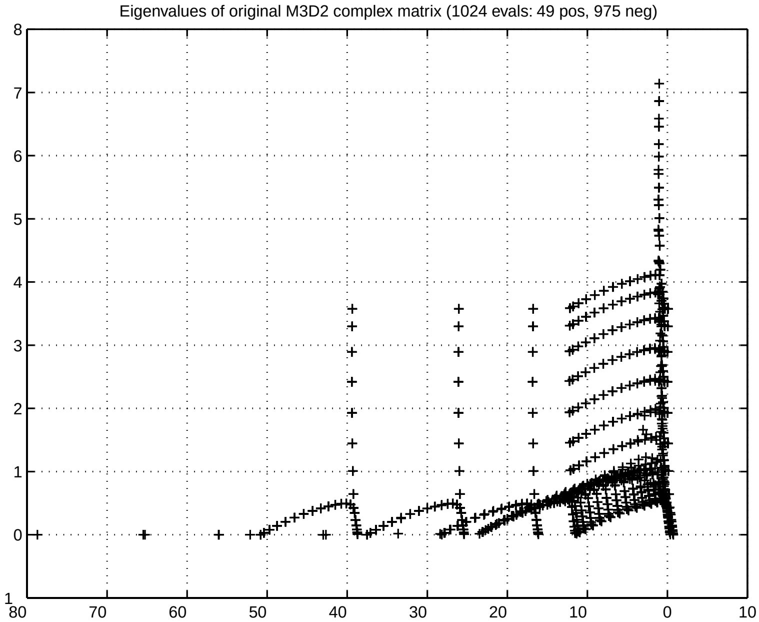


FIG. 3.3. *Eigenvalues of the original complex matrix in problem $M3D2$.*

method for linear systems to resolve multiple eigenvalues simultaneously, we observe in practice that the K formulation has identical convergence properties as a true complex solver for complex Hermitian problems.

For the non-Hermitian case, we saw from Examples 3 and 4 above that if the disk enclosing the spectrum of the preconditioned matrix C is not far from the point (1,0) in the complex plane, then the asymptotic convergence rate of the K formulation is close to the convergence rate of a true complex solver with the rate dictated by the size of the angle θ in Figure 3.1. As we will see in section 3.4, a high quality preconditioner tends to move the spectrum of C toward (1,0), setting up very favorable conditions for the K formulation.

3.4. Spectral case study. For problem M3D2 listed in Table 4.1, we computed the spectrum of the original and preconditioned matrix using the `eig` function of MATLAB. Figure 3.3 shows the distribution of the eigenvalues of the original matrix.[1] Figure 3.4 shows the eigenvalues of the K matrix, and, as expected, the eigenvalues of the K matrix are the eigenvalues of the complex matrix plus their reflection about the real axis.

Figure 3.5 shows the spectrum of the preconditioned matrix using `luinc(A,1e-1)` from MATLAB. `luinc` computes an incomplete LU factorization of a given sparse matrix. It provides several means to reduce the fill that would occur with an exact factorization. We chose to specify a drop tolerance. Given this drop tolerance, `luinc` will perform the LU factorization column by column as though it were doing an exact factorization, but as each column is computed, the terms in the column that are smaller in magnitude than the drop tolerance times the norm of the column are set to

[1] A note of thanks to Tom Wright and Nick Trefethen. They analyzed the pseudospectra of this matrix and determined that the eigenvalues of this matrix obtained via `eig` would be accurately computed.

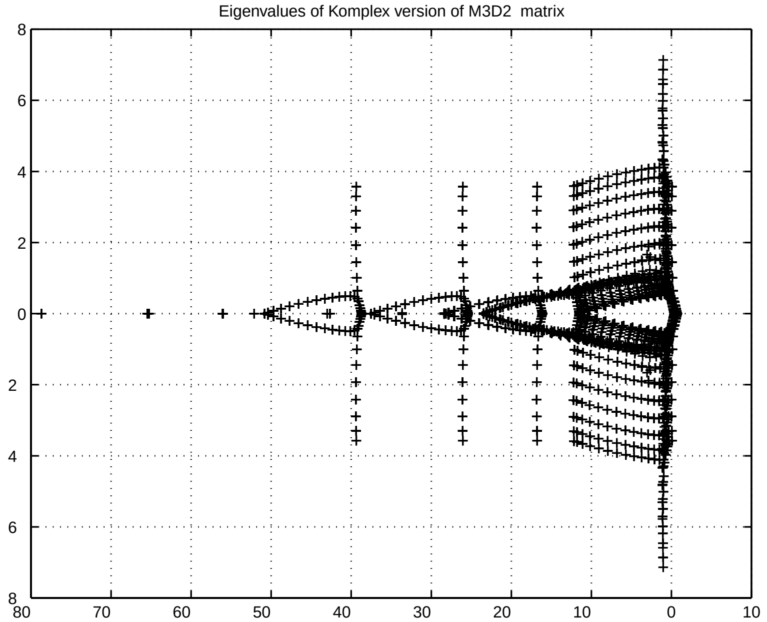


FIG. 3.4. *Eigenvalues of the K formulation matrix in problem M3D2.*

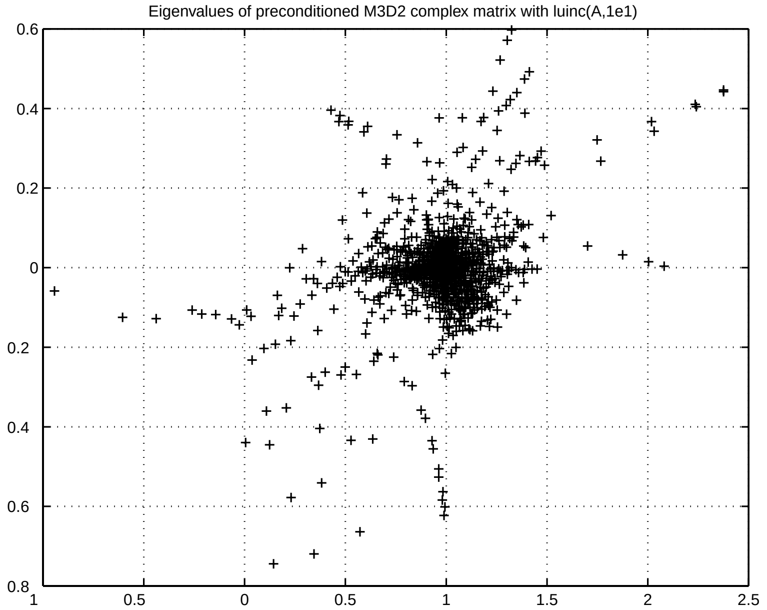


FIG. 3.5. *Eigenvalues of the complex matrix in problem M3D2, preconditioned by* `luinc(A,1e-1)`.

zero. `luinc(A,1e-1)` uses a drop tolerance of 10^{-1}. Note that the eigenvalues start to cluster around the point (1,0) in the complex plane.

Figure 3.6 shows the spectrum using `luinc(A,1e-2)`, which uses a drop tolerance of 10^{-2}. This preconditioner will typically keep more entries than `luinc(A,1e-1)` and

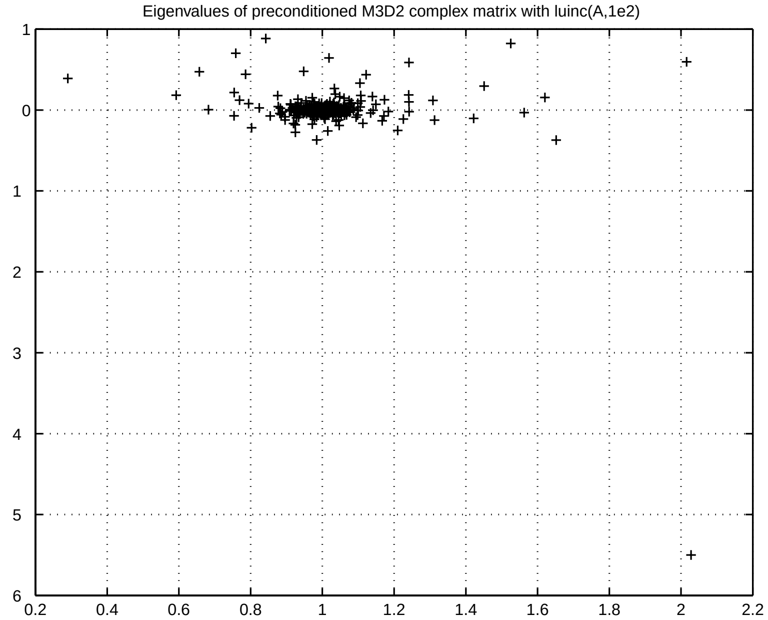


FIG. 3.6. *Eigenvalues of the complex matrix in problem M3D2, preconditioned by* `luinc(A,1e-2)`.

will be a better approximation to the exact LU factorization. With the exception of one outlyer, the eigenvalues in this case are closely clustered around (1,0).

Coupling this observation with the analysis from section 3.2, we see that high quality preconditioning, which tends to cluster the eigenvalues around the point (1,0), additionally minimizes the differences in asymptotic convergence rates between the true complex formulation and the K formulation. Thus, it simultaneously improves the convergence of both formulations and reduces the differences in convergence rates between them.

4. Computational results.

We have used the K formulation to solve complex linear systems coming from two application areas (see Table 4.1). Each system comes from a real application. These problems are very ill conditioned in the sense that inexpensive preconditioners like Jacobi or block Jacobi are not sufficient for convergence. Eigenvalue problems with complex eigenvalues are important applications for complex linear solvers. Complex linear systems arise first in computing the eigenvalue using a complex shift-invert implicitly restarted Arnoldi method [11] and next in tracking the eigenvalues as a function of a parameter using Newton's method. Selecting a shift near to the eigenvalues of interest to accelerate the convergence of Arnoldi's method creates a linear system with a large spectral condition number. We have found the K formulation with ILU preconditioning to be very effective.

4.1. Overview of problems and solution methods.

Our computational problems come from molecular dynamics and fluid dynamics. The first two problems listed in Table 4.1 come from given data sets where we are unfamiliar with the details of the applications. The last two problems come from efforts to understand the stability of a computed solution to a particular computational fluid dynamics problem using MPSalsa [18, 17]. The stability problems are representative of the linear

Table 4.1
Test problem descriptions.

Problem	Dim	# Nonzeros	Description
M3D2	1024	12480	Computational Chemistry Model I, Sherry Li, LBL/NERSC
M4D2	10000	127400	Computational Chemistry Model II, Sherry Li, LBL/NERSC
LINSTAB1	10590	276979	MPSalsa Linear Stability Analysis, Andrew Salinger, Rich Lehoucq, Cayley Transform Approach
LINSTAB2	10590	276979	MPSalsa Linear Stability Analysis, Andrew Salinger, Shift-and-Invert with shift equal to $33i$

systems solved in a complex shift-invert formulation of the eigenvalue problem.

MATLAB results were obtained using version 5.3.1 [12]. In particular, we used the built-in functions `luinc`, described in section 3.4, and `gmres`. `gmres` solves a linear system using the GMRES method [16] preconditioned by the ILU preconditioner computed by `luinc`.

The remaining results come from the Komplex Solver Package [10], an add-on module to Aztec 2.1 [19] that forms equivalent real systems from complex systems, uses Aztec to solve the real systems, and returns the complex results. For each of the problems we use overlapping domain decomposition preconditioning, specifically, additive Schwarz with one level of overlap, and we use GMRES(∞). For the M4D2 problem, the matrix has a natural block structure, and we used a block ILU preconditioner with a level fill of l (BILU(l)) on the subdomains, where we view the matrix as a sparse matrix whose entries are dense 8-by-8 matrices. The level fill parameter l refers to the pattern of the BILU factors. A level fill of $l = 0$ means that the only block entries kept are those that correspond to the pattern of the original matrix. A level fill of $l > 0$ is defined recursively to allow the BILU factors to have entries that correspond to the terms from the level fill $l - 1$ and any fill-in that comes directly from these terms. For LINSTAB1 and LINSTAB2 we replace BILU(l) with a variation of ILUT [14], a dual-threshold ILU preconditioner, as the subdomain preconditioner. The ILUT factors are computed row by row, dropping terms that fall below the threshold parameter multiplied by the norm of the row, and then keeping (at most) a prescribed number of the largest terms in the row. Details of the solvers and preconditioners can be found in [19].

Table 4.2
MATLAB test results using GMRES(∞) with luinc(droptol) preconditioning.

Problem	droptol	*nz(ILU)/nz(A)*	$\|r\|/\|b\|$	*C Iters*	*K Iters*
M3D2	1×10^{-3}	5.8	3×10^{-11}	12	12
	1×10^{-2}	4.5	8×10^{-11}	30	40
	1×10^{-1}	0.5	5×10^{-11}	107	181
M4D2	1×10^{-4}	13.1	5×10^{-11}	17	23
	1×10^{-3}	6.7	6×10^{-11}	72	109
LINSTAB1	1×10^{-3}	10.7	9×10^{-11}	71	93

4.2. Results. The first set of results (in Table 4.2) were obtained using a MATLAB code where we compare a true complex preconditioned iterative solver to the K formulation. For these problems, the preconditioned operators are exactly equivalent; i.e., using the notation from section 3, if M_C and M_K are the C and K preconditioners, respectively, then $M_K = f(M_C)$. Thus what we are comparing are the differences due to having a complex inner product over n-space versus a real inner product over $2n$-space. Note that, as the quality of the preconditioner improves, the difference in iteration counts between the two approaches diminishes.

Our MATLAB results did not have any relevant solution time statistics, so we cannot precisely measure the relative costs. However, the results in Table 4.3 show that higher quality preconditioners also provide the best time to solution, up to a point where the time spent constructing and applying the higher quality preconditioner exceeds the reduction in time due to fewer iterations. These results are from the Komplex Solver Package [10], an add-on module to Aztec. We used a BILU preconditioner with 8-by-8 blocks and GMRES(∞). The best time to solution comes from BILU with a level fill of 2. This result suggests a general observation that a high quality preconditioner provides both the best time to solution and makes the difference in iteration counts between the true complex and the K formulations minimal.

Table 4.3
Komplex test results for M4D2 using GMRES(∞) with block ILU preconditioning, 8-by-8 blocks.

Problem	l	$\|r\|/\|b\|$	*K Iters*	Time(s)
M4D2	0	1×10^{-11}	322	354
	1	1×10^{-11}	179	182
	2	1×10^{-11}	75	95
	3	1×10^{-11}	60	125

The final set of results (in Table 4.4) comes from using the Komplex Solver Package to solve linear stability problems in computational fluid dynamics. The primary purpose of these results is to illustrate that, with a modest amount of new software development, we are able to provide a general-purpose parallel preconditioned iterative solver for complex-valued linear systems by leveraging existing real-valued solvers. Results are given for 1, 2, 4, 8, and 16 processors of an 8-node, 16-processor PC-based Beowulf [13] cluster. The second column of this table lists the ratio of nonzero entries in the ILU factors to the original matrix. In all cases we iterated until the scaled residual (the norm of the residual divided by the norm of the right-hand side) was below 10^{-13}. The increase in the ILU fill is due to the effect of overlap used by our parallel overlapping Schwarz preconditioners. Another trend we see is that the iterations increase as we add processors, an additional effect that is common with these preconditioners. However, even with these increasing serial costs, the parallel timings are quite reasonable.

Table 4.4
Komplex test results using GMRES(∞) with ILUT preconditioning.

Problem	*nz(ILU)/nz(A)*	$\|r\|/\|b\|$	# Proc	*K Iters*	Time(s)
LINSTAB1	2.0	1×10^{-13}	1	27	151.4
	2.2		2	38	75.6
	2.4		4	55	42.6
	2.7		8	64	22.7
	2.8		16	79	13.3
LINSTAB2	2.0	1×10^{-13}	1	49	158.7
	2.2		2	57	80.8
	2.4		4	83	45.8
	2.7		8	93	26.3
	2.8		16	112	16.7

5. Conclusions.

In this paper we presented a discussion of how to solve complex-valued linear systems via equivalent real formulations. We listed approaches that failed and presented the K formulation, which works very well. Although it is clear from our results that the K formulation is not superior to a true complex solver, and clearly if you have easy access to a complex-valued solver you should use it, we do think that equivalent real formulations should receive more attention than they have in the past.

For many challenging problems, a high quality preconditioner is a requirement for convergence. Such a preconditioner has the tendency to map the spectrum around the point (1,0) in the complex plane. This, in turn, as the analysis in section 3.2 shows, minimizes the spectral difference between a true complex-valued iterative solver and the K formulation and leads to our observation that the requirement of a high quality preconditioner simultaneously provides the best solution times and diminishes the convergence differences between a true complex iterative solver and the K formulation.

Finally, for applications such as linear stability analysis, where all operators are real-valued except for the presence of complex shift value, the equivalent real formulation can be very attractive since it utilizes the native real-valued solver, and installation of the K formulation usually involves a modest amount of extra effort on the part of the application developer.

REFERENCES

[1] S. Balay, W. Gropp, L. McInnes, and B. Smith, *Efficient management of parallelism in object oriented numerical software libraries*, in Modern Software Tools for Scientific Computing, E. Arge, A. M. Bruaset, and H. P. Langtangen, eds., Birkhäuser Boston, Boston, 1997, pp. 163–202.

[2] S. Balay, W. Gropp, L. McInnes, and B. Smith, *PETSc* 2.0 *Users Manual*, Tech. report ANL-95/11—Revision 2.0.22, Argonne National Laboratory, Argonne, IL, 1998.

[3] S. Balay, W. Gropp, L. McInnes, and B. Smith, *PETSc home page*, http://www.mcs.anl.gov/petsc, 1998.

[4] R. Barrett, M. W. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, Philadelphia, 1994.

[5] B. Fischer, *Polynomial Based Iteration Methods for Symmetric Linear Systems*, 1st ed., Wiley, Chichester, UK, Teubner, Stuttgart, 1996.

[6] V. Fraysee, L. Giraud, and S. Gratton, *A Set of GMRES Routines for Real and Complex Arithmetics*, Tech. report TR/PA/97/49, CERFACS, Toulouse, France, 1997.

[7] R. W. Freund, *Conjugate gradient-type methods for linear systems with complex symmetric coefficient matrices*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 425–448.

[8] R. W. Freund and N. Nachtigal, *QMRPACK: A package of QMR algorithms*, ACM Trans. Math. Software, 22 (1996), pp. 46–77.

[9] G. Golub and C. Van Loan, *Matrix Computations*, 3rd ed., The Johns Hopkins University Press, Baltimore, MD, 1996.

[10] M. A. Heroux, *The Komplex Solver Package Reference Manual* 1.0, Sandia National Laboratories, Albuquerque, NM, 2000.

[11] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK Users' Guide*, SIAM, Philadelphia, 1998.

[12] MathWorks, *MATLAB home page*, http://www.mathworks.com, 2000.

[13] P. Merkey, *Beowulf home page*, http://beowulf.gsfc.nasa.gov, 1999.

[14] Y. Saad, *ILUT: A dual threshold incomplete LU factorization*, Numer. Linear Algebra Appl., 1 (1994), pp. 387–402.

[15] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 1st ed., PWS Publishing Company, Boston, 1996.

[16] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[17] A. Salinger, K. Devine, G. Hennigan, H. Moffat, S. Hutchinson, and J. Shadid, *MP-Salsa: A Finite Element Computer Program for Reacting Flow Problems Part* 2*—User's Guide*, Tech. report SAND96–2331, Sandia National Laboratories, Albuquerque, NM, 1996.

[18] J. N. Shadid, H. K. Moffat, S. A. Hutchinson, G. L. Hennigan, K. D. Devine, and A. G. Salinger, *MPSalsa: A Finite Element Computer Program for Reacting Flow Problems Part* 1*—Theoretical Development*, Tech. report SAND95–2752, Sandia National Laboratories, Albuquerque, NM, 1995.

[19] R. S. Tuminaro, M. A. H. S. A. Hutchinson, and J. N. Shadid, *Official Aztec User's Guide, Version* 2.1, Sandia National Laboratories, Albuquerque, NM, 1999.

SIAM J. Sci. Comput.
Vol. 23, No. 2, pp. 499–516

PRECONDITIONING STRATEGIES FOR FULLY IMPLICIT RADIATION DIFFUSION WITH MATERIAL-ENERGY TRANSFER*

PETER N. BROWN† and CAROL S. WOODWARD†

Abstract. In this paper, we present a comparison of four preconditioning strategies for Jacobian systems arising in the fully implicit solution of radiation diffusion coupled with material energy transfer. The four preconditioning methods are block Jacobi, Schur complement, and operator splitting approaches that split the preconditioner solve into two steps. One splitting method includes the coupling of the radiation and material fields that appears in the matrix diagonal in the first solve, and the other method puts this coupling into the second solve. All preconditioning approaches use multigrid methods to invert blocks of the matrix formed from the diffusion operator. The Schur complement approach is clearly seen to be the most effective for a large range of weightings between the diffusion and energy coupling terms. In addition, tabulated opacity studies were conducted where, again, the Schur preconditioner performed well. Last, a parallel scaling study was done showing algorithmic scalability of the Schur preconditioner.

Key words. preconditioning, Newton–Krylov, operator splitting, nonlinear solvers, radiation diffusion

AMS subject classifications. 65F10, 65N40, 65N55, 65H10

PII. S106482750037295X

1. Introduction.

In this paper, we present a comparison of preconditioners for a new numerical approach to the solution of very large-scale radiation diffusion problems. In this model, energy can be transferred to a material through coupling terms in both the radiation and material energy equations. These problems are important in modeling photon energy progression through an optically thick regime, a situation common in some laser and stellar fusion applications. Traditionally, solutions for these problems have been developed using operator split and time-lag techniques to reduce the coupled system of nonlinear equations to the solution of a series of linear problems. These solution techniques, however, lead to requirements of unacceptably small time steps. Furthermore, as computers have become faster, researchers have attempted to simulate larger problems despite existing solution methods that did not scale well for increased numbers of unknowns.

For these reasons, we have developed a solution method for solving radiation diffusion problems formulated in a fully implicit manner [5]. The fully implicit formulation allows larger time steps to be taken without sacrificing accuracy. Furthermore, recent work in iterative methods has provided computational scientists with new tools for solving these problems—tools that scale well to large numbers of unknowns. In order to solve this fully implicit formulation, we employ ODE time integration techniques which then require an implicit solve for the solution at each time step. We use an inexact Newton method for these solves, with a preconditioned Krylov method for solving the linear Jacobian systems that arise within the Newton iterations. The

*Received by the editors June 5, 2000; accepted for publication (in revised form) December 11, 2000; published electronically July 10, 2001. This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract W-7405-Eng-48. The U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes. Copyright is owned by SIAM to the extent not limited by these rights.

http://www.siam.org/journals/sisc/23-2/37295.html

†Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, CA 94551 (pnbrown@llnl.gov, cswoodward@llnl.gov).

Newton method provides fast nonlinear convergence, and the Krylov method gives a robust linear solver.

We consider four methods for solving the Jacobian preconditioning step in the Krylov method. All four schemes neglect the nonlinearity in the diffusion coefficient. The four methods primarily differ in how they approximate the coupling between the radiation and material energies. The first scheme is just to use the block diagonal part of the Jacobian matrix, thereby neglecting the majority of the coupling between the two fields. The second scheme is to factor the Jacobian and use a Schur complement preconditioner. The remaining methods use operator splitting approaches to split the preconditioner solve into two steps. One method includes the coupling of the radiation and material fields that appears in the matrix diagonal in the first solve, and the other method puts this coupling into the second solve. All preconditioning approaches use multigrid methods to invert blocks of the matrix formed from the diffusion operator.

Recent work by Mousseau, Knoll, and Rider [15] has considered the fully implicit formulation of radiation diffusion using an operator splitting preconditioner similar to the second splitting method mentioned above. They saw this preconditioner to be quite effective in solving one- and two-dimensional problems. In addition, earlier work by Knoll, Rider, and Olson [14] showed that the fully implicit form of the one-dimensional radiation diffusion problem gave greater accuracy in shorter times than did traditional methods. In previous work, we found that the block Jacobi preconditioning method was effective for test problems in three dimensions on parallel computers [5]. Further work has shown that this method is not as effective as we would like for cases where the material coupling dominates the diffusion operator. In this paper, we will compare the four preconditioning methods mentioned above on three-dimensional test problems. We will show an effective, fully implicit, parallel solution strategy for these problems.

The rest of this paper is organized as follows. In the next section, we present the mathematical models we are considering for this work. In sections 3 and 4 we discuss the spatial and temporal discretization techniques used, and in section 5 we detail the four preconditioning methods compared and show analysis indicating what qualitative behavior we expect of each for different problem parameters. We briefly discuss multigrid methods in section 6. In section 7 we give some numerical results showing algorithm performance on problems with various degrees of difficulty and in parallel on problems in three dimensions. Section 8 provides some concluding remarks.

2. Problem formulation. Our model for radiation diffusion is a simplification of the full radiation transport equation given in [16]. We assume isotropic radiation (no angular dependence), Fick's law of diffusion, no scattering effects, and that the photon energy is Planckian, and then integrate the transport equation in frequency to get the diffusion model [2],

$$\frac{\partial E_R}{\partial t} = \nabla \cdot \left(\frac{c}{3\rho\kappa_R(T_R)} \nabla E_R \right) + c\rho\kappa_P(T_M) \cdot \left(aT_M^4 - E_R \right), \tag{2.1}$$

where $E_R(\mathbf{x}, t)$ is the radiation energy density ($\mathbf{x} = (x, y, z)$), $T_M(\mathbf{x}, t)$ is the material temperature, $\rho(\mathbf{x})$ is the material density, c is the speed of light, and $a = 4\sigma/c$, where σ is the Stephan–Boltzmann constant. The Rosseland opacity, κ_R, is a nonlinear function of the radiation temperature, T_R, which is defined by the relation $E_R = aT_R^4$. The Planck opacity, κ_P, is a nonlinear function of material temperature, T_M, which is related to the material energy through an equation of state, $E_M = EOS(T_M)$. In many instances, the two opacities will take on similar values.

We also consider a spatially dependent source term in this equation expressing sources or sinks in the radiation field given by

$$\chi(\mathbf{x}) c a T_{\text{source}}^4, \tag{2.2}$$

where T_{source} is a given source temperature and $\chi(\mathbf{x})$ is a function of the spatial variable $\mathbf{x}$. Computed solutions to (2.1) may result in photon velocities which exceed the speed of light. To prevent this nonphysical phenomenon, a flux-limiter is often added to the diffusion term [2]. We use a flux-limiter of the form $\frac{\|\nabla E_R\|}{E_R}$, where the norm $\|\cdot\|$ is just the l^2 norm of the gradient vector.

The resulting radiation diffusion equation we use as our model is

$$\begin{aligned} \frac{\partial E_R}{\partial t} &= \nabla \cdot \left(\frac{c}{3\rho\kappa_R(T_R) + \frac{\|\nabla E_R\|}{E_R}} \nabla E_R \right) \\ &\quad + c\rho\kappa_P(T_M) \cdot \left(a T_M^4 - E_R \right) + \chi(\mathbf{x}) c a T_{\text{source}}^4. \end{aligned} \tag{2.3}$$

This equation is coupled to an equation expressing conservation of material energy given by

$$\frac{\partial E_M}{\partial t} = -c\rho\kappa_P(T_M) \cdot \left(a T_M^4 - E_R \right). \tag{2.4}$$

We will focus on the development of solution methods for the system (2.3)–(2.4) in what follows.

3. Solution method.

We apply a method of lines approach to the solution of (2.3)–(2.4). The spatial discretization used is as follows. We use a tensor product grid with N_x, N_y, and N_z cells in the x, y, and z directions, respectively. Define $E_{R,i,j,k}(t) \approx E_R(\mathbf{x}_{i,j,k}, t)$ and $E_{M,i,j,k}(t) \approx E_M(\mathbf{x}_{i,j,k}, t)$, with $\mathbf{x}_{i,j,k} = (x_i, y_j, z_k)$ the cell centers. Next, define

$$\mathbf{E}_R \equiv \begin{pmatrix} E_{R,1,1,1} \\ \vdots \\ E_{R,N_x,N_y,N_z} \end{pmatrix} \text{ and } \mathbf{E}_M \equiv \begin{pmatrix} E_{M,1,1,1} \\ \vdots \\ E_{M,N_x,N_y,N_z} \end{pmatrix}.$$

We employ a cell-centered finite difference scheme over the computational mesh and write our discrete equations in terms of a discrete diffusion operator given by $\mathbf{L}(\mathbf{E}_R) \equiv \left(L_{1,1,1}(\mathbf{E}_R), \ldots, L_{N_x,N_y,N_z}(\mathbf{E}_R) \right)^T$, where

$$\begin{aligned} L_{i,j,k}(\mathbf{E}_R) = &\left(\frac{c}{3\rho_{i+1/2,j,k}\kappa_{R,i+1/2,j,k} + \frac{\|\nabla E_R\|_{i+1/2,j,k}}{E_{R,i+1/2,j,k}}} \frac{E_{R,i+1,j,k} - E_{R,i,j,k}}{x_{i+1} - x_i} \right. \\ &\left. - \frac{c}{3\rho_{i-1/2,j,k}\kappa_{R,i-1/2,j,k} + \frac{\|\nabla E_R\|_{i-1/2,j,k}}{E_{R,i-1/2,j,k}}} \frac{E_{R,i,j,k} - E_{R,i-1,j,k}}{x_i - x_{i-1}} \right) \\ &/(x_{i+1/2} - x_{i-1/2}) + \ y \text{ and } z \text{ terms,} \end{aligned} \tag{3.1}$$

and a local operator $\mathbf{S}(\mathbf{E}_R, \mathbf{E}_M) \equiv (S_{1,1,1}(\mathbf{E}_R, \mathbf{E}_M), \ldots, S_{N_x,N_y,N_z}(\mathbf{E}_R, \mathbf{E}_M))^T$, where

$$S_{i,j,k}(E_{R,i,j,k}, E_{M,i,j,k}) = c\rho_{i,j,k}\kappa_{P,i,j,k} \left(a T_{M,i,j,k}^4 - E_{R,i,j,k} \right). \tag{3.2}$$

Thus, our discrete scheme is to find $\mathbf{E}_R(t)$ and $\mathbf{E}_M(t)$ such that

$$\frac{d\mathbf{E}_R}{dt} = \mathbf{L}(\mathbf{E}_R) + \mathbf{S}(\mathbf{E}_R, \mathbf{E}_M) + \mathbf{Q}, \tag{3.3}$$

$$\frac{d\mathbf{E}_M}{dt} = -\mathbf{S}(\mathbf{E}_R, \mathbf{E}_M), \tag{3.4}$$

where $\mathbf{Q} \equiv caT^4_{\text{source}}(\chi(\mathbf{x}_{1,1,1}), \ldots, \chi(\mathbf{x}_{N_x,N_y,N_z}))^T$. The system (3.3)–(3.4) is an ODE system and our time integration technique will be based on ODE time integration methods.

4. Time integration. We have developed a three-dimensional, parallel simulator that employs a fully implicit formulation and solution process for the radiation diffusion model in (3.3)–(3.4) and with which algorithms for radiation diffusion can be studied. In order to allow for accurate time-stepping as well as larger steps than what traditional methods allow, we use an ODE time integrator to handle the temporal discretization. This simulator uses the parallel ODE solver, PVODE [8], developed at Lawrence Livermore National Laboratory and based on the VODPK package (variable order ODE solver with preconditioned Krylov methods; see [7]). PVODE employs the fixed leading coefficient variant of the backward differentiation formula (BDF) method [4, 12] and allows for variation in the order of the time discretization as well as in the time step size. Time step sizes are chosen to minimize the local truncation error and thus give a solution that obeys a user-specified accuracy bound.

This time integration technique leads to a coupled, nonlinear system of equations that must be solved at each time step. For the solution of this system, we use an inexact Newton–Krylov method with Jacobian-vector products approximated by finite differences. As the methods in PVODE are predictor-corrector in nature, an explicit predictor is used for an initial guess in the nonlinear solve.

In the methods discussed above, we use the scaling technique incorporated into PVODE. Thus, we include an absolute tolerance (ATOL) for each unknown and a relative tolerance (RTOL) which is applied to all unknowns. These tolerances are then used to form a weight which is applied to each solution component during the time step from t_{n-1} to t_n. This weight is given as

$$w_i = RTOL|y^i_{n-1}| + ATOL_i \tag{4.1}$$

and is also used to weight a root mean square norm which is applied to all error-like vectors within the solution process. This scaling gives each vector component equal weight when calculating norms. For our application, we supply two absolute tolerances, one to be used with the radiation energy unknowns and one to be used with the material energy unknowns.

5. Preconditioners. The use of Newton–Krylov methods necessitates the use of preconditioning, and we consider several strategies. Before detailing the four preconditioning strategies we compare in this work, we consider the content and structure of the Jacobian matrix we are trying to precondition. We formulate our system of ODEs as $\dot{y} = f(t, y)$, set $y = (\mathbf{E}_R^T, \mathbf{E}_M^T)^T$, and then form f using the right-hand sides of (3.3)–(3.4). The Jacobian matrices used in the Newton method are of the general form $F'(y) = (I - \gamma J)$, where $J = \partial f/\partial y$ is the Jacobian of the nonlinear function f, and the parameter $\gamma \equiv \Delta t\beta$ with Δt the current time step value and β a coefficient depending on the order of the BDF method. Recalling the definitions of the discrete divergence and source operators, defined in (3.1) and (3.2), the block form of the Jacobian of f is

$$J = \begin{pmatrix} \partial \mathbf{L}/\partial \mathbf{E}_R + \partial \mathbf{S}/\partial \mathbf{E}_R & \partial \mathbf{S}/\partial \mathbf{E}_M \\ -\partial \mathbf{S}/\partial \mathbf{E}_R & -\partial \mathbf{S}/\partial \mathbf{E}_M \end{pmatrix} = \begin{pmatrix} A+G & B \\ C & D \end{pmatrix},$$

where $A = \partial \mathbf{L}/\partial \mathbf{E}_R$, $G = \partial \mathbf{S}/\partial \mathbf{E}_R$, $B = \partial \mathbf{S}/\partial \mathbf{E}_M$, $C = -\partial \mathbf{S}/\partial \mathbf{E}_R$, and $D = -\partial \mathbf{S}/\partial \mathbf{E}_M$. We note that G, B, C, and D are all diagonal matrices.

Since Jacobian approximations can be expensive to compute, the preconditioner is not updated with every Newton iteration. Preconditioner updates occur only when the Newton iteration fails to converge, when 20 time steps pass without an update, or when there is a significant change in the time step size and order of the ODE method.

On close inspection of the nonlinear diffusion operator $\mathbf{L}(\mathbf{E}_R)$, we can write

$$\mathbf{L}(\mathbf{E}_R) = \hat{\mathbf{L}}(\mathbf{E}_R)\mathbf{E}_R,$$

where $\hat{\mathbf{L}}$ is a nonlinear matrix-valued function of $\mathbf{E}_R$. In all of our preconditioning strategies, we neglect the nonlinearity in the diffusion term and use the approximation

$$A = \partial \mathbf{L}(\hat{\mathbf{E}}_R)/\partial \mathbf{E}_R \approx \hat{\mathbf{L}}(\hat{\mathbf{E}}_R) \equiv \tilde{A},$$

where $\partial \mathbf{L}(\hat{\mathbf{E}}_R)/\partial \mathbf{E}_R$ is the Jacobian of $\mathbf{L}$ evaluated at a radiation energy, $\hat{\mathbf{E}}_R$. The size of the neglected term is related to the derivatives of the Rosseland opacity and the flux-limiter. Our motivation for neglecting this term arises from the fact that $-\tilde{A}$ is symmetric and positive definite, whereas A has a first-order term that leads to nonsymmetries in its discretized form. This first-order term includes derivatives of the Rosseland opacity and flux-limiter. By neglecting the nonlinearity, this term is removed from the preconditioner, and a symmetric approximation results. In addition, calculation of $\tilde{A}$ is much cheaper than for A, as no derivatives of the flux-limiter need be computed. We also note that computation of the derivative of the flux-limiter may lead to numerical errors if $\nabla \mathbf{E}_R$ approaches 0.

Our preconditioning strategies differ in how they approximately solve systems with the matrix

$$M = I - \gamma \begin{pmatrix} \tilde{A}+G & B \\ C & D \end{pmatrix}.$$

In all cases, multigrid methods are used to invert the $\tilde{A}$ blocks. We will discuss the specifics of the multigrid scheme after detailing the four strategies.

In the following discussion, we will be examining the preconditioner performance responses to changes in the κ_P, κ_R, and Δt parameters in the problem. For these discussions, it will be useful to consider the same radiation diffusion model as given in (3.3)–(3.4) but neglecting flux-limiting and assuming κ_P and κ_R constant. In this case, we see that $\tilde{A} = O(1/\kappa_R)$ and that B, C, and D are all $O(\kappa_P)$.

5.1. Block Jacobi. Our first strategy is to approximate the Jacobian system with

$$M_{\text{Jacobi}} = I - \gamma \begin{pmatrix} \tilde{A}+G & 0 \\ 0 & D \end{pmatrix}.$$

This method effectively neglects the coupling between the radiation and material energy fields.

We now examine the error, $Err_{\text{Jacobi}} = (I - \gamma J) - M_{\text{Jacobi}}$. We see that

$$Err_{\text{Jacobi}} = \gamma \begin{pmatrix} \tilde{A} - A & -B \\ -C & 0 \end{pmatrix}.$$

Thus, the error for this block Jacobi preconditioning strategy is $O(\Delta t)$. In addition, for the non-flux-limited, constant opacity case, we see that $\tilde{A} = A$ and the error is $O(\Delta t \times \kappa_P)$ since both B and C are $O(\kappa_P)$. When $\Delta t \times \kappa_P$ gets large, we would expect this preconditioner to perform poorly.

5.2. Schur complement. Our second preconditioning strategy is to factor the matrix

$$\begin{pmatrix} P & Q \\ R & T \end{pmatrix} \equiv \begin{pmatrix} I - \gamma(\tilde{A} + G) & -\gamma B \\ -\gamma C & I - \gamma D \end{pmatrix} = M$$

into the following:

$$M_{\text{Schur}} = \begin{pmatrix} I & QT^{-1} \\ 0 & I \end{pmatrix} \begin{pmatrix} P - QT^{-1}R & 0 \\ 0 & T \end{pmatrix} \begin{pmatrix} I & 0 \\ T^{-1}R & I \end{pmatrix}.$$

Letting $S = P - QT^{-1}R$, we write the solution to $M_{\text{Schur}}x = b$ as

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} S^{-1}(b_1 - QT^{-1}b_2) \\ T^{-1}(-Rx_1 + b_2) \end{pmatrix}.$$

The error in this preconditioner, $Err_{\text{Schur}} = (I - \gamma J) - M_{\text{Schur}}$, is given by

$$Err_{\text{Schur}} = \gamma \begin{pmatrix} \tilde{A} - A & 0 \\ 0 & 0 \end{pmatrix}.$$

If the Schur complement, S, is exactly inverted, there will be no error associated with this preconditioner for the non-flux-limited, constant opacity case. In addition, because D and hence T are diagonal, there is no penalty associated with inverting T for every iteration of a method that inverts S, as there would be if a material energy diffusion term were added to the equations. Also note that S is formed by modifying the diagonal of P, so we can still employ multigrid methods to invert this Schur complement, as we would to invert the $\tilde{A}$ matrix.

5.3. Matrix split. Our third strategy is motivated by a preconditioner developed in [11] where a splitting of the Jacobian matrix is used. Our preconditioner is written as

$$M_{\text{matrix\_split}} = (I - \gamma J_{\text{diag}})(I - \gamma J_{\text{border}}), \tag{5.1}$$

where

$$J_{\text{diag}} = \begin{pmatrix} \tilde{A} + G & 0 \\ 0 & 0 \end{pmatrix} \text{ and } J_{\text{border}} = \begin{pmatrix} 0 & B \\ C & D \end{pmatrix}.$$

Solving systems of the form $M_{\text{matrix\_split}}x = b$ requires two steps. The first step consists of a solve with the system $(I - \gamma J_{\text{diag}})y = b$, and the second step consists of a solve with the system $(I - \gamma J_{\text{border}})x = y$. Multigrid methods can be used to solve the first system.

To see how the second system can be easily inverted, we consider a reordering of the unknowns and equations of the system, so that unknowns are first ordered by space and then by energy type for each spatial point. Equations are reordered similarly. With this new ordering, $I - \gamma J_{\text{border}}$ will be a block diagonal matrix with 2×2 blocks. Each of these blocks can be written as

$$(I - \gamma J_{\text{border}})_i = \begin{pmatrix} 1 & b_i \\ c_i & d_i \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ c_i & 1 \end{pmatrix} \begin{pmatrix} 1 & b_i \\ 0 & d' \end{pmatrix},$$

where $d' = d_i - b_i c_i$. Solutions of the second step in the application of $M_{\text{matrix\_split}}$ are easily obtained with this factorization.

The error in this preconditioner, $Err_{\text{matrix\_split}} = (I - \gamma J) - M_{\text{matrix\_split}}$, is given by

$$Err_{\text{matrix\_split}} = \gamma \begin{pmatrix} -A + \tilde{A} & 0 \\ 0 & 0 \end{pmatrix} - \gamma^2 \begin{pmatrix} 0 & (\tilde{A} + G)B \\ 0 & 0 \end{pmatrix}. \tag{5.2}$$

For the non-flux-limited, constant coefficient case, $\tilde{A} = A$. The term $(\tilde{A} + G)B$ is of order $O((\frac{1}{\kappa_R} + \kappa_P)\kappa_P) \approx O(\kappa_P^2)$. Thus, this preconditioner has error $O((\Delta t)^2 \times \kappa_P^2)$. So, as $\Delta t \times \kappa_P$ gets large, we would expect this preconditioner's effectiveness to deteriorate rapidly.

5.4. Operator split. Our last strategy is motivated by a preconditioner developed in [6] where an operator splitting of the Jacobian operator is used to split the preconditioning into two steps. This preconditioner is very similar to the previous strategy except that the G term is part of the second step, rather than the first. This preconditioner is written as

$$M_{\text{operator\_split}} = (I - \gamma J_{\text{diff}})(I - \gamma J_{\text{coupling}}), \tag{5.3}$$

where

$$J_{\text{diff}} = \begin{pmatrix} \tilde{A} & 0 \\ 0 & 0 \end{pmatrix} \text{ and } J_{\text{coupling}} = \begin{pmatrix} G & B \\ C & D \end{pmatrix}.$$

Again, the first step consists of a solve with the system $(I - \gamma J_{\text{diff}})y = b$, and the second step consists of a solve with the system $(I - \gamma J_{\text{coupling}})x = y$. Multigrid methods can be used to solve the first system.

The second system here is also easily inverted by a simple LU decomposition of a reordered problem. However, the 2×2 blocks have a nonidentity upper left entry, so that the decomposition is

$$(I - \gamma J_{\text{coupling}})_i = \begin{pmatrix} g_i & b_i \\ c_i & d_i \end{pmatrix} = \begin{pmatrix} g_i & 0 \\ c_i & 1 \end{pmatrix} \begin{pmatrix} 1 & b_i/g_i \\ 0 & d' \end{pmatrix},$$

where $d' = d_i - (b_i c_i)/g_i$.

The error associated with this preconditioner, $Err_{\text{operator\_split}} = (I - \gamma J) - M_{\text{operator\_split}}$, is given by

$$Err_{\text{operator\_split}} = \gamma \begin{pmatrix} -A + \tilde{A} & 0 \\ 0 & 0 \end{pmatrix} - \gamma^2 \begin{pmatrix} \tilde{A}G & \tilde{A}B \\ 0 & 0 \end{pmatrix}. \tag{5.4}$$

Again, for the non-flux-limited, constant coefficient case, $\tilde{A} = A$ and the terms $\tilde{A}G$ and $\tilde{A}B$ are of order $O(\frac{\kappa_P}{\kappa_R})$. Thus, this preconditioner has $O((\Delta t)^2 \times \frac{\kappa_P}{\kappa_R})$ error.

If $\kappa_R \approx \kappa_P$, we would expect this preconditioner to show minimal deterioration in effectiveness as the opacities get large.

Note that $G = -C$ and $B = -D$ so that as $\Delta t \to \infty$, the second stage of this preconditioner $I - \gamma J_{\text{coupling}}$ becomes singular. As a result, we may expect this preconditioner to deteriorate for extremely large time steps.

6. Multigrid methods. The Rosseland opacity will exhibit large changes where material interfaces exist in the domain. The temperature dependence gives rise to large value changes as well. These changes imply that the problem can be very heterogeneous. As a result, to invert matrix blocks formed from the diffusion operator, we use a multigrid method designed to handle large changes in problem coefficients. In particular, we use 1 V-cycle of the ParFlow multigrid (PFMG) algorithm developed by Ashby and Falgout [1] as our multigrid solver. Other multigrid methods have been developed for highly heterogeneous problems. A comparison of PFMG and another of these methods can be found in [13]. We use PFMG here because it is fast and scales extremely well. More information about multigrid methods can be found in [3].

7. Numerical results. To understand how these preconditioning strategies perform for problems with varying degrees of difficulty, we performed a number of studies. Our first study looked at the effects of changing the relative weighting of the diffusion and coupling terms in the radiation equation by setting the two opacities constant and equal and then investigating preconditioner responses to increasing the value. Our second study looked at the effects of tabulated opacities, as are currently used in applications of interest, on these preconditioners. Last, we performed a parallel scaling study with the most effective preconditioner to verify algorithmic scalability of the solution method.

For all runs, we used the PVODE package default settings, with the following exception. Some of our tests led to nonstable solutions with higher-order methods, so we have limited the ODE method order to 2 for all cases, except where explicitly noted. We are presently looking into why these situations occur. Some work has been done in the area of avoiding these sorts of instabilities [10], and we will investigate its applicability here. Note that the default setting for the maximum number of GMRES iterations for PVODE is 5. No restart is performed.

7.1. Constant opacity results. In our first study, we set the Rosseland and Planck opacities equal to a single parameter, κ. We then changed the value of this parameter from $\kappa = 1$ to $\kappa = 100{,}000$.

For this problem, we set $E_M = T_M$. The system (2.3)–(2.4) is solved on the box $\mathcal{D} \equiv \{\mathbf{x} = (x, y, z) : 0 \le x, y, z \le 1\text{cm}\}$. The function $\chi(\mathbf{x})$ in (2.3) is defined by

$$\chi(\mathbf{x}) = \begin{cases} 1 & \text{if } 0.4 \le x, y, z \le 0.6, \\ 0 & \text{otherwise.} \end{cases} \tag{7.1}$$

The parameter T_{source} was 3,481,440 $°K$ (approximately $300eV$), and the initial conditions were taken as $E_R = aT_{R,0}^4$ and $E_M = T_{M,0}$, where $T_{R,0} = T_{M,0} = 300\ °K$. Dirichlet boundary conditions were consistent with the initial conditions. The density was taken to be 1.0g/cc. The spatial grid was uniform with $N_x = N_y = N_z = 20$.

We examined solver statistics at 50 intervals of about 0.1002s with a final simulation time of about 5.0104s. Flux-limiting was applied to the problem as discussed above. We asked for a relative tolerance on each solution component of 10^{-4} and an absolute tolerance on each of the energies of 200.

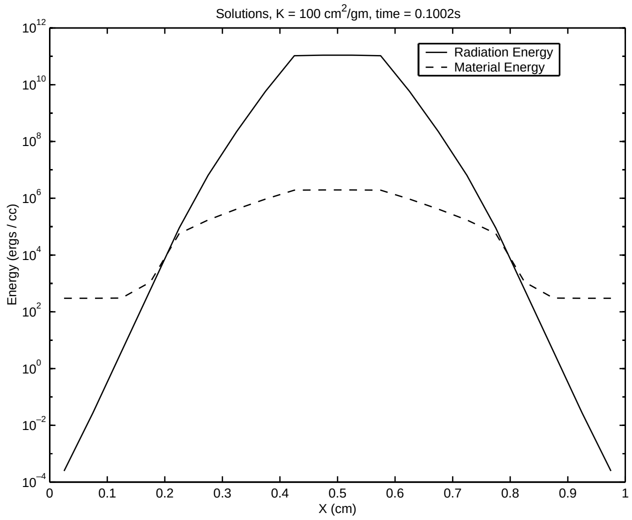


FIG. 7.1. *Radiation and material energies for* $\kappa = 100$*, time* $= 0.1002\text{s}, y = z = 0.475\text{cm}$.

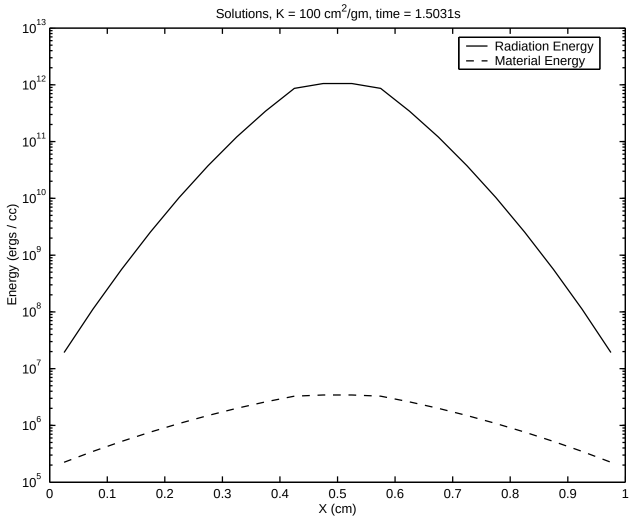


FIG. 7.2. *Radiation and material energies for* $\kappa = 100$*, time* $= 1.5031\text{s}, y = z = 0.475\text{cm}$.

Figures 7.1 and 7.2 show the solutions over the x-line for $y = z = 0.475\text{cm}$ at times 0.1002s and 1.5031s for the value $\kappa = 100\text{cm}^2/\text{g}$. The time 0.1002s is the first output time we recorded. The solutions have felt the effects of the source at this point. By 1.5031s, the energies have increased due to the source at the domain center. We also see the effects of the diffusion operator spreading out the radiation energy.

Figures 7.3 and 7.4 show the solutions at these times but for $\kappa = 10^5\text{cm}^2/\text{g}$. Here the "bump" in the domain center is much more pronounced. This difference from the lower κ case is due to the increased coupling between the two energy fields. For higher values of κ, the radiation energy diffuses much less, and more of its energy is transferred to the material. For these cases, the local physics is clearly dominating the calculation.

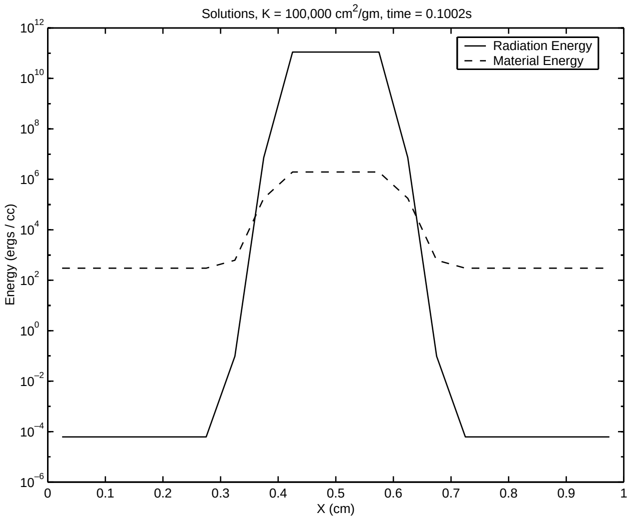


FIG. 7.3. *Radiation and material energies for* $\kappa = 10^5$*, time* $= 0.1002$s*,* $y = z = 0.475$cm.

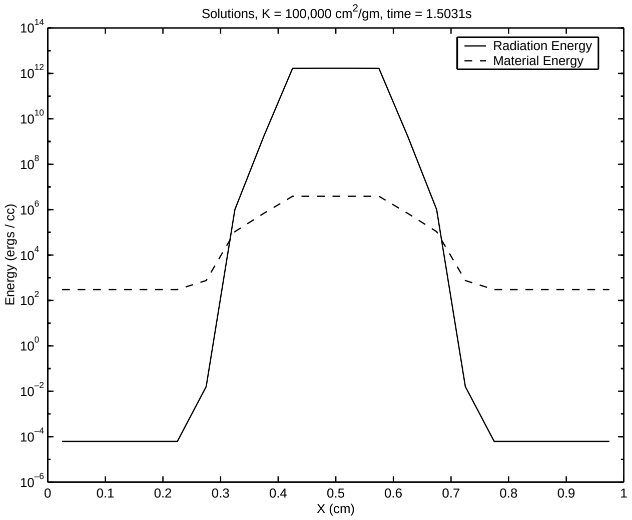


FIG. 7.4. *Radiation and material energies for* $\kappa = 10^5$*, time* $= 1.5031$s*,* $y = z = 0.475$cm.

Table 7.1 shows the cumulative solver statistics for these runs. In this and subsequent tables,

S	=	Schur preconditioner,
BJ	=	block Jacobi preconditioner,
MS	=	matrix split preconditioner, and
OS	=	operator split preconditioner,

and the statistical counters are

Table 7.1
Solver statistics for constant opacity problem.

PC	κ	NST	NNI	NLI	RT	NCFN	NCFL
S	1	1,831	1,884	2,168	2,648	0	0
BJ	1	1,830	1,886	2,167	2,711	0	0
MS	1	1,830	1,884	2,171	2,617	0	0
OS	1	1,830	1,884	2,170	2,626	0	0
S	100	1,782	1,875	3,065	3,145	0	0
BJ	100	1,778	1,874	3,273	3,366	0	0
MS	100	1,786	1,894	3,141	3,213	3	17
OS	100	1,782	1,877	2,995	3,129	0	0
S	10,000	660	723	1,271	1,315	0	0
BJ	10,000	694	831	2,299	2,116	8	88
MS	10,000	2,828	4,939	19,343	17,371	930	2,893
OS	10,000	670	733	2,285	1,950	0	134
S	100,000	424	474	786	846	0	0
BJ	100,000	1,288	2,084	6,010	5,822	88	439
MS	100,000	2,359	3,787	14,780	12,995	518	2,090
OS	100,000	650	949	2,912	2,609	8	214

NST = total number of time steps,
NNI = total number of nonlinear iterations,
NLI = total number of linear iterations,
NFE = total number of $f(t, y)$ evaluations,
NPE = total number of preconditioner evaluations,
NPS = total number of preconditioner solves,
HU = step size that was used on the last step (scaled by c),
RT = run time in seconds,
NCFN = total number of nonlinear convergence failures, and
NCFL = total number of linear convergence failures.

We see that the Schur preconditioner is consistently performing better and faster than the others. As the coupling term between the radiation and material grows in weight relative to the diffusion term, we see that the matrix split preconditioner is the first to show significant signs of struggle. This degradation in the matrix split preconditioner performance is expected since its error was on the order of κ_P^2. However, all but the Schur preconditioner are struggling for $\kappa = 10^4$ and $\kappa = 10^5$. The operator split preconditioner shows the second best performance, which is also expected since its error is on the order of κ_P/κ_R. For lower opacity values, the preconditioners all perform fairly well.

Figure 7.5 shows the cumulative numbers of nonlinear iterations taken by each of the preconditioners for the 50 output times for the two cases of $\kappa = 100\text{cm}^2/\text{g}$ and $\kappa = 10^5\text{cm}^2/\text{g}$. For the lower κ value, all preconditioners result in about the same number of nonlinear iterations at each time step. For the higher κ value, however, the preconditioners show distinctly different performances. During the transition to steady state, the matrix split preconditioner has the most degradation with the block Jacobi preconditioner also showing degradation, but less. These results bear out the analysis given above in that these two preconditioners have the strongest dependence on the κ value with the matrix split the strongest. After the solution gets close to steady state, however, the four preconditioners all require very few nonlinear iterations to resolve the physics. Similar results bear out for the linear iteration counts.

7.2. Tabular opacities results. In this section we give results of using the above preconditioners on several problems involving the use of tabular opacities. We

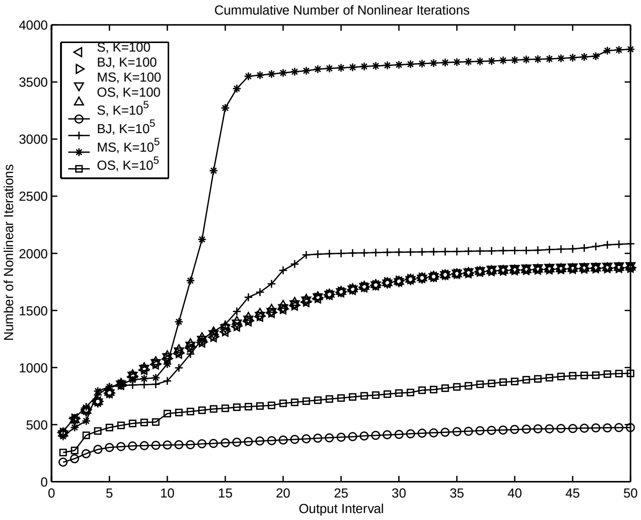


FIG. 7.5. *Cumulative nonlinear iteration counts for all four preconditioners for* $\kappa = 100\text{cm}^2/\text{g}$ *and* $\kappa = 10^5\text{cm}^2/\text{g}$.

use the LEOS package (Livermore Equation of State; see [9]) to give the Rosseland and Planck opacities as nonlinear functions of the radiation temperature T_R and material temperature T_M, respectively. The system (2.3)–(2.4) is solved on the box $\mathcal{D} \equiv \{\mathbf{x} = (x, y, z) : 0 \leq x, y, z \leq 1\text{cm}\}$ with Dirichlet boundary conditions. The function $\chi(\mathbf{x})$ in (2.3) is defined by

$$\chi(\mathbf{x}) = \begin{cases} 1 & \text{if } 0.3 \leq x, y, z \leq 0.7, \\ 0 & \text{otherwise.} \end{cases} \tag{7.2}$$

The parameter T_{source} was 3,481,440 °K (approximately $300eV$), and the initial conditions were taken as $E_R = aT_{R,0}^4$ and $E_M = EOS(\rho, T_{M,0})$, where $T_{R,0} = T_{M,0} =$ 116,100 °K (approximately $10eV$), and $EOS(\rho, T_M)$ is the equation of state function in the LEOS package giving E_M as a function of ρ and T_M. The Dirichlet boundary values for E_R and E_M are taken to be consistent with the initial conditions. The material used was carbon at a reference density of $\rho = 1.05\text{g/cc}$. The spatial grid was uniform with $N_x = N_y = N_z = 20$.

For this problem, the time behavior consists of an initial transient in which the material heats up in the region of the source (from 0 to .01 microseconds), followed by a radiation front traveling to the boundary (continuing to .41 microseconds), and then a final phase in which the solution approaches a steady state (integrated to about 1.33 microseconds). Figures 7.6 and 7.7 show the solutions plotted on the line $y = z = 0.475\text{cm}$ at .01 and .03 microseconds. The only preconditioner that was effective for the entire course of the simulation was the Schur preconditioner. We note that for the initial conditions, the starting values of the Rosseland and Planck opacities are on the order of 10^4 and 10^5, respectively. Table 7.2 compares the statistics of the PVODE solver at .01 microseconds.

At this early output time, the matrix split preconditioner is actually performing the best. However, in the next phase of the solution all the preconditioners start having large numbers of linear convergence failures except for the Schur preconditioner. It has zero linear and nonlinear convergence failures for this problem. When a linear

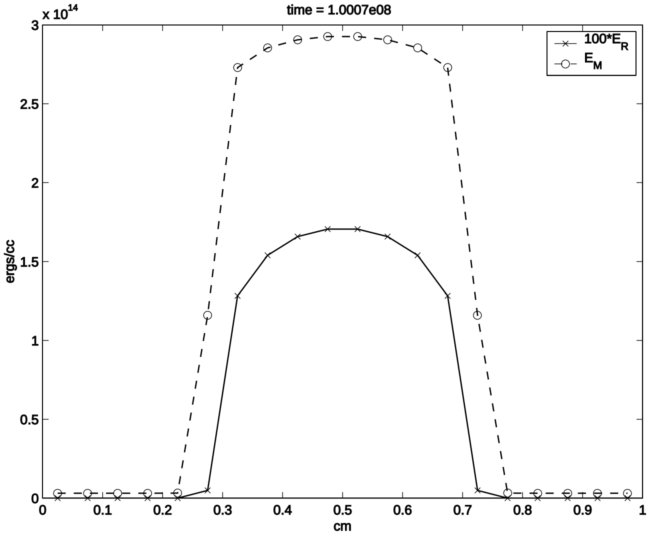


Fig. 7.6. *Snapshot of* E_R *and* E_M *for the first LEOS problem on the line* $y = z = 0.475\text{cm}$ *at* .01 *microseconds.*

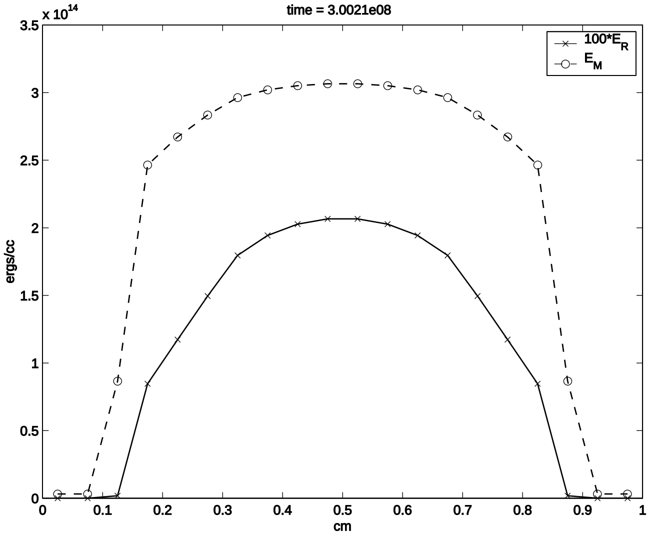


Fig. 7.7. *Snapshot of* E_R *and* E_M *for the first LEOS problem on the line* $y = z = 0.475\text{cm}$ *at* .03 *microseconds.*

Table 7.2

Statistics for LEOS problem 1 *at* .01 *microseconds.*

PC	NST	NNI	NLI	NFE	NPE	NPS	HU	RT	NCFN	NCFL
S	323	420	562	985	59	975	1.52	298.43	0	0
BJ	305	518	1,023	1,544	68	1,534	.95	488.59	0	14
MS	226	308	280	591	54	581	6.54	188.91	0	0
OS	493	663	1,511	2,179	103	2,164	.78	663.87	0	33

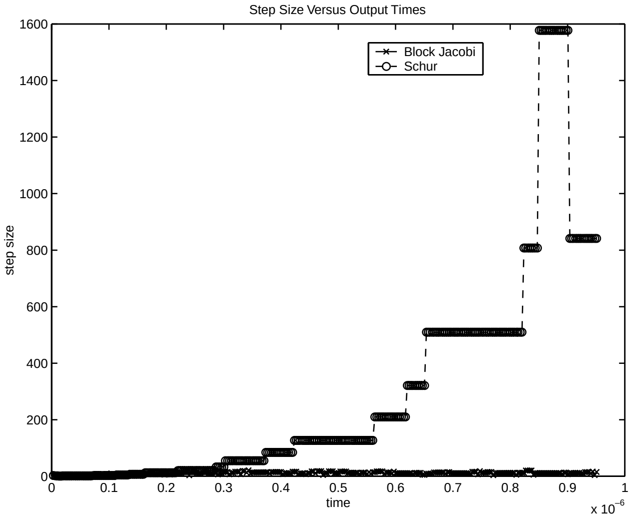


FIG. 7.8. *Plot of step size comparisons for first LEOS problem. Note that the time steps are actually $c\Delta t$.*

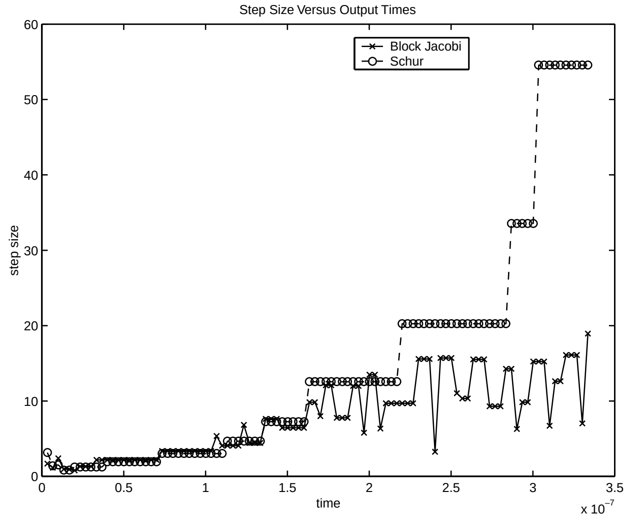


FIG. 7.9. *Plot of step size comparisons for first LEOS problem at early times. Note that the time steps are actually $c\Delta t$.*

convergence failure occurs and the preconditioner is current, the PVODE solver reduces the step size and tries the step over. This has the effect of increasing the total number of steps for the simulation. Also note that the step sizes used by PVODE are much larger than one would expect for the split and Jacobi preconditioners to be effective. With step sizes of order 1, the errors in the split and Jacobi preconditioners are extremely large. Hence, it is not hard to understand the failure of these preconditioners (or their high cost since the step sizes must be kept small) for the latter part of the simulation. Figures 7.8 and 7.9 show the step size behavior as a function of output times for the Schur and block Jacobi preconditioners. While the step sizes change

Table 7.3
Statistics for LEOS problem 1 at 1.33 microseconds.

PC	NST	NNI	NLI	NPE	HU	RT	NCFN	NCFL
S	2,654	2,942	5,930	189	4,421	2,592	0	0
BJ	>4,197	>4,969	>23,387	>451	15.11	>7,208	>55	>1,720
MS	>3,226	>5,261	>20,691	>1,653	.00166	>6,559	>473	>1,680
OS	>4,743	>6,720	>22,528	>962	.321	>6,867	>10	>185

Table 7.4
Statistics for LEOS problem 1 restricted to first order.

PC	NST	NNI	NLI	Final time reached (μs)	RT	NCFN	NCFL
S	9,236	9,726	14,725	.0097	7,142	0	0
BJ	5,970	6,280	16,256	.0047	6,862	0	18
MS	6,564	8,177	13,717	.0031	6,894	191	701
OS	5,629	5,968	18,706	.0053	7,161	0	64

fairly smoothly for the Schur preconditioner, the behavior has a sawtooth flavor for the block Jacobi preconditioner. The final statistics for the Schur preconditioner are given in Table 7.3, as well as the final computed statistics for the other preconditioners. For this problem, we requested 400 output snapshots. The Schur preconditioner finished the computation in under a 2-hour limit, while the block Jacobi reached 285 output points, the operator split reached 14, and the matrix split reached only 7. For the Schur preconditioner, there were on average 1.1 nonlinear iterations per time step and 2.0 linear iterations per nonlinear iteration.

For this first problem, we also investigated the effect of restricting the ODE solver to first order (i.e., backward Euler). While this had a significant effect on reducing the number of linear and nonlinear convergence failures for all but the matrix split preconditioner, the number of time steps increased dramatically. As a result, all of the preconditioners failed to produce the requested 400 output points within the 2-hour run time limit. Table 7.4 gives statistics and final times reached for each preconditioner. From these results, it is apparent that the higher-order time integration methods can be extremely effective in reducing overall run time costs.

A second problem was run with hydrogen as the material at a reference density of $\rho = .874$g/cc. This problem has the same general behavior as the first, except that the time to reach steady state is an order of magnitude lower, i.e., the simulation was run to .133 microseconds. Figures 7.10, 7.11, and 7.12 show the solutions plotted on the line $y = z = 0.475$cm at .01, .02, and .133 microseconds. Table 7.5 contains the statistics for this problem. As before, the Schur preconditioner was the only effective preconditioner for the entire run. There were on average 1.1 nonlinear iterations per time step, and 2.15 linear iterations per nonlinear iteration. Note that there were 13 linear convergence failures and 1 nonlinear failure.

7.3. Scalability study with tabulated opacities. A scalability study was performed on a third problem. The system (2.3)–(2.4) was solved on the box $\mathcal{D} \equiv \{\mathbf{x} = (x, y, z) : 0 \leq x, y, z \leq 1\text{cm}\}$ with Dirichlet boundary conditions. The function $\chi(\mathbf{x})$ in (2.3) is defined by (7.1). The parameter T_{source} was 3,481,440 °K, and the initial conditions were obtained using $T_{R,0} = T_{M,0} = 300$ °K. The Dirichlet boundary values were taken to be consistent with the initial conditions. The material used was carbon at a reference density of $\rho = 1.05$g/cc, and the spatial grid per processor was uniform with $N_x = N_y = N_z = 40$. Thus, problem size and computational resources

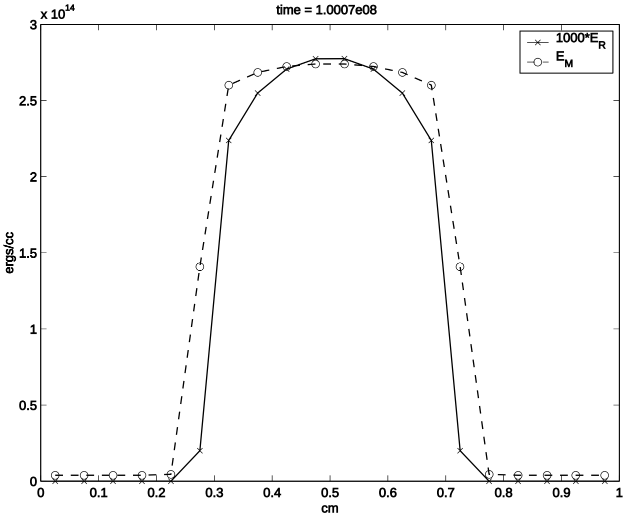


FIG. 7.10. *Snapshot of E_R and E_M for the second LEOS problem on the line $y = z = 0.475$cm at .01 microseconds.*

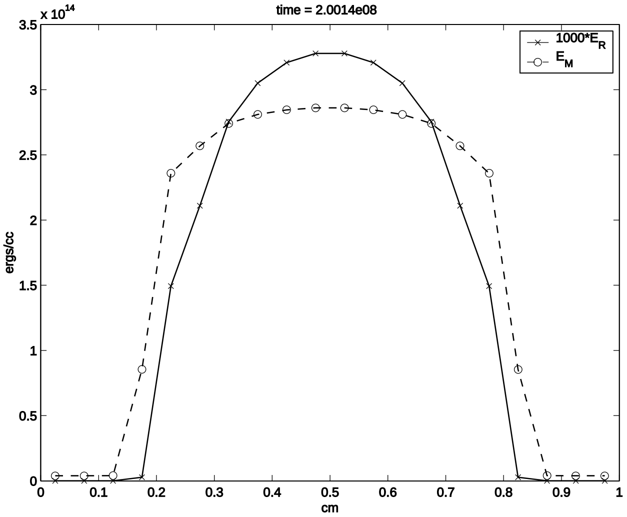


FIG. 7.11. *Snapshot of E_R and E_M for the second LEOS problem on the line $y = z = 0.475$cm at .02 microseconds.*

were simultaneously increased for this study. Only the Schur preconditioner was used. Table 7.6 contains the results of the scalability study. The reported scaled efficiency for a run on N processors was calculated by dividing the run time for the single processor case by the run time for the N processor case. As can be seen, except for the run times, all the statistics scaled extremely well. (We note that when this study was performed, the run time environment on the IBM ASCI Blue Pacific machine at LLNL was under a state of flux. Earlier scalability studies performed showed a much better scalability of run times.) The simulation was run until approximately .001 microseconds, which is very early in the time history.

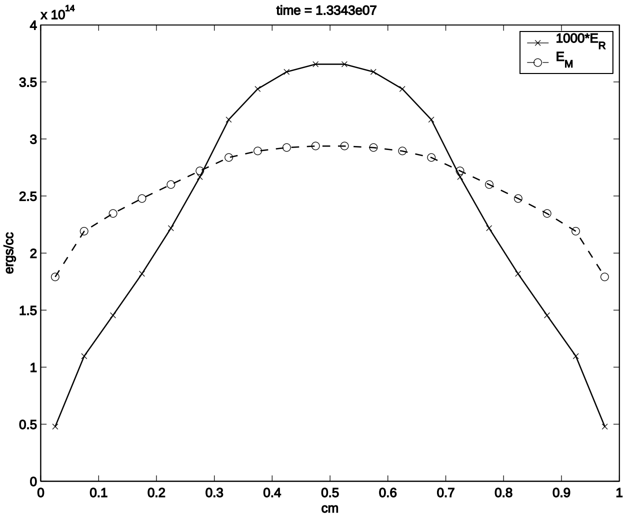


FIG. 7.12. *Snapshot of E_R and E_M for the second LEOS problem on the line $y = z = 0.475$cm at* .133 *microseconds.*

TABLE 7.5
Statistics for LEOS problem 2 *at* .133 *microseconds.*

PC	NST	NNI	NLI	NFE	NPE	NPS	HU	RT	NCFN	NCFL
S	834	943	2,029	2,976	81	2,967	26.69	879	1	13

TABLE 7.6
Statistics for scalability study.

Processor topology	NST	NNI	NLI	NPE	RT	RT scaled efficiency	Avg. cost per step
$1 \times 1 \times 1$	217	329	423	70	2,015	–	9.3
$2 \times 2 \times 2$	214	324	411	75	2,287	88.1%	10.7
$4 \times 4 \times 4$	196	295	378	66	2,220	90.7%	11.3
$8 \times 8 \times 8$	197	273	374	57	2,575	78.2%	13.1
$16 \times 8 \times 8$	190	273	376	60	3,106	64.8%	16.4

8. Conclusions. We have presented a comparison of four preconditioning strategies for Jacobian systems arising in the fully implicit solution of radiation diffusion coupled with material energy transfer. The four preconditioning methods are block Jacobi, Schur complement, and operator splitting approaches that split the preconditioner solve into two steps. From our results, it is apparent that the Schur complement approach is clearly seen to be the most effective for a large range of weightings between the diffusion and energy coupling terms. For problems using tabulated opacities, the Schur preconditioner outperformed the other preconditioners by a wide margin. One conclusion we can draw from our studies is that it appears to be more effective to use full matrix approaches to developing preconditioners for radiation transport problems rather than approaches based on splittings or on only parts of the matrix.

While limiting the time integration methods to first order helps lower the number of step failures, there is a marked increase in the number of steps. At least for the problems we have considered, the better preconditioner allows for much larger step sizes within the allowed error bounds used by the PVODE solver, and this significantly reduces the overall work. Our parallel scaling study demonstrated good algorithmic scalability of our solution approach. Finally, when material diffusion is added to equation (2.4), the Schur approach may fail to be competitive. In addition, extending the physics in the problem to include multigroup diffusion (where the radiation energy spectrum is resolved) leads to a Schur complement that is a full matrix. Our future work will include exploring the use of system-based multigrid solvers as preconditioners, as well as other multilevel methods, to address material energy diffusion and multigroup energy resolution.

REFERENCES

[1] S. F. Ashby and R. D. Falgout, *A parallel multigrid preconditioned conjugate gradient algorithm for groundwater flow simulations*, Nuclear Science and Engineering, 124 (1996), pp. 145–159.

[2] R. L. Bowers and J. R. Wilson, *Numerical Modeling in Applied Physics and Astrophysics*, Jones and Bartlett, Boston, 1991.

[3] W. F. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial*, 2nd ed., SIAM, Philadelphia, PA, 2000.

[4] P. N. Brown, G. D. Byrne, and A. C. Hindmarsh, *VODE: A variable-coefficient ODE solver*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 1038–1051.

[5] P. N. Brown, B. Chang, F. Graziani, and C. S. Woodward, *Implicit solution of large-scale radiation-material energy transfer problems*, in Iterative Methods in Scientific Computation IV, D. R. Kincaid and A. C. Elster, eds., Comput. Appl. Math. 5, International Association for Mathematics and Computers in Simulations, New Brunswick, NJ, 1999, pp. 343–356.

[6] P. N. Brown and A. C. Hindmarsh, *Reduced storage matrix methods in stiff ODE systems*, Appl. Math. Comput., 31 (1989), pp. 40–91.

[7] G. D. Byrne, *Pragmatic experiments with Krylov methods in the stiff ODE setting*, in Computational Ordinary Differential Equations, J. R. Cash and I. Gladwell, eds., Oxford University Press, Oxford, UK, 1992, pp. 323–356.

[8] G. D. Byrne and A. C. Hindmarsh, *PVODE, an ODE solver for parallel computers*, Int. J. High Perf. Comput. Appl., 13 (1999), pp. 354–365.

[9] E. M. Corey and D. A. Young, *A New Prototype Equation of State Data Library*, Tech. report UCRL-JC-127698, Lawrence Livermore National Laboratory, 1997. American Physical Society Meeting, submitted.

[10] A. C. Hindmarsh, *Avoiding BDF stability barriers in the MOL solution of advection-dominated problems*, Appl. Numer. Math., 17 (1995), pp. 311–318.

[11] A. C. Hindmarsh and M. D. Rotter, *Using an ODE solver for a class of integro-differential systems*, J. Comput. Phys., to appear.

[12] K. R. Jackson and R. Sacks-Davis, *An alternative implementation of variable step-size multistep formulas for stiff ODEs*, ACM Trans. Math. Software, 6 (1980), pp. 295–318.

[13] J. E. Jones and C. S. Woodward, *Newton-Krylov-multigrid solvers for large-scale, highly heterogeneous, variably saturated flow problems*, Advances in Water Resources, to appear.

[14] D. A. Knoll, W. J. Rider, and G. L. Olson, *An efficient nonlinear solution method for nonequilibrium radiation diffusion*, J. Quant. Spec. and Rad. Trans., 63 (1999), pp. 15–29.

[15] V. A. Mousseau, D. A. Knoll, and W. J. Rider, *Physics-based preconditioning and the Newton-Krylov method for non-equilibrium radiation diffusion*, J. Comput. Phys., 160 (2000), pp. 743–765.

[16] G. C. Pomraning, *The Equations of Radiation Hydrodynamics*, Pergamon Press, New York, 1973.

SIAM J. SCI. COMPUT.
Vol. 23, No. 2, pp. 517–541

TOWARD THE OPTIMAL PRECONDITIONED EIGENSOLVER: LOCALLY OPTIMAL BLOCK PRECONDITIONED CONJUGATE GRADIENT METHOD*

ANDREW V. KNYAZEV†

Abstract. We describe new algorithms of the locally optimal block preconditioned conjugate gradient (LOBPCG) method for symmetric eigenvalue problems, based on a local optimization of a three-term recurrence, and suggest several other new methods. To be able to compare numerically different methods in the class, with different preconditioners, we propose a common system of model tests, using random preconditioners and initial guesses. As the "ideal" control algorithm, we advocate the standard preconditioned conjugate gradient method for finding an eigenvector as an element of the null-space of the corresponding homogeneous system of linear equations under the assumption that the eigenvalue is known. We recommend that every new preconditioned eigensolver be compared with this "ideal" algorithm on our model test problems in terms of the speed of convergence, costs of every iteration, and memory requirements. We provide such comparison for our LOBPCG method. Numerical results establish that our algorithm is practically as efficient as the "ideal" algorithm when the same preconditioner is used in both methods. We also show numerically that the LOBPCG method provides approximations to first eigenpairs of about the same quality as those by the much more expensive global optimization method on the same generalized block Krylov subspace. We propose a new version of block Davidson's method as a generalization of the LOBPCG method. Finally, direct numerical comparisons with the Jacobi–Davidson method show that our method is more robust and converges almost two times faster.

Key words. symmetric eigenvalue problems, preconditioning, conjugate gradient methods, the Lanczos method

AMS subject classifications. 65F15, 65N25

PII. S1064827500366124

1. Introduction.

We consider a generalized *symmetric definite* eigenvalue problem of the form $(A-\lambda B)x = 0$ with real symmetric n-by-n matrices A and B, assuming that A is positive definite. That describes a regular matrix pencil $A - \lambda B$ with a discrete spectrum (set of eigenvalues λ). It is well known that such a generalized eigenvalue problem has all real eigenvalues λ_i, and corresponding (right) eigenvectors x_i, satisfying $(A - \lambda_i B)x_i = 0$, can be chosen orthogonal in the following sense: $(x_i, Ax_j) = (x_i, Bx_j) = 0,\ i \neq j$. In some applications, the matrix B is simply the identity $B = I$, and then we have the standard symmetric eigenvalue problem with matrix A, which has n real positive eigenvalues

$$0 < \lambda_{\min} = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n = \lambda_{\max}.$$

In general, when $B \neq I$, the pencil $A - \lambda B$ has n real, some possibly infinite, eigenvalues. If B is nonsingular, all eigenvalues are finite. If B is positive semidefinite, some eigenvalues are infinite, all other eigenvalues are positive, and we consider the

*Received by the editors May 18, 2000; accepted for publication (in revised form) October 6, 2000; published electronically July 10, 2001. The main results of the paper were presented by the author at the Mini Symposium on Very Large Eigenvalue Problems at the Fifth U.S. National Congress on Computational Mechanics, University of Colorado at Boulder, 1999, and at the Sixth Copper Mountain Conference on Iterative Methods, Copper Mountain, CO, 2000. A preliminary version of the paper was published as a technical report [16].

http://www.siam.org/journals/sisc/23-2/36612.html

†Department of Mathematics, University of Colorado at Denver, P.O. Box 173364, Campus Box 170, Denver, CO 80217-3364 (andrew.knyazev@cudenver.edu, http://www-math.cudenver.edu/~aknyazev).

problem of computing the smallest m eigenvalues of the pencil $A - \lambda B$. When B is indefinite, it is convenient to consider the pencil $\mu A - B$ with eigenvalues

$$\mu = \frac{1}{\lambda},\ \mu_{\min} = \mu_n \leq \cdots \leq \mu_1 = \mu_{\max},$$

where we want to compute the largest m eigenvalues, $\mu_1, \ldots \mu_m$, and corresponding eigenvectors.

An important class of eigenproblems is the class of *mesh eigenproblems*, arising from discretizations of boundary value problems with self-adjoint differential operators of mathematical physics. Such problems appear, e.g., in structural mechanics, where it is usual to call A a *stiffness* matrix and B a *mass* matrix. A mass matrix is usually positive definite, but in some applications, e.g., in buckling, the matrix B is only nonnegative, or even indefinite, while A is positive definite.

Typical properties of mesh eigenproblems are well known; see, e.g., [22]. We just want to highlight that the desired eigenpairs of the matrix pencil $B - \mu A$ are rarely needed with high accuracy as the pencil itself is just an approximation of the original continuous problem and the approximation error may not be small in practice. It means that the typical ratio of the number of iterations and the number of unknowns should be small when a preconditioner is of a reasonable quality. For that reason, in the present paper we are not much interested in such properties of eigensolvers, e.g., in superlinear convergence, which could be observed only after large number of steps.

In the rest of the paper, for brevity, we deal with the pencil $B - \mu A$ mostly. With $B = I$ and $\lambda = 1/\mu$, our results and arguments are readily applied for the pencil $A - \lambda I$.

The paper is organized as follows.

In section 2, we introduce, following [21, 22], preconditioning for eigenvalue solvers and give general definitions of preconditioned single-vector and block, or simultaneous, iterative eigensolvers. We describe the global optimization method on the corresponding generalized Krylov subspace. No efficient algorithm is presently known to perform a global optimization of the Rayleigh quotient on the generalized Krylov subspace. We shall show numerically in section 8, however, that the method we suggest in section 4 provides approximations often quite close to those of the global optimization, at a small fraction of the cost.

In section 3, we outline the "ideal" control algorithm, namely, the standard preconditioned conjugate gradient method, for finding an eigenvector as an element of the null-space of the corresponding homogeneous system of linear equations under assumption that the eigenvalue is known. The algorithm cannot, of course, be used in practice as it requires knowledge of the extreme eigenvalue, but it seems to be a perfect choice as a benchmark for preconditioned eigensolvers for computing the extreme eigenpair.

In section 4, we describe, in some details, a new algorithm of the locally optimal preconditioned conjugate gradient method, based on the local optimization of the three-term recurrence suggested by the author in [19, 21, 22]. In the original algorithm of [19], the three-term recurrence contains the current eigenvector approximation, the current preconditioned residual, and the previous eigenvector approximation. As the current eigenvector approximation and the previous eigenvector approximation get closer in the process of iterations, special measures need to be used in the algorithm to overcome the potential instability. In our new algorithm, the three-term recurrence contains the current eigenvector approximation, the current preconditioned residual, and the implicitly computed difference of the current and previous eigenvector approximations. Such choice resolves instability issues and allows us to write a much simpler and more efficient code.

We present block versions of the method, the locally optimal block preconditioned conjugate gradient (LOBPCG) methods for symmetric eigenvalue problems, in section 5.

To be able to compare numerically different methods in the class, with different preconditioners, we suggest, in section 6, a common system of model tests, using random preconditioners and initial guesses. We recommend that every new preconditioned eigensolver for computing the extreme eigenpair be compared with our "ideal" algorithm on our model test problems in terms of the speed of convergence, costs of every iteration, and memory requirements. We also recommend a comparison with the global optimization method in terms of accuracy.

We provide such comparison for our LOBPCG method in sections 7 and 8. Numerical results of section 7 establish that our algorithm is practically as efficient as the "ideal" algorithm when the same preconditioner is used in both methods. In section 8 we show numerically that the block version of our method comes close to finding the global optimum of the Rayleigh quotient on the corresponding generalized block Krylov subspace.

Section 9 contains an informal discussion of the block Davidson method. We describe a nonstandard restart strategy that makes the block Davidson method a generalization of our LOBPCG method. We argue, however, that such generalization may not be beneficial for symmetric eigenvalue problems.

In section 10, we compare directly our method with a version of the Jacobi–Davidson method [14] for $B = I$. No MATLAB code of the Jacobi–Davidson method for a generalized eigenvalue problem is currently publicly available. We find that our method is much more robust and typically converges almost two times faster. This is not very surprising, as the MATLAB version of the Jacobi–Davidson method available to us for testing is not apparently optimized for symmetric eigenvalue problems, while our method takes full advantage of the symmetry by using a three-term recurrence.

Finally, section 11 contains references to some relevant software written by the author.

We note that the simplicity, robustness, and fast convergence of preconditioned eigensolvers we propose make them a more and more popular choice in applications, e.g., in band structure calculations in two- and three-dimensional photonic crystals [6, 7] and eigenvalue problems for thin elastic structures [32]. Some eigenvalue problems in mechanics, e.g., vibration of a beam supported by springs, lead to equations with nonlinear dependence on the spectral parameter. Preconditioned eigensolvers for such equations are analyzed in [39, 40], where, in particular, a generalization of the theory of a preconditioned subspace iteration method of [9, 10] is presented.

2. Preconditioning for eigenvalue problems.

First, we briefly review a traditional approach for large symmetric generalized eigenproblems, based on using classical methods, e.g., the Lanczos method, for a shifted-and-inverted operator $(B - \nu A)^{-1}A$. It typically lets us quickly compute the eigenvalues closest to the shift ν, assuming that this operation may be implemented with an efficient factorization of $B - \nu A$. However, for very large problems such factorizations are usually too expensive. An inner iterative solver is often used to somewhat circumvent this difficulty; see a review and references in [21, 22] and a recent paper [38].

If B is efficiently factorizable, e.g., $B = I$, so that we can multiply vectors by

AB^{-1}, or $B^{-1}A$, we take $\nu = 0$. In this case, a single iteration may not be expensive, but eigenvalues μ close to zero are usually not of practical interest, and the convergence for eigenvalues of interest is often very slow.

Thus, the traditional approach is inefficient for very large mesh eigenproblems. Preconditioning is the key for significant improvement of the performance as it allows one to find a path between the Scylla of expensive factorizations and the Charybdis of slow convergence.

Preconditioned methods are designed to handle the case when the only operation we can perform with matrices A and B of the pencil is multiplication of a vector by A and B. To accelerate the convergence, we introduce a *preconditioner* T.

In many engineering applications, preconditioned iterative solvers for linear systems $Ax = b$ are already available, and efficient preconditioners $T \approx A^{-1}$ are constructed. It is important to realize that the same preconditioner T can be used to solve an eigenvalue problem $Ax = \lambda x$, or $Bx = \mu Ax$.

We assume that the preconditioner T is *symmetric positive definite.* As A is also symmetric positive definite, there exist positive constants $\delta_1 \geq \delta_0 > 0$ such that

$$\delta_0(T^{-1}x, x) \leq (Ax, x) \leq \delta_1(T^{-1}x, x). \tag{2.1}$$

The ratio δ_1/δ_0 can be viewed as the spectral condition number $\kappa(TA)$ of the preconditioned matrix TA and measures how well the preconditioner T approximates, up to a scaling, the matrix A^{-1}. A smaller ratio δ_1/δ_0 usually ensures faster convergence.

We want to highlight that the assumption we just made on T is essential for the theory (see [21]) but may not be an actual limitation in numerical computations for some methods. In particular, our own method of section 5 is quite robust in practice with respect to the choice of the preconditioner, even when the assumptions above are not satisfied; see Figure 5.2 below as an example of using an indefinite preconditioner.

As we want to discuss an optimality of preconditioned eigensolvers, we need to have a formal definition of the whole class of such methods. We first define, following [21], a preconditioned single-vector iterative solver for the pencil $B - \mu A$, as a generalized polynomial method of the following kind:

$$x^{(k)} = P_k(TA, TB)x^{(0)}, \tag{2.2}$$

where P_k is a polynomial of the kth degree of two independent variables, $x^{(0)}$ is an initial guess, and T is a fixed preconditioner.

We need only choose a polynomial, either a priori or during the process of iterations, and use a recursive formula which leads to an iterative scheme. For an approximation $\mu^{(i)}$ $(\lambda^{(i)})$ to an eigenvalue of the pencil $B - \mu A$ $(A - \lambda B)$ for a given eigenvector approximation $x^{(i)}$ the Rayleigh quotient $\mu(x)$ $(\lambda(x))$ defined as

$$\mu(x^{(i)}) = \frac{(x^{(i)}, Bx^{(i)})}{(x^{(i)}, Ax^{(i)})} \quad \left(\lambda(x^{(i)}) = \frac{(x^{(i)}, Ax^{(i)})}{(x^{(i)}, Bx^{(i)})}\right) \tag{2.3}$$

is typically used.

Let us now define the generalized Krylov subspace:

$$K_k\left(TA, TB, x^{(0)}\right) = \left\{P_k(TA, TB)x^{(0)}\right\}, \tag{2.4}$$

where P_k runs through the set of all polynomials of the kth degree of two independent variables and $x^{(0)}$ is a fixed initial vector. In particular,

$$\begin{aligned} &K_2\left(TA, TB, x^{(0)}\right) \\ &\quad = \operatorname{span}\left\{x^{(0)}, TAx^{(0)}, TBx^{(0)}, (TA)^2x^{(0)}, TATBx^{(0)}, TBTAx^{(0)}, (TB)^2x^{(0)}\right\}. \end{aligned}$$

We notice that in our definition (2.2) of the preconditioned eigensolver

$$x^{(k)} \in K_k\left(TA, TB, x^{(0)}\right).$$

Having definition (2.2) of the whole class of preconditioned eigensolvers, we can introduce, as in [21], the *global optimization* method for computing the first eigenpair simply by maximizing the Rayleigh quotient $\mu(x)$ on the Krylov subspace (2.4):

$$x_o^{(k)} = \operatorname{argmax}_{x \in K_k\left(TA, TB, x^{(0)}\right)} \mu(x). \tag{2.5}$$

We want to highlight that an efficient algorithm for finding $x_o^{(k)}$, e.g., based on short-term recurrences, is not presently known, and that the number of vectors in the basis of the Krylov subspace (2.4) grows exponentially, which makes the method very expensive in practice, similarly to the situation with Davidson's method (see discussion in [21, 22]), unless restarts are used. Therefore, the global optimization method (2.5) is optimal only in the sense that it provides the global maximum of the Rayleigh quotient on the Krylov subspace, but it may not be optimal if we also count computational costs.

For block methods, we introduce the generalized block Krylov subspace:

$$K_k\left(TA, TB, X^{(0)}\right) = \operatorname{span}\left\{P_k(TA, TB)x_j^{(0)},\ j = 1, \dots, m\right\}, \tag{2.6}$$

where P_k runs through the set of all polynomials of the kth degree of two independent variables and $X^{(0)} = \operatorname{span}\{x_j^{(0)},\ j = 1, \dots, m\}$.

A general preconditioned block eigensolver is a generalization of method (2.2) with a single vector being replaced with several ones. Using the Rayleigh–Ritz method is typical for block methods; see [21, 22].

Here, we only want to define the *block global optimization* method, Algorithm 2.1, as we use it later in our numerical experiments.

```
ALGORITHM 2.1. THE BLOCK GLOBALLY OPTIMAL PRECONDITIONED EIGENSOLVER.
Input: $m$ starting vectors $x_1^{(0)}, \dots x_m^{(0)}$, devices to compute: $Ax$, $Bx$, and $Tx$ for
    a given vector $x$, and the vector inner product $(x, y)$.
  1. Start: select $x_j^{(0)}$, $j = 1, \dots, m$.
  2. Iterate to compute the basis of the generalized block Krylov subspace (2.6).
  3. Use the Rayleigh–Ritz method for the pencil $B - \mu A$ in the subspace
       to compute the Ritz values $\mu_j^{(k)}$ and the corresponding Ritz vectors $x_j^{(k)}$.
Output: the approximations $\mu_j^{(k)}$ and $x_j^{(k)}$ to the largest eigenvalues
    $\mu_j$ and corresponding eigenvectors, $j = 1, \dots, m$.
```

In our code of the block global optimization method, we do not even try to minimize computation costs and simply compute recursively

$$K_{k+1} = K_k + TAK_k + TBK_k, \tag{2.7}$$

followed by complete orthogonalization. The only purpose of the code is to provide a comparison, in terms of accuracy, for the actual block method we suggest in section 5.

3. The "ideal" preconditioned conjugate gradient method.

In this section, we outline the "ideal" control algorithm, namely, the standard preconditioned conjugate gradient (PCG) method for finding an eigenvector, corresponding to the minimal eigenvalue, as an element of the null-space of the corresponding homogeneous system of linear equations under the assumption that the eigenvalue is known.

We assume $B > 0$ in this section. Let us *suppose* that the minimal eigenvalue λ_1 is *already known*, and we just need to compute the corresponding eigenvector x_1, an element of the null-space of the homogeneous system of linear equations

$$(A - \lambda_1 B)x_1 = 0,$$

where the matrix of the system is symmetric and nonnegative definite. What would be an *ideal* preconditioned method of computing x_1? As such, we choose the *standard PCG method.* It is well known that a PCG method can be used to compute a nonzero element of the null-space of a homogeneous system of linear equations with a symmetric and nonnegative definite matrix if a nonzero initial guess is used. While fitting perfectly the definition of a single-vector preconditioned eigensolver of the previous section, this ideal method *cannot be used in practice* as it requires knowledge of the eigenvalue.

We suggest using the method as a *control* in numerical comparison with practical eigenvalue solvers, in particular, with PCG eigensolvers, e.g., with our locally optimal PCG method. If an eigensolver finds the eigenvector u_1 with the same accuracy and costs as the ideal method, we have reasons to call such an eigensolver "optimal" for computing the eigenvector u_1.

For our numerical tests, we write a code PCGNULL.m, which is a slightly modified version of the built-in MATLAB code PCG.m, revision 1.11, of the standard PCG method to cover solution of homogeneous systems of linear equations with symmetric and nonnegative definite system matrices.

The standard theory of the PCG method for computing an element of the null-space of a symmetric nonnegative definite matrix implies convergence to the eigenvector x_1, which is the T^{-1}-orthogonal projection of the initial guess $y^{(0)}$ to the null-space of the matrix $A - \lambda_1 B$. On the $(i+1)$st step, the "energy" norm of the error, in our case it's actually the seminorm based on $A - \lambda_1 B$, i.e.,

$$\sqrt{(y^{(i+1)}, (A - \lambda_1 B)y^{(i+1)})}, \tag{3.1}$$

is minimized over the hyperplane

$$H_{i+1} = y^{(0)} + K_i(T(A - \lambda_1 B),\ T(A - \lambda_1 B)y^{(0)}), \tag{3.2}$$

where K_i is the standard Krylov subspace. As x_1 is in the null-space of the matrix $A - \lambda_1 B$, we have

$$(y, T^{-1}x_1) = (y^{(0)}, T^{-1}x_1) \quad \forall y \in H_i, \tag{3.3}$$

in particular, $(y^{(i)}, T^{-1}x_1)$ does not change in the process of iterations, while (3.1) converges to zero at least linearly with the asymptotic average convergence factor $(1-\sqrt{\xi})/(1+\sqrt{\xi})$, where ξ is an upper bound for the ratio of the largest and smallest nonzero eigenvalues of the preconditioned matrix $T(A - \lambda_1 B)$. Using estimates of [18] of eigenvalues of $T(A - \lambda_1 B)$ in terms of eigenvalues λ of the pencil $A - \lambda B$ and constants of (2.1), we can obtain such an upper bound and get the following upper bound of the asymptotic average convergence factor:

$$q = \left(\frac{1 - \sqrt{\xi}}{1 + \sqrt{\xi}}\right), \ \xi = \frac{1}{\kappa(TA)}\left(1 - \frac{\lambda_1}{\lambda_2}\right). \tag{3.4}$$

Finally, we would like to remind the reader of a long forgotten version of the PCG method based on optimization of a three-term recurrence, e.g., [35]:

$$y^{(i+1)} = y^{(i)} + \alpha^{(i)} v^{(i)} + \beta^{(i)}(y^{(i)} - y^{(i-1)}), \ v^{(i)} = T(A - \lambda_1 B)y^{(i)}, \ \beta^{(0)} = 0, \tag{3.5}$$

with both scalar parameters $\alpha^{(i)}$ and $\beta^{(i)}$ computed by minimizing seminorm (3.1) of $y^{(i+1)}$. This version is mathematically equivalent, in exact arithmetic, to the standard version implemented in PCGNULL, which uses two linked two-term recurrences.

This provides an insight into the locally optimal PCG eigensolver [19] we discuss in the next section, where we simply replace in (3.5) exact λ_1 with its approximation, and instead of minimizing seminorm (3.1) of $y^{(i+1)}$ we compute $y^{(i+1)}$ by using the Rayleigh–Ritz method on the subspace

$$y^{(i+1)} \in \text{Span}\ \left\{v^{(i)},\ y^{(i)},\ y^{(i-1)}\right\}. \tag{3.6}$$

An investigation of a possible connection between the two methods is in progress. As we see in section 7, they behave quite similarly in our numerical tests.

4. The PCG methods.

In this section, we propose a new version of the *locally optimal PCG method* [19].

In [19], the author suggested the following PCG method for the pencil $B - \mu A$:

$$x^{(i+1)} = w^{(i)} + \tau^{(i)} x^{(i)} + \gamma^{(i)} x^{(i-1)}, \ w^{(i)} = T(Bx^{(i)} - \mu^{(i)} Ax^{(i)}), \ \mu^{(i)} = \mu(x^{(i)}), \ \gamma^{(0)} = 0, \tag{4.1}$$

with scalar iteration parameters $\tau^{(i)}$ and $\gamma^{(i)}$ chosen using an idea of *local optimality* [19], namely, select $\tau^{(i)}$ and $\gamma^{(i)}$ that maximize the Rayleigh quotient $\mu(x^{(i+1)})$ by using the Rayleigh–Ritz method. As the current eigenvector approximation $x^{(i)}$ and the previous eigenvector approximation $x^{(i-1)}$ are getting closer to each other in the process of iterations, special measures need to be used in the algorithm to overcome the potential instability.

Formula (4.1) has been used in an earlier revision of our MATLAB code LOBPCG. In our numerical tests, it often led to so ill-conditioned Gram matrices that the Rayleigh–Ritz method would produce spurious eigenpairs. As a cure, we had to use an A-based orthogonalization of the three-dimensional trial subspace, which increased computational costs as had to multiply by A more often.

In our new algorithm, the three-term recurrence contains the current eigenvector approximation, the current preconditioned residual, and the implicitly computed difference between the current and the previous eigenvector approximations:

$$\begin{aligned} x^{(i+1)} &= w^{(i)} + \tau^{(i)} x^{(i)} + \gamma^{(i)} p^{(i)}, \quad w^{(i)} = T(Bx^{(i)} - \mu^{(i)} Ax^{(i)}), \\ p^{(i+1)} &= w^{(i)} + \gamma^{(i)} p^{(i)}, \quad p^{(0)} = 0, \quad \mu^{(i)} = \mu(x^{(i)}), \end{aligned} \tag{4.2}$$

with scalar iteration parameters $\tau^{(i)}$ and $\gamma^{(i)}$ chosen using the idea of *local optimality* as above, namely, select $\tau^{(i)}$ and $\gamma^{(i)}$ that maximize the Rayleigh quotient $\mu(x^{(i+1)})$

by using the Rayleigh–Ritz method. We see that

$$p^{(i+1)} = x^{(i+1)} - \tau^{(i)} x^{(i)};$$

thus,

$$x^{(i+1)} \in \text{Span}\,\{w^{(i)},\ x^{(i)},\ p^{(i)}\} = \text{Span}\,\{w^{(i)},\ x^{(i)},\ x^{(i-1)}\},$$

and therefore the new formula (4.2) is mathematically equivalent to the previous one, (4.1), in exact arithmetic.

We describe the actual algorithm of the method given by (4.2) as Algorithm 4.1.

Our experiments confirm that Algorithm 4.1 is much more numerically stable compared to the previous version (4.1) and that it can be used without extra A-orthogonalization in most situations. However, for ill-conditioned problems and when a high accuracy is required, even our new choice of the basis $w^{(i)}, x^{(i)}, p^{(i)}$ of the trial subspace of the Rayleigh–Ritz method may lead to ill-conditioned 3-by-3 Gram matrices, which makes necessary orthogonalization prior to the use of the Rayleigh–Ritz method. In the actual code of LOBPCG, we check for ill-conditioned Gram matrices on every iteration and implement A-orthogonalization if necessary. Since by our assumptions matrix B may not be positive definite, there is no other option except to use the A-based scalar product for the orthogonalization. This typically increases the cost of iterations, but it makes the algorithm more robust.

Algorithm 4.1. The locally optimal PCG method.

```
Input: devices to compute: $Ax$, $Bx$, and $Tx$ for a given vector $x$, the vector inner
    product $(x, y)$, and a starting vector $x^{(0)}$.
 1. Start: select $x^{(0)}$ and set $p^{(0)} = 0$.
 2. Iterate: For $i = 0, \ldots,$ Until Convergence Do:
 3.     $\mu^{(i)} := (x^{(i)}, B\,x^{(i)})/(x^{(i)}, A\,x^{(i)})$
 4.     $r := B\,x^{(i)} - \mu^{(i)} A x^{(i)}$
 5.     $w^{(i)} := Tr$
 6.     Use the Rayleigh–Ritz method for the pencil $B - \mu A$
        on the trial subspace Span $\{w^{(i)}, x^{(i)}, p^{(i)}\}$
 7.     $x^{(i+1)} := w^{(i)} + \tau^{(i)} x^{(i)} + \gamma^{(i)} p^{(i)}$,
        (the Ritz vector corresponding to the maximal Ritz value)
 8.     $p^{(i+1)} := w^{(i)} + \gamma^{(i)} p^{(i)}$
 9. EndDo
Output: the approximations $\mu^{(k)}$ and $x^{(k)}$ to the largest eigenvalue $\mu_1$ and its
    corresponding eigenvector.
```

We want to highlight that the algorithm can be implemented with only one application of the preconditioner T, one matrix-vector product Bx, and one matrix-vector product Ax, per iteration.

Storage requirements for Algorithm 4.1 are small—only several n-vectors and no n-by-n matrices at all. Such methods are sometimes called *matrix-free*.

For the stopping criterion, we compute some norms of the preconditioned residual $w^{(i)}$ on every iteration. Such norms may provide accurate two-sided bounds for eigenvalues and a posteriori error bounds for eigenvectors; see [18]. For brevity, we do not consider it in the present paper.

Let us also mention the possibility of avoiding the residual in method (4.1) by splitting it into two vectors:

$$x^{(i+1)} \in \text{Span}\ \left\{TAx^{(i)}, TBx^{(i)}, x^{(i)}, x^{(i-1)}\right\}. \tag{4.3}$$

In this new method, the trial subspace is enlarged, which may lead to somewhat faster convergence. However, we need to apply the preconditioner two times now on every iteration; therefore, in our opinion, method (4.3) will not be more efficient for computing the extreme eigenpair than our favorite method, Algorithm 4.1, since the latter already converges practically with the optimal speed; see sections 7 and 8.

Other known CG methods for eigenproblems, e.g., [41, 12, 15, 11, 42, 13, 1, 26], starting from Bradbury and Fletcher [3], are usually constructed as general CG minimization methods, applied to the Rayleigh quotient or to a quadratic form (x, Bx) under the constrain $(x, Ax) = 1$. They are often based on (now standard for linear systems) two linked two-term recurrences,

$$p^{(i)} = -w^{(i)} + \beta^{(i)}p^{(i-1)},\ x^{(i+1)} = x^{(i)} + \alpha^{(i)}p^{(i)},$$

where $\alpha^{(i)}$ is chosen using a line search to minimize the Rayleigh quotient of $x^{(i+1)}$, which leads to a quadratic equation for $\alpha^{(i)}$, but $\beta^{(i)}$ is computed to make directions $p^{(i)}$ to be conjugate in some sense. These methods *do not utilize* the specific property of the Rayleigh quotient, i.e., that the local minimization of the Rayleigh quotient can be cheaply carried out using the Rayleigh–Ritz method not just in two dimensions, for line search for finding $\alpha^{(i)}$, but in three-dimensional or larger dimensional subspaces as well.

Let us now discuss theoretical convergence rate results for Algorithm 4.1.

The basic fact is that the locally optimal version of the PCG method trivially converges not slower on every step than the preconditioned steepest ascent in terms of the maximizing the Rayleigh quotient [19]; thus, we can use known and well-developed convergence theory of the latter method (e.g., [19]; see also very recent results by Klaus Neymeyr [29, 30]). Our numerical comparison in [21, 22] shows, however, that the PCG method converges much faster in practice than the preconditioned steepest ascent. A ten-fold increase of δ_1/δ_0 leads to the increase in number of iterations, ten-fold for the preconditioned steepest ascent, but only about three-fold for the PCG method, exactly as we would expect for a genuine PCG solver.

No comprehensive convergence theory that would explain such a fast convergence is available yet. Moreover, no even similar results are apparently known at present, e.g., there is no adequate convergence theory of CG methods in nonquadratic optimization. Even if one considers the simplest version of (4.1) for the standard eigenproblem and with no preconditioning, $T = A = I$ when it is reduced to the following trivial method, suggested in [17, 18, 25]: find

$$x^{(i+1)} \in \text{Span}\ \left\{Bx^{(i)}, x^{(i)}, x^{(i-1)}\right\} \tag{4.4}$$

by maximizing the Rayleigh quotient $(x, Bx)/(x, x)$ on every step. There is still no convergence theory currently known that would be able to compare (4.4) with the optimal method, in this case, with the Lanczos method for the global maximization of Rayleigh quotient $(x, Bx)/(x, x)$ on the corresponding standard Krylov subspace.

One possible approach to develop an adequate convergence theory may be based on comparison of method (4.4) with the following stationary three-term recurrence:

$$x^{(i+1)} = \alpha Bx^{(i)} + \beta x^{(i)} + \gamma x^{(i-1)},$$

where α, β, γ are fixed scalar parameters, sometimes called the *heavy ball method* in optimization [35]. However, for this simpler method, no accurate convergence theory in terms of the Rayleigh quotient apparently exists yet; cf. [25].

In this paper, we do not prove any new theoretical convergence rate results for Algorithm 4.1, but we suggest a different kind of remedy: numerical comparisons using the benchmark routines we propose; see sections 7 and 8.

We present block versions of Algorithm 4.1 in the next section.

5. Preconditioned simultaneous iterations. The well-known idea of using *simultaneous*, or *block*, iterations provides an important improvement over single-vector methods, and permits us to compute an $(m > 1)$-dimensional invariant subspace, rather than one eigenvector at a time. It can also serve as an acceleration technique over single-vector methods on parallel computers, as convergence for extreme eigenvalues usually increases with the size of the block, and every step can be naturally implemented on wide varieties of multiprocessor computers.

As in other block methods, the block size should be chosen, if possible, to provide a large gap between first m eigenvalues and the rest of the spectrum as this typically leads to a better convergence; see (5.5) below. Let us also mention that block methods generally handle clusters in the spectrum and multiple eigenvalues quite well; and the block methods we propose below are no exceptions. An attempt should be made to include the whole cluster of eigenvalues into the block, while for multiple eigenvalues this is not essential at all; e.g., if $\mu_{m-1} > \mu_m = \mu_{m+1} > \mu_{m+2}$, then the convergence rate will be determined by the gap $\mu_{m+1} - \mu_{m+2}$ even though the block size is only m; however, only one vector of the two-dimensional eigenspace corresponding to $\mu_m = \mu_{m+1}$ will be computed, as we observe in numerical experiments.

A block version of the locally optimal PCG method [19] was suggested in [21, 22]:

$$x_j^{(i+1)} \in \text{Span } \left\{x_1^{(i-1)},\ x_1^{(i)},\ T(B - \mu_1^{(i)} A)x_1^{(i)}, \dots,\ x_m^{(i-1)},\ x_m^{(i)},\ T(B - \mu_m^{(i)} A)x_m^{(i)}\right\}, \tag{5.1}$$

where $x_j^{(i+1)}$ is computed as the jth Ritz vector. It shares the same problem, as the single-vector version of [19] discussed in the previous section, of having close vectors in the trial subspace.

Our new block Algorithm 5.1 is a straightforward generalization of the single-vector Algorithm 4.1 and is combined with the Rayleigh–Ritz procedure. Here we present two different variants of the algorithm. They differ in the way that the Rayleigh–Ritz method is used. The first version is mathematically equivalent (without round-off errors) to that of [21, 22], but uses directions $p_j^{(i)}$ instead of $x_j^{(i-1)}$, similar to that of Algorithm 4.1.

Here, the column-vector

$$\left(\alpha_1^{(i)}, \dots, \alpha_m^{(i)},\ \tau_1^{(i)}, \dots, \tau_m^{(i)}\ \gamma_1^{(i)}, \dots, \gamma_m^{(i)}\right)^T$$

is the jth eigenvector corresponding to the jth largest eigenvalue of the $3m$-by-$3m$ eigenvalue problem of the Rayleigh–Ritz method, so it should have had an index j also, but we run out of space for indexes.

We observe that

$$p_j^{(i+1)} = x_j^{(i+1)} - \sum_{k=1,\dots,m} \tau_k^{(i)} x_k^{(i)},$$

```
ALGORITHM 5.1. THE LOBPCG METHOD I.
Input: $m$ starting vectors $x_1^{(0)}, \dots x_m^{(0)}$, devices to compute: $Ax$, $Bx$, and $Tx$
    for a given vector $x$, and the vector inner product $(x, y)$.
Start: select $x_j^{(0)}$, and set $p_j^{(0)} = 0,\ j = 1, \dots, m$.
Iterate: For $i = 0, \dots,$ Until Convergence Do:
    $\mu_j^{(i)} := (x_j^{(i)}, B\,x_j^{(i)})/(x_j^{(i)}, A\,x_j^{(i)}),\ j = 1, \dots, m$;
    $r_j := B\,x_j^{(i)} - \mu_j^{(i)} A x_j^{(i)},\ j = 1, \dots, m$;
    $w_j^{(i)} := T r_j,\ j = 1, \dots, m$;
    Use the Rayleigh–Ritz method for the pencil $B - \mu A$ on the trial subspace
    Span $\{w_1^{(i)}, \dots, w_m^{(i)}, x_1^{(i)}, \dots, x_m^{(i)}, p_1^{(i)}, \dots, p_m^{(i)}\}$;
    $x_j^{(i+1)} := \sum_{k=1,\dots,m} \alpha_k^{(i)} w_k^{(i)} + \tau_k^{(i)} x_k^{(i)} + \gamma_k^{(i)} p_k^{(i)}$,
    (the $j$th Ritz vector corresponding to the $j$th largest Ritz value),
    $j = 1, \dots, m$;
    $p_j^{(i+1)} := \sum_{k=1,\dots,m} \alpha_k^{(i)} w_k^{(i)} + \gamma_k^{(i)} p_k^{(i)}$;
EndDo
Output: the approximations $\mu_j^{(k)}$ and $x_j^{(k)}$ to the largest eigenvalues $\mu_j$ and
    corresponding eigenvectors, $j = 1, \dots, m$.
```

and thus,

$$\begin{aligned} x_j^{(i+1)} &\in \text{Span}\,\{w_1^{(i)}, \dots, w_m^{(i)}, x_1^{(i)}, \dots, x_m^{(i)}, p_1^{(i)}, \dots, p_m^{(i)}\} \\ &= \text{Span}\,\{w_1^{(i)}, \dots, w_m^{(i)}, x_1^{(i)}, \dots, x_m^{(i)}, x_1^{(i-1)}, \dots, x_m^{(i-1)}\}, \end{aligned}$$

and, indeed, the new Algorithm 5.1 is mathematically equivalent to method (5.1).

We note that Algorithm 5.1 without preconditioning, i.e., with $T = I$, appears in [25] as a "W-accelerated simultaneous gradient method."

Example 5.1. Let us consider the problem of computing first eigenpairs of the standard five-point finite-difference approximation of the Laplacian Δ_h in a $[-1, 1] \times [-1, 1]$ square on a uniform mesh with the step $h = 1/10$, such that the total number of the unknowns is 361. We set $B = I$ and $A = \Delta_h$. The initial approximations are random. No preconditioning is used in the first test, i.e., $T = I$. We plot in Figure 5.1 on a semilog scale relative errors in this example defined as $\|Ax_j^{(i)} - \lambda_j^{(i)} x_j^{(i)}\|_A / \|x_j^{(i)}\|_A$, where we use the standard notation of an A-based norm, $\|\cdot\|_A^2 = (\cdot, A\cdot)$.

Figure 5.1 shows a clear superlinear convergence. This is quite important as some authors, e.g., [34], consider a superlinear convergence as a sign of a genuine CG method. We see that a larger block size in the method improves convergence in this example. We also observe different convergence speed for different eigenpairs on Figure 5.1—the convergence is faster for smaller eigenvalues λ, in particular, λ_1 converges first.

In the second test, we use a preconditioner $T = A - \nu I$ with the shift $\nu = 20$, which is approximately in the middle of the group of the first ten eigenvalues that we compute, starting with random initial approximations. We plot $\|Ax_j^{(i)} - \lambda_j^{(i)} x_j^{(i)}\| / \|x_j^{(i)}\|$, using the Euclidean norm, in Figure 5.2 on a semilog scale. Figure 5.2 shows a superior convergence of the preconditioning. Now, we have a better convergence for eigenvalues close to the shift on the right part of Figure 5.2.

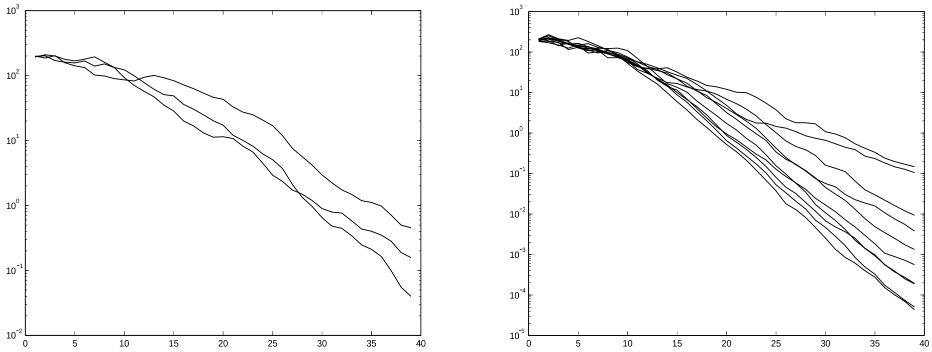


FIG. 5.1. *Errors for the Laplace operator in a square without preconditioning for three (left) and ten (right) first eigenpairs.*

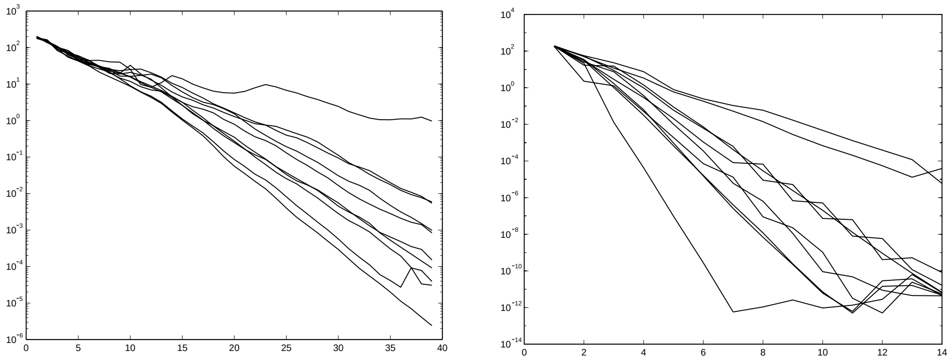


FIG. 5.2. *Errors for the Laplacian operator in a square without preconditioning (left) and with an indefinite preconditioner(right).*

The example thus illustrates that our method works without preconditioning and may even work with an indefinite preconditioner.

Example 5.2. As our next example, we solve a similar eigenvalue problem, but in the L-shaped domain—a union of three unit squares with a mesh uniform in both directions with the step 1/8, such that the total number of unknowns is 161. A specialized domain-decomposition without overlap method for such problem is suggested in [23]. Our numerically computed eigenvalues are consistent with those found in [23]. We compare the performance of the method without preconditioning and with preconditioning based on an incomplete Choleski decomposition of A with a drop tolerance of 10^{-3}. We plot $\|Ax_j^{(i)} - \lambda_j^{(i)} x_j^{(i)}\| / \|x_j^{(i)}\|$ in Figure 5.3. We see that preconditioning leads to approximately quadruple acceleration.

The actual MATLAB code LOBPCG of Algorithm 5.1 that we wrote uses the basis of the trial subspace exactly the way it appears in Algorithm 5.1 until this choice leads to ill-conditioned Gram matrices in the Rayleigh–Ritz method. When such an ill-conditioning shows up, we perform a selected orthogonalization. If this does not fix the problem, as a last resort we apply a complete orthogonalization prior to the Rayleigh–Ritz method. By our assumptions, A may be the only positive definite

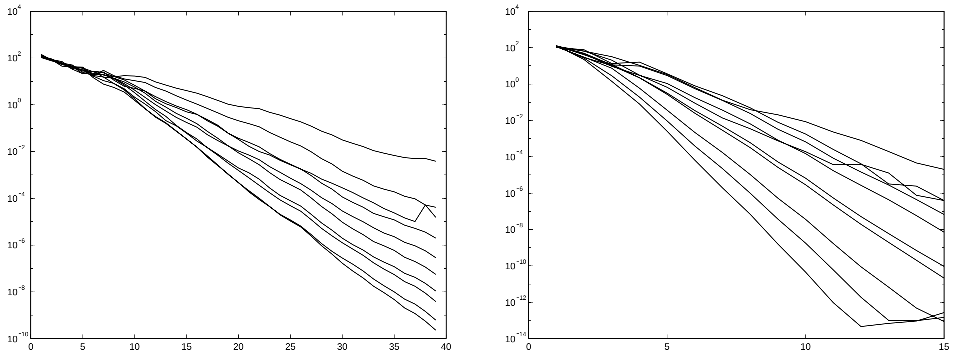


FIG. 5.3. *Residuals for the Laplacian operator in the L-shaped domain with no preconditioning (left) and an ILU preconditioning (right).*

matrix of the pencil; thus, the A-based scalar product is used for the orthogonalization.

There is no theory available yet to predict accurately the speed of convergence of Algorithm 5.1. Similarly to the single-vector case of the previous section, we can easily see that Algorithm 5.1 does not converge slower than the block steepest ascent on every step. Unfortunately, earlier known convergence results [9, 10, 4] for block preconditioned eigensolvers cannot be used to take advantage of this fact. Only the very recent result by Neymeyr [28] for the block preconditioned simple iterations allows us to conclude that Algorithm 5.1 converges linearly with the average asymptotic convergence factors

$$q_j = 1 - \frac{2}{\kappa(TA)+1}\frac{\mu_j - \mu_{j+1}}{\mu_j - \mu_{\min}},\ j = 1,\dots,m. \tag{5.2}$$

Let us formulate and prove this as the following.

THEOREM 5.1. *For simplicity, let*

$$\mu_j > \mu_j^{(i)} > \mu_{j+1},\ j = 1,\dots,m.$$

Then

$$\frac{\mu_j - \mu_j^{(i+1)}}{\mu_j^{(i+1)} - \mu_{j+1}} \le q_j^2 \frac{\mu_j - \mu_j^{(i)}}{\mu_j^{(i)} - \mu_{j+1}}, \tag{5.3}$$

with q_j given by (5.2).

Proof. The proof is straightforward and is based on a possibility of using the convergence rate estimate of [28] recursively and on the fact that our Algorithm 5.1 does not converge slower than the method of [28] in terms of Ritz values as our trial subspace is enlarged compared to that of [28].

The first step is to notice that all results of [28], written for the pencil $A - \lambda B$ with $B = I$, can be trivially applied to a more general case of $B \neq I,\ B > 0$ by using substitutions involving $B^{1/2}$. Secondly, we take the main convergence result of [28], presented as a nasty looking inequality for $\lambda_j^{(i+1)}$, and after tedious but elementary algebraic manipulations we show that it can be reduced to (5.3) with

$$q_j = 1 - \frac{2}{\kappa(TA)+1}\left(1 - \frac{\lambda_j}{\lambda_{j+1}}\right) = 1 - \frac{2}{\kappa(TA)+1}\frac{\mu_j - \mu_{j+1}}{\mu_j}. \tag{5.4}$$

Here, T is optimally scaled such that $\|I - TA\|_A \leq (\kappa(TA) - 1)/(\kappa(TA) + 1)$ to be consistent with assumptions of [28]. Neymeyr has recently found (private communication) an easier proof that takes only two pages.

Now, we use a trick, suggested in [17] and reproduced in [8]; namely, we substitute our actual matrix B, which is not necessarily positive definite with positive definite $B_\alpha = B - \alpha A > 0$ with a scalar $\alpha < \mu_{\min}$ and apply the previous estimate to the pencil $B_\alpha - \mu_\alpha A$ with eigenvalues $\mu_\alpha = \mu - \alpha$, which gives (5.3) with

$$q_j = 1 - \frac{2}{\kappa(TA) + 1} \frac{\mu_j - \mu_{j+1}}{\mu_j - \alpha}.$$

Finally, we realize that the block method of [28], which is the same as that of [4], itself is invariant with respect to α, and everything depends continuously on $\alpha < \mu_{\min}$, so we can take the limit $\alpha = \mu_{\min}$ as well. This proves estimate (5.3) with q_j given by (5.2) for the block method used in [4, 28] for the general pencil $B - \mu A$ satisfying assumptions of the present paper. However, in Algorithm 5.1 the trial subspace is enlarged compared to that of the method of [4, 28]; thus, the convergence rate estimate (5.3) with q_j given by (5.2) holds for our method, too. □

This result, however, does not appear to be sharp for our method, while it is sharp for the method of [28] in a certain sense.

First, if the computed eigenvalues form, or include, a cluster, the factor given by (5.2) is quite pessimistic as it depends on $\mu_j - \mu_{j+1}$, which may be small. For a block method, we expect to have a term $\mu_j - \mu_{m+1}$ instead, where m is the block size.

Second, for a genuine CG method we should count on having $\sqrt{\kappa(TA)}$ instead of $\kappa(TA)$ in the expression for q.

In [21, 22], we demonstrate numerically that our method is much faster than that of [4, 28]. Having in mind estimate (3.4) we have for the PCGNULL and results of [20] on the Rayleigh–Ritz method used in [4], which informally speaking allows us to analyze the error in jth Ritz vector $x_j^{(i)}$ ignoring components $(x_j^{(i)}, x_k)_A,\ k = 1, \dots, m,\ k \neq j$, we should expect convergence of norms of residuals to be asymptotically linear with the average asymptotic convergence factors

$$q_j = \left(\frac{1 - \sqrt{\xi_j}}{1 + \sqrt{\xi_j}} \right),\ \xi_j = \frac{1}{\kappa(TA)} \frac{\mu_j - \mu_{m+1}}{\mu_j - \mu_{\min}},\ j = 1, \dots, m. \tag{5.5}$$

Thus, convergence of $\mu_j^{(i)}$ to μ_j should be linear with the ratio q_j^2.

All our numerical tests (see some selected results in sections 7 and 8 below) support our expectation of having asymptotically linear convergence with the ratio (5.5).

A potential disadvantage of Algorithm 5.1 can manifest itself when the number m of eigenpairs of interest is large, as we need to form $3m$-by-$3m$ matrices and solve the corresponding eigenvalue problem of the size $3m$ in the Rayleigh–Ritz method on every step. It seems that vectors $w_j^{(i)}$ and $p_j^{(i)}$ may not always be helpful as basis vectors of the trial subspace of the Rayleigh–Ritz method to improve the approximation of $x_k^{(i+1)}$ for $j \neq k$. And adding unnecessary vectors in the trial subspace increases computational costs. Even more importantly, it may make the algorithm less stable, as the $3m$-by-$3m$ eigenvalue problem of the Rayleigh–Ritz method is likely to inherit ill-conditioning of the original eigenvalue problem when m is large.

In our second variant, Algorithm 5.2, we apply the Rayleigh–Ritz method in two stages: first, as in Algorithm 4.1, for individual indices j, and, second, we include only the minimal set of vectors, namely, just current approximations to different eigenvectors into the trial subspace.

Thus, on the first stage of an iteration of Algorithm 5.2 we solve m three-dimensional eigenproblems, and on the second stage we solve one m-dimensional eigenproblem. But this m-dimensional eigenproblem is constructed using approximate eigenvectors, corresponding to extreme eigenvalues, as a basis of the trial subspace. Therefore, this eigenvalue problem should not be ill-conditioned and no orthogonalization would be required.

We note that arguments of the previous theorem cannot be applied to Algorithm 5.2. Theoretical investigation of accurate convergence estimates of Algorithm 5.2 does not seem to be a trivial exercise.

Our numerical comparison using model problems described in section 6 below shows that Algorithm 5.2 converges with practically the same speed as Algorithm 5.1, except for the case of finding a cluster of eigenvalues with high accuracy. For eigenvalues in a cluster, Algorithm 5.2 at first converges similarly to Algorithm 5.1, but it then slows down and soon after may hit a plateau, i.e., may stop improving the quality of approximations. See Figure 5.4, which shows convergence history of eigenvalue errors of a one run of Algorithms 5.1 (solid) and 5.2 (dotted) with block size $m = 3$. The left picture is for well-separated eigenvalues, and errors for different algorithms are not possible to distinguish. The right picture corresponds to the cluster of three eigenvalues with $(\mu_1 - \mu_3)/(\mu_1 - \mu_{\text{min}}) \approx 10^{-11}$.

```
ALGORITHM 5.2. THE LOBPCG METHOD II.
Input: $m$ starting vectors $x_1^{(0)}, \dots x_m^{(0)}$,devices to compute: $Ax$, $Bx$, and $Tx$ for a
    given vector $x$,and the vector inner product $(x, y)$,
  Start: select $x_j^{(0)}$, and set $p_j^{(0)} = 0,\ j = 1, \dots, m$.
  Iterate: For $i = 0, \dots,$ Until Convergence Do:
      $\mu_j^{(i)} := (x_j^{(i)}, B x_j^{(i)})/(x_j^{(i)}, A x_j^{(i)}),\ j = 1, \dots, m$;
      $r_j := B x_j^{(i)} - \mu_j^{(i)} A x_j^{(i)},\ j = 1, \dots, m$;
      $w_j^{(i)} := T r_j,\ j = 1, \dots, m$;
      Use the Rayleigh–Ritz method $m$ times for the pencil $B - \mu A$ on the trial
        subspaces Span $\{w_j^{(i)},\ x_j^{(i)},\ p_j^{(i)}\},\ j = 1, \dots, m$;
      $\widehat{x}_j^{(i+1)} := w_j^{(i)} + \tau_j^{(i)} x_j^{(i)} + \gamma_j^{(i)} p_j^{(i)}$,
        (the Ritz vector corresponding to the maximal Ritz value), $j = 1, \dots, m$;
      $p_j^{(i+1)} := w_j^{(i)} + \gamma_j^{(i)} p_j^{(i)}$;
      Use the Rayleigh–Ritz method for the pencil $B - \mu A$ on the trial
        subspaces $\{\widehat{x}_1^{(i+1)}, \dots, \widehat{x}_m^{(i+1)}\}$;
      $x_j^{(i+1)} :=$ the $j$th Ritz vector corresponding to the $j$th largest Ritz value,
        $j = 1, \dots, m$;
  EndDo
Output: the approximations $\mu_j^{(k)}$ and $x_j^{(k)}$ to the largest eigenvalues $\mu_j$ and
    corresponding eigenvectors, $j = 1, \dots, m$.
```

For mesh eigenproblems, when high accuracy of computed eigenvalues and eigenvectors is not often required, Algorithm 5.2 can be recommended. For a general purpose method, it seems best to use a mix of Algorithms 5.1 and 5.2 such that eigenvalues, which have already formed a cluster, are treated together using Algorithm 5.1,

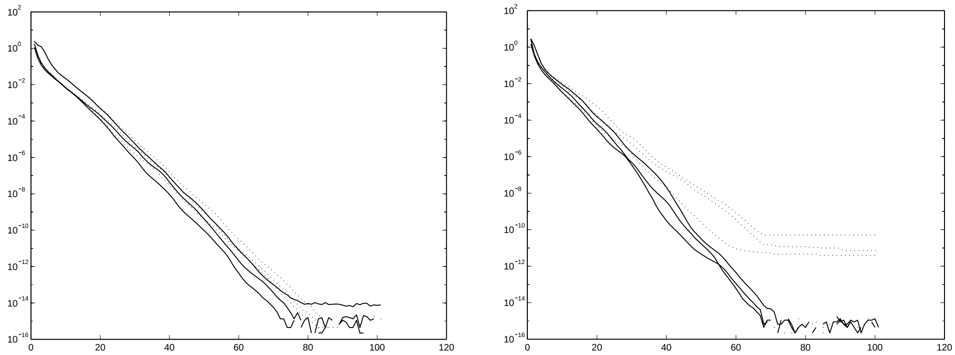


FIG. 5.4. *LOBPCG* I *(solid) vs. LOBPCG* II *(dotted): well-separated eigenvalues (left) and eigenvalue cluster (right).*

while all other eigenvalues in the block are treated separately, as in Algorithm 5.2. This approach will be implemented in the code LOBPCG.m in future revisions.

In the rest of the paper, all our numerical results are for Algorithm 5.1 only.

As theoretical investigation and comparison of preconditioned eigensolvers are quite tedious ([4] provides a perfect example of this), a possibility of a fair numerical comparison becomes even more important than usual. In the next section, we suggest a numerical benchmark for preconditioned eigensolvers.

6. Model test problems with random preconditioners. To be able to compare numerically different methods in the class with different preconditioners, we suggest the following system of model tests, with random preconditioners and initial guesses, be used for benchmarking.

For simplicity, we take the mass matrix $B = I$.

The stiffness matrix A is in our model tests a diagonal matrix with the minimal entry 1 and the maximal entry 10^{10}; therefore, all eigenvalues μ of the pencil $B - \mu A$ lie on the semiclosed interval $(0, 1]$. We are interested in finding a group of the largest eigenvalues and corresponding eigenvectors. In most of the tests of the present paper, we compute only one, the largest eigenvalue $\mu = 1$.

For preconditioned eigensolvers, we expect that the convergence does not slow down when the condition number of A gets larger [18, 19, 4, 21, 22], with a properly chosen preconditioner. Because of this, we simply use a fixed large value, 10^{10}, for the maximal entry of A. Our code seems to work robustly for condition number of A as large as 10^{16} in double precision arithmetic.

The gap between computed and excluded parts of the spectrum is known to play an important role in the convergence speed. It seems necessary to fix the gap within a series of tests. We do it with the gap ranging from 1 to 0.01 in different series. A small value of the gap may or may not lead to slow convergence, depending on several factors, the first of which is the distribution of eigenvalues in the excluded part of the spectrum close to the desired part. This makes comparison of different methods somewhat unreliable when the gap is small.

It is also necessary to choose a distribution of eigenvalues. The desired eigenvalues, if there is more then one, are distributed randomly on a given interval, as their distribution should not affect performance significantly. In the rest of the spectrum, the distribution of eigenvalues does not noticeably affect the speed of convergence in our tests for a general preconditioner. If, however, the preconditioner commutes with A, e.g., $T = I$, or $T = A^{-1}$, we do observe a strong influence of the distribution on convergence. For such cases, we choose a distribution that mimics that of an ordinary differential equation of the fourth order, but with the given maximal entry 10^{10} (see above). The initial guess is fixed for every run of the actual and the control codes but is changed for every new run as a vector with random entries, chosen from a normal distribution with mean zero and variance one.

The preconditioner T is also fixed for every run of the actual and the control programs but is modified for every new run as a random symmetric positive definite matrix with the fixed value of $\kappa(TA) = \delta_1/\delta_0$ of (2.1). We construct T as follows.

First, we chose a diagonal matrix D with random diagonal entries, chosen from a uniform distribution on the interval $(0, 1)$. Then we find minimal $\min D$ and maximal $\max D$ values of D and do a linear scaling $D = 1+(D-\min D)/(\max D-\min D)*(\kappa-1)$, where $\kappa = \kappa(TA)$ is the desired fixed value of the spectral condition number of TA. That makes diagonal entries of D uniformly distributed on the interval $(1, \kappa)$ with minimal and maximal values exactly at the end points of the interval; thus, the condition number of D equals exactly κ.

Second, we take a square matrix with random entries, chosen from a normal distribution with mean zero and variance one, and perform the standard orthogonalization procedure on it. That produces a random orthogonal matrix Q. We now scale it, $S = QA^{-1/2}$, keeping in mind that Q is orthogonal and A is diagonal, and therefore, $S^T = A^{-1/2}Q^{-1}$.

Finally, we form $T = S^T DS$. The matrix T is clearly symmetric. Moreover, the diagonal entries of D and columns of Q are eigenpairs of the matrix $A^{1/2}TA^{1/2} = Q^{-1}DQ$, which completes our argument.

There are two reasons for using random preconditioners for our model test problems. First, it is a natural choice when solving eigenvalue problems for diagonal matrices. Second, it allows us to make a fair numerical comparison of different eigensolvers and gives a simple opportunity to check that the best method in the class consistently outperforms other methods independently of the choice of the preconditioner for a fixed value of κ.

The size of the problem varies from 1000–4000.

The upper bound, 4000, is simply determined by our computer resources. The above algorithm of constructing a random preconditioner is quite expensive and leads to a full matrix T. The total cost grows cubically with the size of the problem. We find in our tests that small problems may lead to unreliable conclusions when the number of iterations is large enough. Namely, in some tests, depending on distribution of excluded eigenvalues of the original pencil, we observe a superlinear convergence of our methods when the total number of steps was more than 30% of the size of the problem. However, in practical applications of interest, eigenvalue problems are so large that the number of steps should not usually exceed 20% of the size of the problem, taking also into account that the high accuracy of desired eigenpairs of the algebraic pencil is rarely needed, as the pencil itself is just an approximation of the original continuous problem and the approximation error may not be small. Thus, one should not count in practice on having a superlinear convergence; and we try to rule such a possibility out by choosing the size of the problem large enough.

We recommend every new preconditioned eigensolver be compared with our "ideal" algorithm in our model test problems in terms of the speed of convergence, costs of every iteration, and memory requirements. We provide such comparison for our

LOBPCG method in the next section.

7. Numerical results: LOBPCG vs. PCGNULL. Here, we solve a model eigenvalue problem with $\lambda_1 = 1,\ \lambda_2 = 2$ and take the condition number of A to be 10^{10}.

For simplicity, the mass matrix equals the identity $B = I$.

We compute only the first eigenpair, i.e., the smallest eigenvalue $\lambda_1 = 1$ and the corresponding eigenvector. Thus, we set the block size $m = 1$ in our LOBPCG Algorithm 5.1.

The initial guess is fixed for every run of the actual and the control codes but is modified randomly for every new run. The preconditioner T is also fixed for every run of the actual and the control programs but is modified for every new run as a random symmetric positive definite matrix with the fixed value of the condition number $\kappa(TA)$; see above. We vary $\kappa(TA)$ in different series of tests.

The straight (and green on a color print) line corresponds to linear convergence with the residual reduction rate (3.4), which is the same as (5.5) with $m = 1$ and $\mu_{\min} = 0$. To be consistent with MATLAB's built-in code PCG.m, we measure the error as the Euclidean norm of the residual, i.e., $\|(A-\lambda_1 B)x^{(i)}\|/\|x^{(i)}\|$ in PCGNULL and $\|(A-\lambda^{(i)} B)x^{(i)}\|/\|x^{(i)}\|$ in our code. With these definitions, norms of the residuals are quite large on the first few iterations in our tests as our matrix A is very ill-conditioned.

The average slope is the most important. We observe in Figure 7.1 that the average residual reduction rate is about the same for the "ideal" method, PCGNULL, and for our LOBPCG, and is quite close to the theoretical prediction. Convergence history lines for every method are tightly bundled together, with the bundle for our LOBPCG (colored red in the electronic version of the paper) consistently a bit lower than the bundle for the PCGNULL (dotted and blue). We present here, because of space limitations, only cases $\kappa(TA) = 4$ and $\kappa(TA) = 1000$. Pictures with other values of κ are similar to those shown in Figure 7.1. Thus, our code converges essentially as fast as the "ideal" method.

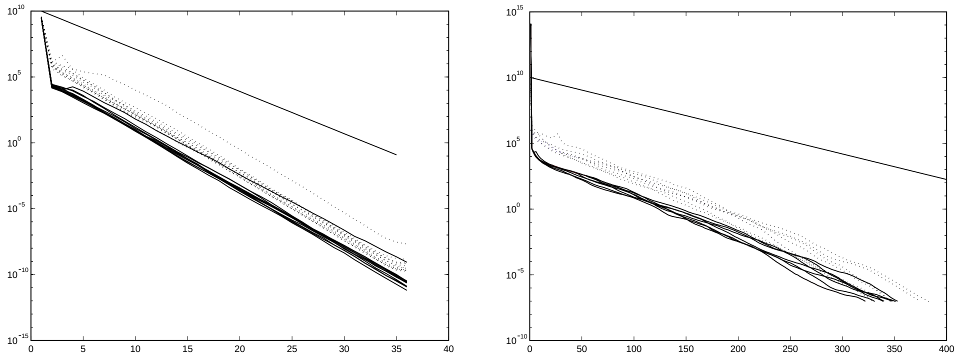


FIG. 7.1. *LOPCG (solid) vs. ideal (dotted):* $\kappa(TA) = 4$ *(left) and* $\kappa(TA) = 1000$ *(right).*

Let us now compare computational costs of a single iteration. PCGNULL involves one application of the preconditioner T, and one multiplication of A and a vector. LOBPCG (revision 3.2.9) has exactly the same major costs; however, for very ill-conditioned problems and when a very high accuracy is required, to increase stability we perform A-based orthogonalization of basis vectors of the trial subspace, which may require matrix A be multiplied two times instead of one on every iteration.

In terms of memory use, both methods are similar, as they require only several vectors, but no large matrices, to be stored.

To conclude, numerical results establish that our algorithm is practically as efficient as the "ideal" algorithm when preconditioners and initial approximations are the same in both methods.

8. Numerical results: LOBPCG vs. GLOBALMIN.

In the previous section, we compare our LOBPCG with the benchmark based on PCGNULL and show that LOBPCG is practically the optimal preconditioned method for finding the extreme eigenvalue and the corresponding eigenvector. LOBPCG can also be used, of course, for computing a group of extreme eigenvalues when the block size $m > 1$.

What benchmark do we advocate for preconditioned eigensolvers for finding several eigenpairs? We do not have an answer to this question as satisfactory as that for a single extreme eigenpair, because we are not yet able to suggest a convincing "ideal" (in terms of speed and costs) solver. We do have, however, the block globally optimal solver, Algorithm 2.1, which computes optimal approximations on the block generalized Krylov subspace.

Let us highlight again that the number of vectors in the basis of the block Krylov subspace (2.6) grows exponentially, which makes Algorithm 2.1 very expensive. On the other hand, it provides the global optimization of the Rayleigh quotient on the block Krylov subspace and, thus, can be used for numerical comparison with actual block preconditioned eigensolvers to check if they provide approximations close to those of the global optimization.

We write a MATLAB code of Algorithm 2.1, called GLOBALMIN.m, using recursion (2.7), followed by complete orthogonalization.

We tested LOBPCG vs. GLOBALMIN on model problems described in section 6 with $n = 3000$. The gap between computed and excluded parts of the spectrum is one. The condition number of A is chosen to be 10^8 as GLOBALMIN fails in some tests for larger condition numbers.

We find in the present section that putting only one run on a figure is more illustrative, as LOBPCG and GLOBALMIN produce very similar results, but they change greatly with different random initial guesses. We remove the initial value of the error from all pictures as it is typically too large to fit the chosen scale. LOBPCG is presented by a solid (and red in a color version of the paper) line, while GLOBALMIN is dashed (and blue). The straight (and green) line corresponds, as in section 7, to the average error reduction predicted by (5.5).

The residual-based error is measured as $\|(A-\lambda_j^{(i)}B)x_j^{(i)}\|/\|x_j^{(i)}\|$ on a corresponding Ritz vector $x_j^{(i)}$. The eigenvalue error is simply measured as the difference of the Ritz value $\lambda_j^{(i)}$ and the corresponding eigenvalue λ_j. Both methods, LOBPCG and GLOBALMIN, should monotonically decrease the eigenvalue error. GLOBALMIN provides the global minimum of the Rayleigh quotient; therefore, the eigenvalue error of GLOBALMIN should be always not larger than that of LOBPCG.

We first compare, on Figure 8.1, errors just for the smallest eigenvalue λ_1 for the LOBPCG and GLOBALMIN both with block size one as in section 7. Figure 8.1 displays errors for the same problems as Figure 7.1, but we also add the eigenvalue error. We highlight again that dimension of the generalized Krylov subspace global (2.4) grows exponentially with the number of iterations. For numerical tests

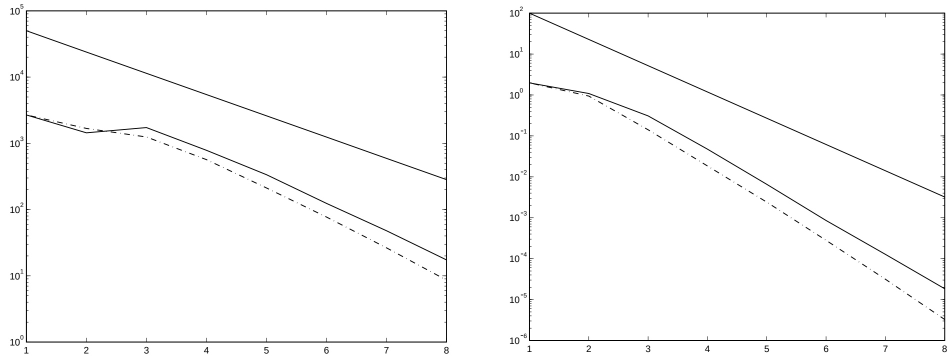


FIG. 8.1. *LOBPCG (solid) vs. GLOBALMIN (dashed), $\kappa(TA) = 4$, $m = 1$: residuals (left) and eigenvalue errors (right).*

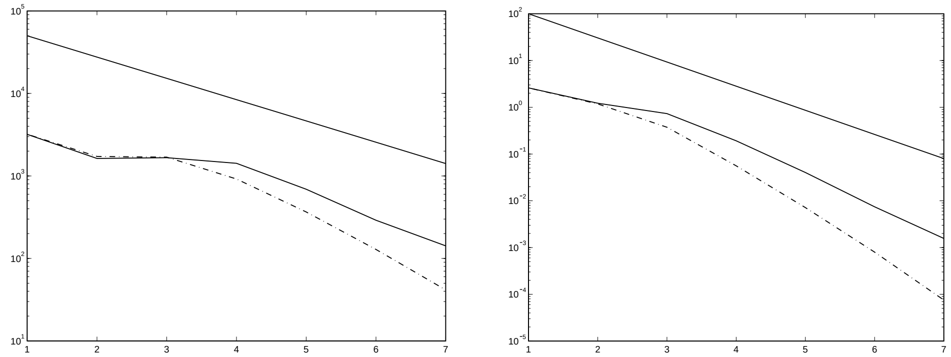


FIG. 8.2. *LOBPCG (solid) vs. GLOBALMIN (dashed), $\kappa(TA) = 4$, $m = 3$: residuals (left) and eigenvalue errors (right).*

presented in Figure 8.1, typical dimensions are $3, 7, 15, 31, 63, 127, 255, 511$, e.g., on the last (eighth) iteration, GLOBALMIN minimizes the Rayleigh quotient on a trial subspace of dimension 511.

In Figures 8.2 and 8.3, we compare the error for the third smallest eigenvalue λ_3 for the LOBPCG and GLOBALMIN, both with block size three. In these experiments, dimensions of the block generalized Krylov subspace global (2.6) typically are $9, 21, 45, 93, 189, 381, 765$. As our code GLOBALMIN is based on complete orthogonalization that filters out possible linearly dependent vectors in the trial subspace, in different tests we observe slightly different dimensions.

The trial subspace in GLOBALMIN is getting large enough—about 10%–20% of the size of the problem—to lead to a superlinear convergence of GLOBALMIN when $\kappa(TA)$ is not too large; see Figure 8.2. We believe that this effect is artificially created by the fact that our problem is of a small size, only 3000, and should be disregarded. Our computer resources do not allow us to solve larger problems.

We first observe that LOBPCG and GLOBALMIN produce almost the same approximations on the first two steps. Most importantly, by comparing the slopes on the figures, we come to the conclusion that our LOBPCG provides approximations close to those of the global optimization method on the generalized block Krylov

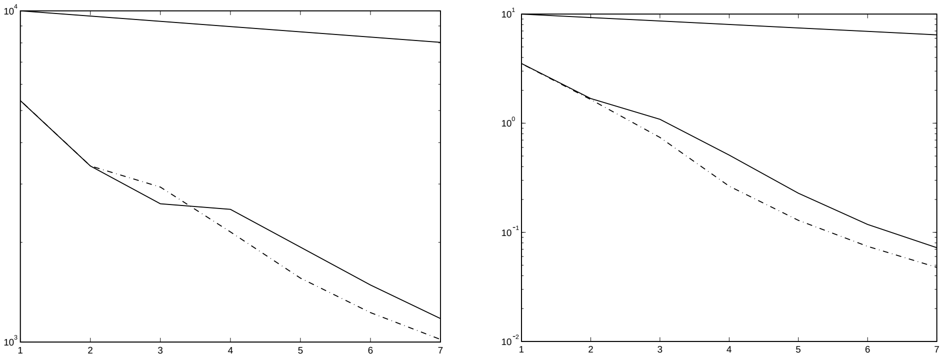


FIG. 8.3. *LOBPCG (solid) vs. GLOBALMIN (dashed),* $\kappa(TA) = 1000$*,* $m = 3$*: residuals (left) and eigenvalue errors (right).*

subspace and has a similar convergence speed.

9. LOBPCG vs. Davidson's method.

The discussion above allows us to make some conclusions on LOBPCG vs. Davidson's method, though we do not have a numerical comparison. The block Davidson method without restarts can be presented (cf. [5, 27, 37]) as the Rayleigh–Ritz method on the trial subspace spanned on vectors:

$$\left\{ \begin{array}{l} x_1^{(0)},\ T(B-\mu_1^{(0)}A)x_1^{(0)},\ T(B-\mu_1^{(1)}A)x_1^{(1)},\dots,\ T(B-\mu_1^{(i)}A)x_1^{(i)},\dots \\ x_m^{(0)},\ T(B-\mu_m^{(0)}A)x_m^{(0)},\ T(B-\mu_m^{(1)}A)x_m^{(1)},\dots,\ T(B-\mu_m^{(i)}A)x_m^{(i)} \end{array} \right\}, \tag{9.1}$$

and $x_j^{(i+1)}$ is computed as the jth Ritz vector. All vectors (9.1) are in the block generalized Krylov subspace (2.6) (assuming a fixed preconditioner), so such defined block Davidson method cannot converge faster than the GLOBALMIN. But our LOBPCG converges with about the same rate as the GLOBALMIN. Therefore, we can expect that LOBPCG is more efficient than Davidson's method, since the former should not converge much slower than, but is significantly less expensive than, the latter.

To make Davidson's method more competitive with our LOBPCG, one needs to restart after every k steps in the following special way: the Rayleigh–Ritz method is now used on the trial subspace spanned by vectors

$$\begin{aligned} &\Big\{x_1^{(i-1)},\ x_1^{(i)},\ T(B-\mu_1^{(i)}A)x_1^{(i)},\ T(B-\mu_1^{(i+1)}A)x_1^{(i+1)},\dots,\ T(B-\mu_1^{(i+k)}A)x_1^{(i+k)},\dots, \\ &\quad x_m^{(i-1)},\ x_m^{(i)},\ T(B-\mu_m^{(i)}A)x_m^{(i)},\ T(B-\mu_m^{(i+1)}A)x_m^{(i+1)},\dots,\ T(B-\mu_m^{(i+k)}A)x_m^{(i+k)}\Big\}, \end{aligned} \tag{9.2}$$

and $x_j^{(i+k+1)}$ is computed as the jth Ritz vector. The new trial subspace is still a subset of the block generalized Krylov subspace (2.6), but its dimension does not depend on the number of iterations.

Compared to a naive method of restarts, we have extra vectors, $x_j^{(i-1)}$, $j = 1,\dots,m$, in the basis of the trial subspace, which we expect will make method (9.2) much faster. We now notice that Davidson's method based on (9.2) with $k = 0$ coincides with our earlier method (5.1). Thus, our Algorithm 5.1 can be viewed as a specially-restarted-at-every-step block Davidson method.

Is there any benefit to using block Davidson method, based on (9.2) with $k > 0$? In our opinion, for symmetric eigenproblems, the answer seems to be no. We expect methods with $k = 0$ and $k > 0$ to be quite similar to each other in terms of speed of convergence, as the method with $k = 0$ already provides approximations practically close to those of the global optimization method on the block Krylov subspace. At the same time, method (9.2) with $k > 0$ will be somewhat more computationally expensive and less stable for ill-conditioned problems simply because the dimension of its trial subspace for the Rayleigh–Ritz method is larger. A direct numerical comparison is yet to be done.

10. Numerical results: LOBPCG vs. JDQR. Here, we present numerical results for the MATLAB code JDQR.m of the inexact Jacobi–Davidson method [14], written by Gerard Sleijpen, which is publicly available on the Internet (http://www.math.uu.nl/people/sleijpen/JD_software/JDQR.html). We are not able to compare it with PCGNULL, as recommended, because norms of the actual residuals are not available in JDQR on every inner iteration. Instead, we provide results of direct numerical comparison of JDQR with our method LOBPCG.

As in the previous two sections, the comparison is made using model eigenvalue problems with $B = I$ (a revision of JDQR code for generalized eigenproblems is not yet available) and random preconditioners, suggested in section 6. JDQR is used with the default tolerance. The number of iterations of our LOBPCG is chosen to match the accuracy of eigenvector approximations provided by the JDQR. We measure the accuracy as the angle between computed and exact invariant subspaces in the two-norm.

First, we find that JDQR is not as robust as our method with respect to ill-conditioning of the matrix A and the number of required eigenpairs. JDQR consistently fails to find even one eigenpair for condition number of A above 10^8. JDQR becomes even more sensitive to ill-conditioning when we increase the number of required eigenpairs. With $cond(A) = 10^6$, JDQR typically fails to compute all ten required eigenpairs in another series of tests, and in some of the tests outputs only one eigenpair out of ten. Attempts to compute forty eigenpairs using JDQR even with $cond(A) = 10$ produce no more than 16 eigenpairs.

JDQR does not handle random initial guess very well. In some tests it converges to the second eigenpair instead of the desired first one, with the smallest eigenvalue λ.

Our LOBPCG is much more robust and successfully computes all required eigenpairs in all tests mentioned above without any difficulties. Moreover, LOBPCG always converges about one-and-a-half to two times faster than JDQR if we count the number of iterations as the number of times the preconditioner is invoked. This may not be very surprising, as the only MATLAB version of the inexact Jacobi–Davidson method available to us for testing can be used for nonsymmetric eigenproblems as well and is not apparently optimized for symmetric eigenvalue problems, while our method takes full advantage of the symmetry by using a three-term recurrence.[1]

Both methods are scalable, as expected, with respect to the size of the problem and the quality of the preconditioner used.

A comparison of LOBPCG with JDQR has also been made for some practical problems, e.g., for an eigenvalue problem describing vibrations of a slender beam,

[1] A new MATLAB package, JDCG by Yvan Notay, was released in 2001; see http://mnsgi.ulb.ac.be/pub/docs/jdcg. It implements the Jacobi–Davidson method with a PCG inner solver, specifically tuned for the symmetric case [31]. According to our numerical tests, while JDCG does accelerate JDQR, it still converges slower than—and for clusters of eigenvalues is not as reliable as—our LOBPCG.

with a domain-decomposition-based preconditioner. This is outside of the scope of the present paper and will be reported elsewhere. Let us just highlight that the conclusions based on these practical comparisons are the same as above for our model diagonal eigenproblem with random preconditioners.

11. Availability of software for the preconditioned eigensolvers. The Internet page http:// www-math.cudenver.edu/~aknyazev/software/CG/ is maintained by the author. It contains all the software used for numerical experiments of the present paper, including benchmarking codes written in MATLAB by the author.

12. Conclusion. Let us formulate here the main points of the paper:

- Numerical results establish that our algorithm LOBPCG is practically as efficient as the "ideal" algorithm for computing an extreme eigenpair and provides approximations close to those of the global optimization method on the generalized block Krylov subspace.
- Our method is much more robust and typically converges about one and a half to two times faster than the inexact Jacobi–Davidson method.
- We provide a system of test problems with random preconditioners that we suggest be used for benchmarking. Every new preconditioned solver for finding an extreme eigenpair should be compared with the "ideal" algorithm in terms of the speed of convergence, costs of every iteration, and memory requirements. As the number of publications on different preconditioned eigenvalue solvers and their applications, e.g., recent papers [36, 43, 44, 24, 2, 34, 33], keeps growing rapidly, a need for such benchmarking becomes evident.

REFERENCES

[1] L. Bergamaschi, G. Gambolati, and G. Pini, *Asymptotic convergence of conjugate gradient methods for the partial symmetric eigenproblem*, Numer. Linear Algebra Appl., 4 (1997), pp. 69–84.

[2] L. Borges and S. Oliveira, *A parallel Davidson-type algorithm for several eigenvalues*, J. Comput. Phys., 144 (1998), pp. 727–748.

[3] W. W. Bradbury and R. Fletcher, *New iterative methods for solution of the eigenproblem*, Numer. Math., 9 (1966), pp. 259–267.

[4] J. H. Bramble, J. E. Pasciak, and A. V. Knyazev, *A subspace preconditioning algorithm for eigenvector/eigenvalue computation*, Adv. Comput. Math., 6 (1996), pp. 159–189.

[5] E. R. Davidson, *Matrix eigenvector methods*, in Methods in Computational Molecular Physics, G. H. F. Direcksen and S. Wilson, eds., D. Reidel, Boston, MA, 1983, pp. 95–113.

[6] D. C. Dobson, *An efficient method for band structure calculations in* 2D *photonic crystals*, J. Comput. Phys., 149 (1999), pp. 363–376.

[7] D. C. Dobson, J. Gopalakrishnan, and J. E. Pasciak, *An efficient method for band structure calculations in* 3D *photonic crystals*, J. Comput. Phys., 161 (2000), pp. 668–679.

[8] E. G. D'yakonov, *Optimization in solving elliptic problems*, CRC Press, Boca Raton, FL, 1996.

[9] E. G. D'yakonov and A. V. Knyazev, *Group iterative method for finding lower-order eigenvalues*, Moscow Univ. Comput. Math. Cybernet., No. 2, 1982, pp. 32–40.

[10] E. G. D'yakonov and A. V. Knyazev, *On an iterative method for finding lower eigenvalues*, Russian J. Numer. Anal. Math. Modelling, 7 (1992), pp. 473–486.

[11] A. Edelman, T. A. Arias, and S. T. Smith, *The geometry of algorithms with orthogonality constraints*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 303–353.

[12] A. Edelman, T. A. Arias, and S. T. Smith, *Curvature in conjugate gradient eigenvalue computation with applications to Materials and Chemistry Calculations*, in Proceedings of the Fifth SIAM Conference on Applied Linear Algebra, J. G. Lewis, ed., SIAM, Philadelphia, 1994, pp. 233–238.

[13] Y. T. Feng and D. R. J. Owen, *Conjugate gradient methods for solving the smallest eigenpair of large symmetric eigenvalue problems*, Internat. J. Numer. Methods Engrg., 39 (1996), pp. 2209–2229.

[14] D. R. Fokkema, G. L. G. Sleijpen, and H. A. Van der Vorst, *Jacobi-Davidson style QR and QZ algorithms for the reduction of matrix pencils*, SIAM J. Sci. Comput., 20 (1999), pp. 94–125.

[15] G. Gambolati, F. Sartoretto, and P. Florian, *An orthogonal accelerated deflation technique for large symmetric eigenproblems*, Comput. Methods Appl. Mech. Engrg., 94 (1992), pp. 13–23.

[16] A. V. Knyazev, *Toward the Optimal Preconditioned Eigensolver: Locally Optimal Block Preconditioned Conjugate Gradient Method*, Technical report UCD-CCM 149, Center for Computational Mathematics, University of Colorado at Denver, 2000.

[17] A. V. Knyazev, *Computation of eigenvalues and eigenvectors for mesh problems: Algorithms and error estimates*, Dept. Numerical Math., USSR Academy of Sciences, Moscow, 1986 (in Russian).

[18] A. V. Knyazev, *Convergence rate estimates for iterative methods for mesh symmetric eigenvalue problem*, Soviet J. Numer. Anal. Math. Modelling, 2 (1987), pp. 371–396.

[19] A. V. Knyazev, *A preconditioned conjugate gradient method for eigenvalue problems and its implementation in a subspace*, in Eigenwertaufgaben in Natur- und Ingenieurwissenschaften und ihre numerische Behandlung, Oberwolfach, 1990, Internat. Ser. Numer. Math. 96, Birkhäuser, Basel, 1991, pp. 143–154.

[20] A. V. Knyazev, *New estimates for Ritz vectors*, Math. Comp., 66 (1997), pp. 985–995.

[21] A. V. Knyazev, *Preconditioned eigensolvers—an oxymoron?*, Electron. Trans. Numer. Anal., 7 (1998), pp. 104–123.

[22] A. V. Knyazev, *Preconditioned eigensolvers: Practical algorithms*, in Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide, SIAM, Philadelphia, 2000, pp. 352–368. An extended version published as Technical report UCD-CCM 143, Center for Computational Mathematics, University of Colorado, Denver, 1999.

[23] A. V. Knyazev and A. L. Skorokhodov, *Preconditioned gradient-type iterative methods in a subspace for partial generalized symmetric eigenvalue problems*, SIAM J. Numer. Anal., 31 (1994), pp. 1226–1239.

[24] S. H. Lui, H. B. Keller, and T. W. C. Kwok, *Homotopy method for the large, sparse, real nonsymmetric eigenvalue problem*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 312–333.

[25] A. Meyer, *Modern Algorithms for Large Sparse Eigenvalue Problems*, Mathematical Research 34, Akademie-Verlag, Berlin, 1987.

[26] M. Mongeau and M. Torki, *Computing Eigenelements of Real Symmetric Matrices via Optimization*, Technical report 99.54, MIP University Paul Sabatier, Toulouse, France, 1999.

[27] R. B. Morgan, *Davidson's method and preconditioning for generalized eigenvalue problems*, J. Comput. Phys., 89 (1990), pp. 241–245.

[28] K. Neymeyr, *A geometric theory for preconditioned inverse iteration applied to a subspace*, Math. Comp., submitted.

[29] K. Neymeyr, *A geometric theory for preconditioned inverse iteration.* I*: Extrema of the Rayleigh quotient*, Linear Algebra Appl., 322 (2001), pp. 61–85.

[30] K. Neymeyr, *A geometric theory for preconditioned inverse iteration.* II*: Sharp convergence estimates*, Linear Algebra Appl., 322 (2001), pp. 87–104.

[31] Y. Notay, *Combination of Jacobi–Davidson and conjugate gradients for the partial symmetric eigenproblem*, Numer. Linear Algebra Appl., to appear. Also available from http://homepages.ulb.ac.be/˜ynotay/.

[32] E. E. Ovtchinnikov and L. S. Xanthis, *Effective dimensional reduction algorithm for eigenvalue problems for thin elastic structures: A paradigm in three dimensions*, Proc. Natl. Acad. Sci. USA, 97 (2000), pp. 967–971.

[33] E. E. Ovtchinnikov and L. S. Xanthis, *Successive eigenvalue relaxation: A new method for generalized eigenvalue problems and convergence estimates*, Proc. Roy. Soc. London Sect. A, 457 (2001), pp. 441–451.

[34] B. G. Pfrommer, J. Demmel, and H. Simon, *Unconstrained energy functionals for electronic structure calculations*, J. Comput. Phys., 150 (1999), pp. 287–298.

[35] B. T. Polyak, *Introduction to Optimization*, Optimization Software Inc. Publications Division, New York, 1987.

[36] A. Ruhe, *Rational Krylov: A practical algorithm for large sparse nonsymmetric matrix pencils*, SIAM J. Sci. Comput., 19 (1998), pp. 1535–1551.

[37] Y. Saad, *Numerical Methods for Large Eigenvalue Problems*, Halsted, New York, 1992.

[38] P. Smit and M. H. C. Paardekooper, *The effects of inexact solvers in algorithms for symmetric eigenvalue problems*, Linear Algebra Appl., 287 (1999), pp. 337–357.

[39] S. I. Solov'ev, *Convergence of the Modified Subspace Iteration Method for Nonlinear Eigenvalue Problems*, Preprint SFB393/99-35, Sonderforschungsbereich 393 an der Technischen

Universität Chemnitz, Technische Universität, Chemnitz, Germany, 1999. Also available from http://www.tu-chemnitz.de/sfb393/Files/PS/sfb99-35.ps.gz.

[40] S. I. Solov'ev, *Preconditioned Gradient Iterative Methods for Nonlinear Eigenvalue Problems*, Preprint SFB393/00-28, Sonderforschungsbereich 393 an der Technischen Universität Chemnitz, Technische Universität, Chemnitz, Germany, 2000. Also available from http://www.tu-chemnitz.de/sfb393/Files/PS/sfb00-28.ps.gz.

[41] E. Suetomi and H. Sekimoto, *Conjugate gradient like methods and their application to eigenvalue problems for neutron diffusion equation*, Ann. Nuclear Energy, 18 (1991), pp. 205.

[42] H. Yang, *Conjugate gradient methods for the Rayleigh quotient minimization of generalized eigenvalue problems*, Computing, 51 (1993), pp. 79–94.

[43] T. Zhang, G. H. Golub, and K. H. Law, *Subspace iterative methods for eigenvalue problems*, Linear Algebra Appl., 294 (1999), pp. 239–258.

[44] T. Zhang, K. H. Law, and G. H. Golub, *On the homotopy method for perturbed symmetric generalized eigenvalue problems*, SIAM J. Sci. Comput., 19 (1998), pp. 1625–1645.

SIAM J. SCI. COMPUT.
Vol. 23, No. 2, pp. 542–562

A SYMMETRIC BAND LANCZOS PROCESS BASED ON COUPLED RECURRENCES AND SOME APPLICATIONS*

ZHAOJUN BAI† AND ROLAND W. FREUND‡

Abstract. The symmetric band Lanczos process is an extension of the classical Lanczos algorithm for symmetric matrices and single starting vectors to multiple starting vectors. After n iterations, the symmetric band Lanczos process has generated an $n \times n$ projection, $\mathbf{T}_n^{\mathrm{s}}$, of the given symmetric matrix onto the n-dimensional subspace spanned by the first n Lanczos vectors. This subspace is closely related to the nth block-Krylov subspace induced by the given symmetric matrix and the given block of multiple starting vectors. The standard algorithm produces the entries of $\mathbf{T}_n^{\mathrm{s}}$ directly. In this paper, we propose a variant of the symmetric band Lanczos process that employs coupled recurrences to generate the factors of an $\mathrm{LDL}^{\mathrm{T}}$ factorization of a closely related $n \times n$ projection, rather than $\mathbf{T}_n^{\mathrm{s}}$. This is done in such a way that the factors of the $\mathrm{LDL}^{\mathrm{T}}$ factorization inherit the "fish-bone" structure of $\mathbf{T}_n^{\mathrm{s}}$. Numerical examples from reduced-order modeling of large electronic circuits and algebraic eigenvalue problems demonstrate that the proposed variant of the band Lanczos process based on coupled recurrences is more robust and accurate than the standard algorithm.

Key words. symmetric matrix, block-Krylov subspace, multiple starting vectors, orthogonal basis, projection, reduced-order modeling, passivity, circuit simulation, eigenvalue problem

AMS subject classifications. 65F15, 65F30, 65L80, 93B40

PII. S1064827500371773

1. Introduction.

In recent years, suitable extensions of the classical Lanczos process [23] for single right and left starting vectors to multiple right and left starting vectors have proven to be powerful tools for reduced-order modeling of large-scale multi-input multi-output linear dynamical systems. The most general such Lanczos-type algorithm is the one proposed in [1]. Given a square matrix $\mathbf{A}$ and two blocks of right and left starting vectors, the algorithm in [1] generates two sequences of biorthogonal basis vectors for the right and left block-Krylov subspaces induced by the given data. The algorithm can handle the most general case of right and left starting blocks of arbitrary sizes, say, m and p. In [12, 13], it was shown that this Lanczos-type algorithm can be employed to generate reduced-order models of m-input p-output linear dynamical systems and that these reduced-order models correspond to matrix-Padé approximants of the system's transfer function. The resulting computational procedure is called the MPVL (matrix-Padé via Lanczos) algorithm. For recent surveys on MPVL, related methods, and their use in VLSI circuit simulation, we refer the reader to [14, 16].

In circuit simulation, an important special case is linear dynamical systems that describe large RCL subcircuits consisting of only resistors, capacitors, and inductors. In this case, by employing so-called modified nodal analysis, the circuit equations can be formulated so that the matrix $\mathbf{A}$ is symmetric and the blocks of right and left starting vectors are identical; for details of such a symmetric formulation, see,

*Received by the editors May 4, 2000; accepted for publication (in revised form) December 11, 2000; published electronically July 10, 2001.
http://www.siam.org/journals/sisc/23-2/37177.html

†Department of Computer Science, University of California at Davis, One Shields Avenue, Davis, CA 95616 (bai@cs.ucdavis.edu). Most of this work was done while this author was on sabbatical at Bell Laboratories, Lucent Technologies, Murray Hill, NJ.

‡Bell Laboratories, Lucent Technologies, 700 Mountain Avenue, Room 2C–525, Murray Hill, NJ 07974–0636 (freund@research.bell-labs.com).

e.g., [18]. The SyMPVL algorithm [17, 18] is a variant of MPVL that exploits this symmetry and as a result requires only half the computational costs and storage of MPVL. The Lanczos-type algorithm underlying SyMPVL is essentially the symmetric band Lanczos process first proposed in [26], with some additional features, such as deflation of Krylov vectors; see [15]. It generates a sequence of $n \times n$ projections $\mathbf{T}_n^{\mathrm{s}}$, $n \geq 1$, of the symmetric matrix $\mathbf{A}$ onto the n-dimensional subspace spanned by the first n Lanczos vectors. This subspace is closely related to the nth block-Krylov subspace induced by $\mathbf{A}$ and the given block of multiple starting vectors. As the classical Lanczos process, the symmetric band Lanczos process generates the entries of $\mathbf{T}_n^{\mathrm{s}}$ directly.

In this paper, we propose a new variant of the symmetric band Lanczos process that employs coupled recurrences to produce the factors of an $\mathrm{LDL}^{\mathrm{T}}$ factorization of an $n \times n$ matrix, $\mathbf{T}_n$, closely related to $\mathbf{T}_n^{\mathrm{s}}$, rather than $\mathbf{T}_n^{\mathrm{s}}$ itself. This work was motivated mainly by numerical experiences with the SyMPVL algorithm applied to RC subcircuits. In this case, the matrix $\mathbf{A}$ is symmetric positive semidefinite, and in order to ensure passivity of the reduced-order models produced by SyMPVL, it is crucial that the projection of $\mathbf{A}$ preserves the positive semidefiniteness of $\mathbf{A}$. In exact arithmetic, the projection $\mathbf{T}_n^{\mathrm{s}}$ of a positive semidefinite matrix $\mathbf{A}$ is guaranteed to be positive semidefinite. However, in finite-precision arithmetic, round-off may cause the computed projection $\mathbf{T}_n^{\mathrm{s}}$ to be slightly indefinite; see [4] for such examples. These problems are clearly due to the direct computation of $\mathbf{T}_n^{\mathrm{s}}$. Indeed, when the projection is obtained via the $\mathrm{LDL}^{\mathrm{T}}$ factorization produced by the proposed symmetric band Lanczos process based on coupled recurrences, the computed projection preserves the positive semidefiniteness of $\mathbf{A}$. In addition, the new variant also produces more accurate reduced-order models with the same number of iterations.

The idea of enforcing positive semidefiniteness of a matrix by generating it via a factorization is of course not new. Indeed, the very same issue arises in Kalman filtering where numerical round-off in the standard update formula of the covariance matrices may result in slightly indefinite matrices. The remedy there is to update suitable factors, rather than the covariance matrices, resulting in so-called square-root filtering; see, e.g., [3, Chapter 6.5]. The idea of square-root filtering seems to have originated with James E. Potter in 1962; see [3, Chapter 6.5] and [8, section 13.7]. The use of square-root filtering was crucial in eliminating problems in the Apollo navigation systems caused by numerical round-off; see [8] and the references given there. The same issue also arises in the computation of positive semidefinite solutions of Lyapunov equations; see [22] and the references therein. In [22], an algorithm is proposed that directly generates the Cholesky factor of the solution matrix, rather than the solution matrix.

Finally, in the context of employing the classical Lanczos process for single starting vectors to the solution of systems of linear equations, it has been observed that coupled two-term recurrences are more robust than the mathematically equivalent three-term recurrences; see, e.g., [19] and the references given there. Some recent analysis of this phenomenon can be found in [21].

The remainder of this paper is organized as follows. In section 2, we describe the governing equations of the symmetric band Lanczos process based on coupled recurrences and discuss connections with the standard algorithm. In section 3, we present a complete statement of the algorithm based on coupled recurrences and establish some of its properties. In sections 4 and 5, we discuss applications of the new variant of the symmetric band Lanczos process to reduced-order modeling and to eigenvalue computations, respectively, and we report results of numerical experiments. In section 6, we make some concluding remarks.

Throughout this article, we use boldface letters to denote vectors and matrices. Unless stated otherwise, vectors and matrices are assumed to have real entries. As usual, $\mathbf{M}^{\mathrm{T}} = [\, m_{kj} \,]$ denotes the transpose of the matrix $\mathbf{M} = [\, m_{jk} \,]$, and $\mathbf{M} \geq \mathbf{0}$ ($\mathbf{M} > \mathbf{0}$) means that $\mathbf{M} = \mathbf{M}^{\mathrm{T}}$ is symmetric positive semidefinite (symmetric positive definite). The vector norm $\|\mathbf{x}\| := \sqrt{\mathbf{x}^{\mathrm{T}}\,\mathbf{x}}$ is always the Euclidean norm, and $\|\mathbf{M}\| := \max_{\|\mathbf{x}\|=1} \|\mathbf{M}\,\mathbf{x}\|$ is the corresponding induced matrix norm. We use $\mathbf{I}_n$ to denote the $n \times n$ identity matrix and $\mathbf{0}_{n\times m}$ to denote the $n \times m$ zero matrix; we will omit these indices whenever the actual dimensions of $\mathbf{I}$ and $\mathbf{0}$ are apparent from the context. The sets of real and complex numbers are denoted by $\mathbb{R}$ and $\mathbb{C}$, respectively. For $s \in \mathbb{C}$, $\mathrm{Re}(s)$ is the real part of s.

2. The symmetric band Lanczos process based on coupled recurrences.

In this section, we describe the governing equations of the symmetric band Lanczos process based on coupled recurrences and discuss connections with the standard algorithm.

2.1. Preliminaries.

In what follows, we assume that

$$\mathbf{A} = \mathbf{A}^{\mathrm{T}} \in \mathbb{R}^{N\times N} \quad \text{and} \quad \mathbf{R} = [\, \mathbf{r}_1 \quad \mathbf{r}_2 \quad \cdots \quad \mathbf{r}_m \,] \in \mathbb{R}^{N\times m} \tag{2.1}$$

are given matrices. The columns of $\mathbf{R}$ are the multiple starting vectors. The purpose of the symmetric band Lanczos process is to generate orthogonal basis vectors for the subspaces spanned by the leading columns of the *block-Krylov matrix*,

$$\mathbf{K}(\mathbf{A}, \mathbf{R}) := [\, \mathbf{R} \quad \mathbf{A}\,\mathbf{R} \quad \mathbf{A}^2\,\mathbf{R} \quad \cdots \quad \mathbf{A}^{N-1}\,\mathbf{R} \,], \tag{2.2}$$

associated with the matrices (2.1) and to compute the projection of $\mathbf{A}$ onto these subspaces. A proper definition of these subspaces is necessarily quite involved; see the discussion in [1]. The main reason is that the columns of the matrix $\mathbf{K}(\mathbf{A}, \mathbf{R})$ in (2.2) are all vectors of length N, and thus at most N of them are linearly independent. As a result, in the symmetric band Lanczos process, one needs to perform so-called *deflation* of linearly dependent or in some sense almost linearly dependent vectors. As we will describe below, the symmetric band Lanczos process has a very simple built-in deflation procedure. In exact arithmetic, only the linearly dependent vectors have to be removed, and we refer to this as *exact* deflation. In the case of exact deflation, the subspaces spanned by the leading columns of the matrix (2.2) can be easily defined, and we will do so in section 2.5 below.

2.2. Governing equations.

The symmetric band Lanczos algorithm is an iterative procedure. After n iterations, the algorithm has generated the first n *Lanczos vectors*,

$$\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n \in \mathbb{R}^N, \tag{2.3}$$

as well as the $m_{\mathrm{c}} = m_{\mathrm{c}}(n)$ vectors,

$$\hat{\mathbf{v}}_{n+1}, \hat{\mathbf{v}}_{n+2}, \ldots, \hat{\mathbf{v}}_{n+m_{\mathrm{c}}} \in \mathbb{R}^N, \tag{2.4}$$

that are the candidates for the next m_{c} Lanczos vectors, $\mathbf{v}_{n+1}, \mathbf{v}_{n+2}, \ldots, \mathbf{v}_{n+m_{\mathrm{c}}}$. Here, $m_{\mathrm{c}} = m_{\mathrm{c}}(n)$ is the *current block size* defined as follows. In the initialization phase, i.e., for $n = 0$, $m_{\mathrm{c}} := m$ is set to be the number of starting vectors in (2.1), and the candidate vectors (2.4) are set to be the starting vectors from (2.1), i.e., $\hat{\mathbf{v}}_i = \mathbf{r}_i$, $1 \le i \le m$. Within the algorithm, $m_{\rm c}$ is then reduced by 1 every time a deflation occurs. Moreover, in the algorithm, the vectors (2.4) are used to detect and perform deflation. More precisely, it can be shown that an exact deflation occurs during the $(n+1)$st iteration of the algorithm if and only if $\hat{\mathbf{v}}_{n+1} = \mathbf{0}$. Therefore, in the algorithm, one simply checks if the first of the candidate vectors is the zero vector or in some sense close to the zero vector, and if so, one performs deflation by removing that first candidate vector.

We remark that due to the use of the candidate vectors (2.4), we need only generate $n \times n$ Lanczos matrices. This is in contrast to approaches such as the symmetric algorithm [26] and the nonsymmetric algorithm [1], which only involve Lanczos vectors, but no candidate vectors, and generate $(n+m_{\rm c})\times n$ Lanczos matrices. Since for the applications we have in mind only the leading $n \times n$ part of the Lanczos matrices is needed anyway, the approach using candidate vectors is more economical.

In addition to (2.3), the symmetric band Lanczos process based on coupled recurrences also generates a second set of vectors,

$$\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n \in \mathbb{R}^N, \tag{2.5}$$

that span the same subspaces as (2.3), i.e.,

$$\operatorname{span}\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_j\} = \operatorname{span}\{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_j\} \quad \text{for all} \quad 1 \le j \le n. \tag{2.6}$$

The vectors (2.3)–(2.5) are constructed to satisfy (in exact arithmetic) the following orthogonality conditions:

$$\mathbf{V}_n^{\rm T}\,\mathbf{V}_n = \mathbf{I}_n, \tag{2.7}$$

$$\mathbf{V}_n^{\rm T}\,[\,\hat{\mathbf{v}}_{n+1} \quad \hat{\mathbf{v}}_{n+2} \quad \cdots \quad \hat{\mathbf{v}}_{n+m_{\rm c}}\,] = \mathbf{0}, \tag{2.8}$$

$$\mathbf{P}_n^{\rm T}\,\mathbf{A}\,\mathbf{P}_n = \boldsymbol{\Delta}_n := \operatorname{diag}(\delta_1, \delta_2, \ldots, \delta_n)\,. \tag{2.9}$$

Here and in what follows,

$$\mathbf{V}_n := [\,\mathbf{v}_1 \quad \mathbf{v}_2 \quad \cdots \quad \mathbf{v}_n\,] \quad \text{and} \quad \mathbf{P}_n := [\,\mathbf{p}_1 \quad \mathbf{p}_2 \quad \cdots \quad \mathbf{p}_n\,] \tag{2.10}$$

denote the matrices whose columns are just the vectors (2.3) and (2.5), respectively. Furthermore, in (2.9), we assume that $\delta_i \neq 0$ for all i. This implies that

$$\boldsymbol{\Delta}_n \quad \text{is nonsingular.} \tag{2.11}$$

Note that, by (2.7), the Lanczos vectors (2.3) are an orthonormal basis of the subspaces (2.6), while, by (2.9), the vectors (2.5) are an $\mathbf{A}$-orthogonal basis of (2.6).

The recurrence relations that are used in the algorithm to generate the vectors (2.3)–(2.5) can be stated as follows. The first m_1 vectors in (2.3) are obtained by applying a modified Gram–Schmidt procedure [20] (with deflation) to the m columns of the block $\mathbf{R}$ to eliminate any linearly dependent or almost linearly dependent starting vectors. This procedure can be summarized in a relation of the form

$$\mathbf{R} = \mathbf{V}_{m_1}\,\boldsymbol{\rho}_{m_1} + \widehat{\mathbf{V}}_0^{\rm df}. \tag{2.12}$$

Here, $m_1\,(\le m)$ denotes the number of columns of the block $\mathbf{R}$ that have not been deflated, the matrix $\widehat{\mathbf{V}}_0^{\rm df} \in \mathbb{R}^{N \times m}$ contains the $m - m_1$ deflated starting vectors and m_1 zero vectors as columns, and

$$\boldsymbol{\rho}_{m_1} = [\,\rho_{i,j}\,]_{i=1,2,\ldots,m_1;\,j=1,2,\ldots,m} \in \mathbb{R}^{m_1 \times m} \tag{2.13}$$

is an upper triangular matrix whose entries are chosen such that the columns of $\mathbf{V}_{m_1}$ are orthonormal. The main relations for generating the vectors (2.3)–(2.5) are coupled recurrences that can be summarized as follows:

$$\mathbf{A}\,\mathbf{P}_n = \mathbf{V}_n\,\mathbf{L}_n\,\boldsymbol{\Delta}_n + \bigl[\,\underbrace{\mathbf{0} \quad \mathbf{0} \quad \cdots \quad \mathbf{0}}_{n-m_{\mathrm{c}}} \quad \underbrace{\hat{\mathbf{v}}_{n+1} \quad \hat{\mathbf{v}}_{n+2} \quad \cdots \quad \hat{\mathbf{v}}_{n+m_{\mathrm{c}}}}_{m_{\mathrm{c}}}\,\bigr] + \widehat{\mathbf{V}}_n^{\mathrm{df}}, \tag{2.14}$$

$$\mathbf{V}_n = \mathbf{P}_n\,\mathbf{U}_n. \tag{2.15}$$

Here, $\boldsymbol{\Delta}_n$ is the diagonal matrix defined in (2.9), and

$$\mathbf{L}_n = [\,\ell_{ij}\,]_{i,j=1,2,\dots,n} \in \mathbb{R}^{n\times n} \quad \text{and} \quad \mathbf{U}_n = [\,u_{ij}\,]_{i,j=1,2,\dots,n} \in \mathbb{R}^{n\times n} \tag{2.16}$$

are unit lower and upper triangular matrices, respectively. The entries of the matrices $\boldsymbol{\Delta}_n$, $\mathbf{L}_n$, and $\mathbf{U}_n$ are chosen such that the vectors (2.3)–(2.5) satisfy the orthogonality conditions (2.7)–(2.9). Furthermore, any candidate vector $\hat{\mathbf{v}}_j$ that was deflated at the jth iteration, where $1 + m_{\mathrm{c}}(j) \le j \le n$, appears as the $(j - m_{\mathrm{c}}(j))$th column of the matrix $\widehat{\mathbf{V}}_n^{\mathrm{df}}$ in (2.14); all other columns of $\widehat{\mathbf{V}}_n^{\mathrm{df}}$ are zero vectors. In the actual Algorithm 3.1 below, we use the index set $\mathcal{I}$ to record the positions of the potentially nonzero columns of $\widehat{\mathbf{V}}_n^{\mathrm{df}}$ due to deflation. If no deflation has occurred or if only exact deflation is performed, then clearly

$$\widehat{\mathbf{V}}_0^{\mathrm{df}} = \mathbf{0} \quad \text{and} \quad \widehat{\mathbf{V}}_n^{\mathrm{df}} = \mathbf{0} \tag{2.17}$$

in (2.12) and (2.14), respectively.

2.3. The Lanczos matrix. By multiplying (2.9) from the left by $\mathbf{U}_n^{\mathrm{T}}$ and from the right by $\mathbf{U}_n$, and by using (2.15), it follows that

$$\mathbf{T}_n := \mathbf{V}_n^{\mathrm{T}}\,\mathbf{A}\,\mathbf{V}_n = \mathbf{U}_n^{\mathrm{T}}\,\boldsymbol{\Delta}_n\,\mathbf{U}_n. \tag{2.18}$$

The matrix $\mathbf{T}_n$ defined in (2.18) is the so-called *nth Lanczos matrix.* It represents the projection of $\mathbf{A}$ onto the subspaces spanned by the Lanczos vectors (2.3). Recall that $\mathbf{U}_n$ is a unit upper triangular matrix and that $\boldsymbol{\Delta}_n$ is a diagonal matrix. Thus, in view of (2.18), the symmetric band Lanczos process based on coupled recurrences computes the factors of an LDL$^{\mathrm{T}}$ decomposition of the Lanczos matrix $\mathbf{T}_n$, rather than $\mathbf{T}_n$ itself.

In the important special case $\mathbf{A} \ge \mathbf{0}$, in view of (2.9), the diagonal entries of $\boldsymbol{\Delta}_n$ satisfy

$$\delta_i = \mathbf{p}_i^{\mathrm{T}}\,\mathbf{A}\,\mathbf{p}_i \ge 0 \quad \text{for all} \quad i, \tag{2.19}$$

and hence $\boldsymbol{\Delta}_n \ge \mathbf{0}$. Thus, for $\mathbf{A} \ge \mathbf{0}$, generating $\mathbf{T}_n$ via the factorization (2.18) always results in a positive semidefinite matrix.

The triangular matrices $\mathbf{L}_n$ in (2.14) and $\mathbf{U}_n$ in (2.15) are closely related. Indeed, by multiplying (2.14) from the left by $\mathbf{V}_n^{\mathrm{T}}$ and from the right by $\mathbf{U}_n$, and by using (2.7), (2.8), and (2.15), we get

$$\mathbf{V}_n^{\mathrm{T}}\,\mathbf{A}\,\mathbf{V}_n = (\mathbf{L}_n\,\boldsymbol{\Delta}_n + \boldsymbol{\Sigma}_n)\,\mathbf{U}_n, \quad \text{where} \quad \boldsymbol{\Sigma}_n := \mathbf{V}_n^{\mathrm{T}}\,\widehat{\mathbf{V}}_n^{\mathrm{df}}. \tag{2.20}$$

By comparing (2.18) and (2.20), it follows that

$$\mathbf{U}_n = \mathbf{L}_n^{\mathrm{T}} + \boldsymbol{\Delta}_n^{-1}\,\boldsymbol{\Sigma}_n^{\mathrm{T}}. \tag{2.21}$$

In particular, if no deflation has occurred or if only exact deflation is performed, then, in view of (2.17), we have $\boldsymbol{\Sigma}_n = \mathbf{0}$ in (2.20) and thus $\mathbf{U}_n = \mathbf{L}_n^{\mathrm{T}}$.

2.4. Structure of the triangular factor $\mathbf{U}_n$. In this subsection, we describe the sparsity structure of the matrix $\mathbf{U}_n$.

The unit upper triangular $\mathbf{U}_n$ consists of a banded part with bandwidth decreasing from $m+1$ to $m_{\mathrm{c}}+1$ and a "spiked" part with potentially nonzero elements only in rows with index $i \in \mathcal{I}$ and to the right of the banded part. Recall that the index set $\mathcal{I}$ records deflation. It turns out that in the additive splitting (2.21) of $\mathbf{U}_n$, the first term, $\mathbf{L}_n^{\mathrm{T}}$, is the banded part, while the second term, $\mathbf{\Delta}_n^{-1}\,\mathbf{\Sigma}_n^{\mathrm{T}}$, is the spiked part.

Next, we present an example that illustrates this structure of $\mathbf{U}_n$. Consider the case that $m = 5$ and that the three candidate vectors $\hat{\mathbf{v}}_8$ (when $m_{\mathrm{c}} = 5$), $\hat{\mathbf{v}}_{11}$ (when $m_{\mathrm{c}} = 4$), and $\hat{\mathbf{v}}_{13}$ (when $m_{\mathrm{c}} = 3$) have been deflated. The associated index set is $\mathcal{I} = \{\,3, 7, 10\,\}$. After $n = 15$ iterations, we then have $m_{\mathrm{c}} = 2$, and the matrix $\mathbf{U}_{15}$ has the following structure:

$$\mathbf{U}_{15} = \begin{bmatrix}
1 & \times & \times & \times & \times & \times & & & & & & & & & \\
 & 1 & \times & \times & \times & \times & \times & & & & & & & & \\
 & & 1 & \times & \times & \times & \times & * & * & * & * & * & * & * & * \\
 & & & 1 & \times & \times & \times & \times & & & & & & & \\
 & & & & 1 & \times & \times & \times & \times & & & & & & \\
 & & & & & 1 & \times & \times & \times & \times & & & & & \\
 & & & & & & 1 & \times & \times & \times & * & * & * & * & * \\
 & & & & & & & 1 & \times & \times & \times & & & & \\
 & & & & & & & & 1 & \times & \times & \times & & & \\
 & & & & & & & & & 1 & \times & \times & * & * & * \\
 & & & & & & & & & & 1 & \times & \times & & \\
 & & & & & & & & & & & 1 & \times & \times & \\
 & & & & & & & & & & & & 1 & \times & \times \\
 & & & & & & & & & & & & & 1 & \times \\
 & & & & & & & & & & & & & & 1
\end{bmatrix}.$$

Here, potentially nonzero off-diagonal entries in the banded part of $\mathbf{U}_{15}$ are marked by "×", and potentially nonzero entries in the spiked part are marked by "$*$".

2.5. Block-Krylov subspaces in the case of exact deflation. It can be shown that performing only exact deflation in the symmetric band Lanczos process is equivalent to scanning the columns of the block-Krylov matrix (2.2) from left to right and deleting each column that is linearly dependent on earlier columns; see [1]. The result of such a deletion of columns in (2.2) is the *deflated* block-Krylov matrix

$$\mathbf{K}^{\mathrm{df}}(\mathbf{A},\mathbf{R}) := [\,\mathbf{R}_1 \quad \mathbf{A}\,\mathbf{R}_2 \quad \mathbf{A}^2\,\mathbf{R}_3 \quad \cdots \quad \mathbf{A}^{j_{\max}-1}\,\mathbf{R}_{j_{\max}}\,].$$

By the structure of (2.2), a column $\mathbf{A}^{j-1}\,\mathbf{r}_i$ being linearly dependent on earlier columns implies that all $\mathbf{A}$-multiples, $\mathbf{A}^k\mathbf{r}_i$, $k \geq j$, of that column are also linearly dependent on earlier columns. As a result, for each $j = 1, 2, \ldots, j_{\max}$, $\mathbf{R}_j$ is a submatrix of $\mathbf{R}_{j-1}$, where, for $j = 1$, we set $\mathbf{R}_0 := \mathbf{R}$.

By construction, all columns of the matrix $\mathbf{K}^{\mathrm{df}}(\mathbf{A},\mathbf{R})$ are linearly independent. The *nth block-Krylov subspace*, $\mathcal{K}_n(\mathbf{A},\mathbf{R})$, induced by $\mathbf{A}$ and $\mathbf{R}$ is now defined as the n-dimensional subspace of $\mathbb{R}^N$ spanned by the first n columns of the matrix $\mathbf{K}^{\mathrm{df}}(\mathbf{A},\mathbf{R})$. When only exact deflation is performed in the symmetric band Lanczos process, then the Lanczos vectors (2.3) span the nth block-Krylov subspace, i.e.,

$$\operatorname{span}\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\} = \mathcal{K}_n(\mathbf{A},\mathbf{R}). \tag{2.22}$$

2.6. The standard symmetric band Lanczos process. In this subsection, we review the standard symmetric band Lanczos process. The standard process only generates Lanczos vectors (2.3) and candidate vectors (2.4), but not the second set of vectors (2.5). We stress that our notion of "standard" process includes the same proper deflation procedure as used in the coupled algorithm. Moreover, all numerical comparisons reported in this paper were run with deflation turned on in both algorithms. In particular, the difference in the numerical behavior is indeed due to the use of coupled vs. standard recurrences, and not due to one algorithm being run with deflation and the other without deflation.

In the standard algorithm, the Lanczos vectors (2.3) and candidate vectors (2.4) are constructed to satisfy the orthogonality conditions (2.7) and (2.8). The first m_1 Lanczos vectors again satisfy a relation of the form (2.12). The main recurrences used in the standard algorithm can be summarized in the form

$$\mathbf{A}\,\mathbf{V}_n^{\mathrm{s}} = \mathbf{V}_n^{\mathrm{s}}\,\widetilde{\mathbf{T}}_n^{\mathrm{s}} + \big[\,\underbrace{\mathbf{0}\quad\mathbf{0}\quad\cdots\quad\mathbf{0}}_{n-m_{\mathrm{c}}}\quad\underbrace{\hat{\mathbf{v}}_{n+1}^{\mathrm{s}}\quad\hat{\mathbf{v}}_{n+2}^{\mathrm{s}}\quad\cdots\quad\hat{\mathbf{v}}_{n+m_{\mathrm{c}}}^{\mathrm{s}}}_{m_{\mathrm{c}}}\,\big] + \widehat{\mathbf{V}}_n^{\mathrm{s,\,df}}. \tag{2.23}$$

Here, the upper index "s" is used to differentiate the vectors and matrices generated by the standard process from those generated by the coupled process.

In (2.23), $\widetilde{\mathbf{T}}_n^{\mathrm{s}}$ is an $n\times n$ matrix whose entries are chosen such that the vectors (2.3) and (2.4) satisfy the orthogonality conditions (2.7) and (2.8). By multiplying (2.23) from the left by $(\mathbf{V}_n^{\mathrm{s}})^{\mathrm{T}}$, and by using (2.7) and (2.8), it follows that

$$\mathbf{T}_n^{\mathrm{s}} := (\mathbf{V}_n^{\mathrm{s}})^{\mathrm{T}}\,\mathbf{A}\mathbf{V}_n^{\mathrm{s}} = \widetilde{\mathbf{T}}_n^{\mathrm{s}} + (\mathbf{V}_n^{\mathrm{s}})^{\mathrm{T}}\,\widehat{\mathbf{V}}_n^{\mathrm{s,\,df}}. \tag{2.24}$$

It turns out that $\mathbf{T}_n^{\mathrm{s}}$ consists of a symmetric banded part with bandwidth decreasing from $2m+1$ to $2m_{\mathrm{c}}+1$ and a "spiked" part with potentially nonzero elements only in rows and columns with index $i \in \mathcal{I}$ and outside of the banded part. This means that the matrix $\mathbf{T}_n^{\mathrm{s}}$ has the same "fish-bone" sparsity structure as $\mathbf{U}_n^{\mathrm{T}} + \mathbf{U}_n$, where $\mathbf{U}_n$ is the triangular factor of $\mathbf{T}_n$ described in section 2.4.

For the example considered in section 2.4, the corresponding matrix $\mathbf{T}_n^{\mathrm{s}}$ produced by the standard algorithm has the following structure:

$$\mathbf{T}_{15}^{\mathrm{s}} = \begin{bmatrix}
\times & \times & \times & \times & \times & \times & & & & & & & & & \\
\times & \times & \times & \times & \times & \times & \times & & & & & & & & \\
\times & \times & \times & \times & \times & \times & \times & * & * & * & * & * & * & * & * \\
\times & \times & \times & \times & \times & \times & \times & \times & & & & & & & \\
\times & \times & \times & \times & \times & \times & \times & \times & \times & & & & & & \\
\times & \times & \times & \times & \times & \times & \times & \times & \times & \times & & & & & \\
 & \times & \times & \times & \times & \times & \times & \times & \times & \times & * & * & * & * & * \\
 & & * & \times & \times & \times & \times & \times & \times & \times & \times & & & & \\
 & & * & & \times & \times & \times & \times & \times & \times & \times & \times & & & \\
 & & * & & & \times & \times & \times & \times & \times & \times & \times & * & * & * \\
 & & * & & & & * & \times & \times & \times & \times & \times & \times & & \\
 & & * & & & & * & & \times & \times & \times & \times & \times & \times & \\
 & & * & & & & * & & & * & \times & \times & \times & \times & \times \\
 & & * & & & & * & & & * & & \times & \times & \times & \times \\
 & & * & & & & * & & & * & & & \times & \times & \times
\end{bmatrix}.$$

Here, potentially nonzero entries in the banded part of $\mathbf{T}_{15}^{\mathrm{s}}$ are marked by "×", and potentially nonzero entries in the spiked part are marked by "$*$".

For further details and properties of the standard symmetric band Lanczos algorithm, we refer the reader to [15]. We would also like to point the reader to the earlier work [9, 10, 26] on symmetric band or block Lanczos algorithms. The symmetric band Lanczos algorithm proposed in [26] identifies Lanczos vectors that need to be deflated in a similar fashion as described above. However, the deflated vectors are then simply discarded, and, as a result, a relation such as (2.24) holds true only approximately for this algorithm. The symmetric block Lanczos algorithm described in [9] and [10, Chapter 7] is a block and not a band procedure. At each block iteration, a new block of m_{c} vectors is computed so that it is orthogonal to the earlier blocks. Similar to the deflation procedure described above, vectors in the new block that are in some sense close to the zero vector are properly deflated, and not just discarded, before the remaining columns in the new block are orthogonalized.

2.7. Connection with the standard process. The standard band Lanczos process and the algorithm based on coupled recurrences are mathematically equivalent in the case that no deflation occurs or that only exact deflation is performed. Indeed, in this case, the first n Lanczos vectors (2.3) generated by both algorithms are identical and, in view of (2.22), span the nth block-Krylov subspace $\mathcal{K}_n(\mathbf{A}, \mathbf{R})$. Hence, the associated projected Lanczos matrices $\mathbf{T}_n$ and $\mathbf{T}_n^{\mathrm{s}}$ given by (2.18) and (2.24) are identical. Moreover, in (2.14) and (2.23), we have $\widehat{\mathbf{V}}_n^{\mathrm{df}} = \widehat{\mathbf{V}}_n^{\mathrm{s,\,df}} = \mathbf{0}$. This implies that the spiked parts of both $\mathbf{T}_n$ and $\mathbf{U}_n$ are actually zero matrices. Thus both $\mathbf{T}_n$ and $\mathbf{U}_n = \mathbf{L}_n^{\mathrm{T}}$ are banded matrices, and (2.18) reduces to the LDL$^{\mathrm{T}}$ factorization

$$\mathbf{T}_n = \mathbf{T}_n^{\mathrm{s}} = \mathbf{L}_n \, \boldsymbol{\Delta}_n \, \mathbf{L}_n^{\mathrm{T}}$$

of the Lanczos matrix associated with both variants of the symmetric band Lanczos process.

As soon as deflation of almost linearly dependent vectors is performed, the two processes are no longer mathematically equivalent in general. In this case, the spiked parts of both $\mathbf{T}_n$ and $\mathbf{U}_n$ are nonzero, and thus

$$\mathbf{T}_n^{\mathrm{s}} \neq \mathbf{T}_n = \mathbf{U}_n^{\mathrm{T}} \, \boldsymbol{\Delta}_n \, \mathbf{U}_n$$

in general. Indeed, $\mathbf{T}_n^{\mathrm{s}}$ has a “fish-bone” sparsity structure, while $\mathbf{T}_n$ is a full matrix in general, even though $\mathbf{U}_n$ is sparse. However, one can show that $\|\mathbf{T}_n - \mathbf{T}_n^{\mathrm{s}}\|$ is bounded by the tolerance used to check for deflation.

3. The algorithm and its properties. In this section, we present a detailed statement of the symmetric band Lanczos process with coupled recurrences, and establish some of its properties.

3.1. Statement of the algorithm. At each pass n through the main loop of the algorithm, one first checks if the candidate vector $\hat{\mathbf{v}}_n$ needs to be deflated. If yes, it is deleted, the indices of all the remaining candidate vectors are reduced by 1, and the deflation check is repeated. If no, the candidate vector $\hat{\mathbf{v}}_n$ is normalized to become the nth Lanczos vector $\mathbf{v}_n$. In the remaining steps of pass n, the algorithm orthogonalizes the candidate vectors against $\mathbf{v}_n$, generates the potentially nonzero entries of the nth column of $\mathbf{U}_n$ and, if $n \leq m_1$, of the nth row of $\boldsymbol{\rho}_{m_1}$, and computes the vector $\mathbf{p}_n$, the scalar δ_n, and a new candidate vector $\hat{\mathbf{v}}_{n+m_{\mathrm{c}}}$.

A detailed statement of the algorithm is as follows.

Algorithm 3.1 (symmetric band Lanczos process with coupled recurrences).

```
INPUT: A matrix $\mathbf{A} = \mathbf{A}^{\mathrm{T}} \in \mathbb{R}^{N \times N}$, and a block $\mathbf{R} = [\, \mathbf{r}_1 \quad \mathbf{r}_2 \quad \dots \quad \mathbf{r}_m \,] \in \mathbb{R}^{N \times m}$
        of $m$ starting vectors.
OUTPUT: Matrices $\mathbf{U}_n$, $\mathbf{\Delta}_n$, and (if $n \geq m_1$) $\boldsymbol{\rho}_{m_1}$.
    (0) Set $\hat{\mathbf{v}}_i = \mathbf{r}_i$ for $i = 1, 2, \dots, m$.
        Set $m_{\mathrm{c}} = m$.
        Set $\mathcal{I} = \emptyset$.
For $n = 1, 2, \dots,$ do:
    (1) (If necessary, deflate.)
        Decide if $\hat{\mathbf{v}}_n$ should be deflated.
        If no, continue with step 2.
        If yes, deflate $\hat{\mathbf{v}}_n$ by doing the following:
         (a) If $n - m_{\mathrm{c}} > 0$, set $\mathcal{I} = \mathcal{I} \cup \{n - m_{\mathrm{c}}\}$ and $\hat{\mathbf{v}}^{\mathrm{df}}_{n-m_{\mathrm{c}}} = \hat{\mathbf{v}}_n$.
         (b) Set $m_{\mathrm{c}} = m_{\mathrm{c}} - 1$. If $m_{\mathrm{c}} = 0$, set $n = n - 1$ and stop.
         (c) For $i = n, n+1, \dots, n + m_{\mathrm{c}} - 1$, set $\hat{\mathbf{v}}_i = \hat{\mathbf{v}}_{i+1}$.
         (d) Repeat all of step 1.
    (2) (Normalize $\hat{\mathbf{v}}_n$ to become the $n$th Lanczos vector $\mathbf{v}_n$.)
        Set $\mathbf{v}_n = \dfrac{\hat{\mathbf{v}}_n}{\|\hat{\mathbf{v}}_n\|}$.
        If $n - m_{\mathrm{c}} > 0$, set $u_{n-m_{\mathrm{c}},n} = \dfrac{\|\hat{\mathbf{v}}_n\|}{\delta_{n-m_{\mathrm{c}}}}$.
        If $n - m_{\mathrm{c}} \leq 0$, set $\rho_{n,n-m_{\mathrm{c}}+m} = \|\hat{\mathbf{v}}_n\|$.
        If $n = m_{\mathrm{c}}$, set $m_1 = m_{\mathrm{c}}$.
    (3) (Orthogonalize the vectors $\hat{\mathbf{v}}_{n+j}$, $1 \leq j < m_{\mathrm{c}}$, against $\mathbf{v}_n$.)
        For $j = 1, 2, \dots, m_{\mathrm{c}} - 1$, do:
            Set $\tau = \mathbf{v}_n^{\mathrm{T}} \hat{\mathbf{v}}_{n+j}$ and $\hat{\mathbf{v}}_{n+j} = \hat{\mathbf{v}}_{n+j} - \mathbf{v}_n \, \tau$.
            If $n + j - m_{\mathrm{c}} > 0$, set $u_{n+j-m_{\mathrm{c}},n} = \dfrac{\tau}{\delta_{n+j-m_{\mathrm{c}}}}$.
            If $n + j - m_{\mathrm{c}} \leq 0$, set $\rho_{n,n+j-m_{\mathrm{c}}+m} = \tau$.
    (4) (Update the spiked part of $\mathbf{U}_n$.)
        For each $j \in \mathcal{I}$, set $u_{jn} = \dfrac{\mathbf{v}_n^{\mathrm{T}} \hat{\mathbf{v}}^{\mathrm{df}}_j}{\delta_j}$.
    (5) (Compute the vector $\mathbf{p}_n$.)
        Set $j_0 = \max\{1,\, n - m_{\mathrm{c}}\}$ and

(3.1)          $\mathbf{p}_n = \mathbf{v}_n - \sum_{j \in \mathcal{I}} \mathbf{p}_j \, u_{jn} - \sum_{j=j_0}^{n-1} \mathbf{p}_j \, u_{jn}$.

        Set $u_{nn} = 1$.
    (6) (Advance the block-Krylov subspace.)
        Set $\hat{\mathbf{v}}_{n+m_{\mathrm{c}}} = \mathbf{A} \, \mathbf{p}_n$.
    (7) Set $\delta_n = \mathbf{p}_n^{\mathrm{T}} \hat{\mathbf{v}}_{n+m_{\mathrm{c}}}$.
        If $\delta_n = 0$, stop: look-ahead would be needed to continue.
    (8) Set $\hat{\mathbf{v}}_{n+m_{\mathrm{c}}} = \hat{\mathbf{v}}_{n+m_{\mathrm{c}}} - \mathbf{v}_n \, \delta_n$.
    (9) Test for convergence.
```

Remark 3.1. In Algorithm 3.1, only the potentially nonzero entries of the matrices $\mathbf{U}_n$ and $\boldsymbol{\rho}_{m_1}$ and the diagonal entries of the diagonal matrix $\boldsymbol{\Delta}_n$ are computed. All other entries of these matrices are set to be zero.

Remark 3.2. The entries of $\mathbf{U}_n$ generated in step 3 of Algorithm 3.1 are just the potentially nonzero off-diagonal entries of the banded part of the unit upper triangular matrix $\mathbf{U}_n$. In view of (2.21), the unit lower triangular matrix $\mathbf{L}_n$ can be obtained by simply transposing the banded part of $\mathbf{U}_n$.

Remark 3.3. A practical criterion for deflation in step 1 of Algorithm 3.1 is as follows. The vector $\hat{\mathbf{v}}_n$ is deflated if and only if

$$\|\hat{\mathbf{v}}_n\| \leq \mathtt{dtol}_n.$$

Here,

$$\mathtt{dtol}_n := \mathtt{dtol} \times \begin{cases} \|\mathbf{r}_{n+m-m_\mathrm{c}}\| & \text{if} \quad n \leq m_1, \\ \mathrm{nest}(\mathbf{A}) & \text{if} \quad n > m_1 \end{cases} \tag{3.2}$$

is the product of an absolute deflation tolerance `dtol` and a scaling factor that makes the deflation check independent of the actual scaling of the columns $\mathbf{r}_i$ of $\mathbf{R}$ and of the matrix $\mathbf{A}$. Ideally, we would like to set $\mathrm{nest}(\mathbf{A}) = \|\mathbf{A}\|$ in (3.2). However, if $\|\mathbf{A}\|$ is not available, then $\mathrm{nest}(\mathbf{A})$ is set to be an estimate of $\|\mathbf{A}\|$. For example, we can use the estimate

$$\mathrm{nest}(\mathbf{A}) := \max_{1 \leq i \leq m_1} \frac{\|\mathbf{A}\,\mathbf{p}_i\|}{\|\mathbf{p}_i\|} \leq \|\mathbf{A}\|\,,$$

which can be obtained from the vectors generated in Algorithm 3.1 at the expense of $2m_1$ additional vector-norm computations. Based on our numerical experiences with Algorithm 3.1, we recommend the absolute deflation tolerance $\mathtt{dtol} = \sqrt{\mathtt{eps}}$, where `eps` is the machine precision.

Remark 3.4. If $\mathbf{A} \geq 0$, then $\delta_n > 0$ for all n and so Algorithm 3.1 never stops in step 7. If $\mathbf{A}$ is indefinite, then it cannot be excluded that Algorithm 3.1 terminates prematurely due to $\delta_n = 0$ in step 7. In this case, look-ahead can be used to continue the process, but in order to keep the algorithm relatively simple, we opted to treat only the no-look-ahead case in this paper. However, we stress that Algorithm 3.1 can be extended to include look-ahead. Such an extension again generates an LDL^T factorization (2.18), where $\boldsymbol{\Delta}_n$ is a block-diagonal matrix in general. Furthermore, the nonsingularity (2.11) of $\boldsymbol{\Delta}_n$ is guaranteed only for the values of n that correspond to the end of a look-ahead step.

Remark 3.5. Algorithm 3.1 requires storage of the vector $\mathbf{v}_n$, the m_c candidate vectors $\hat{\mathbf{v}}_{n+1}, \hat{\mathbf{v}}_{n+2}, \ldots, \hat{\mathbf{v}}_{n+m_\mathrm{c}}$, the m_c vectors $\mathbf{p}_{n-m_\mathrm{c}+1}, \mathbf{p}_{n-m_\mathrm{c}+2}, \ldots, \mathbf{p}_n$, and all the vectors $\mathbf{p}_i$ and $\hat{\mathbf{v}}_i^{\mathrm{df}}$ with $i \in \mathcal{I}$. Since the set $\mathcal{I}$ contains at most $m - m_\mathrm{c}$ elements, the total number of vectors to be stored is at most

$$1 + m_\mathrm{c} + m_\mathrm{c} + 2(m - m_\mathrm{c}) = 2m + 1,$$

which is identical to the storage requirement of the standard symmetric band Lanczos process.

3.2. Properties. We now show that the quantities generated by Algorithm 3.1 indeed satisfy the governing equations stated in section 2.2.

THEOREM 3.2. *The vectors and matrices generated by n iterations of Algorithm* 3.1 *satisfy the recurrence relations* (2.14), (2.15), *and* (*if* $n \geq m_1$) (2.12).

Proof. Using the matrices introduced in (2.9), (2.10), (2.13), and (2.16), it is straightforward to verify that all the recurrences employed in the first n iterations of Algorithm 3.1 can indeed be summarized as the matrix relations (2.12), (2.14), and (2.15). □

THEOREM 3.3. *The vectors and matrices generated by n iterations of Algorithm* 3.1 (*run in exact arithmetic*) *satisfy the orthogonality conditions* (2.7)–(2.9).

Proof. From steps 2, 6, and 7 of Algorithm 3.1, it follows that

$$\mathbf{v}_i^{\mathrm{T}}\,\mathbf{v}_i = 1 \quad \text{and} \quad \mathbf{p}_i^{\mathrm{T}}\,\mathbf{A}\,\mathbf{p}_i = \delta_i \quad \text{for all} \quad 1 \leq i \leq n. \tag{3.3}$$

Next, we use induction on $n\,(\geq 0)$ to show that

$$\mathbf{v}_i^{\mathrm{T}}\,\mathbf{v}_n = 0 \quad \text{for all} \quad 1 \leq i < n, \tag{3.4}$$

$$\mathbf{p}_i^{\mathrm{T}}\,\mathbf{A}\,\mathbf{p}_n = 0 \quad \text{for all} \quad 1 \leq i < n, \tag{3.5}$$

$$\mathbf{v}_i^{\mathrm{T}}\,\hat{\mathbf{v}}_{n+j} = 0 \quad \text{for all} \quad 1 \leq i \leq n,\ 1 \leq j \leq m_{\mathrm{c}}(n). \tag{3.6}$$

Note that the relations (3.4)–(3.6), together with (3.3), are equivalent to the orthogonality conditions (2.7)–(2.9).

The assertions (3.4)–(3.6) are trivially satisfied for $n = 0$, since the sets of indices i in (3.4)–(3.6) are all empty in this case. Now let $n \geq 1$, and as induction hypothesis, assume that for all $0 \leq n' < n$, we have

$$\mathbf{v}_i^{\mathrm{T}}\,\mathbf{v}_{n'} = 0 \quad \text{for all} \quad 1 \leq i < n',$$

$$\mathbf{p}_i^{\mathrm{T}}\,\mathbf{A}\,\mathbf{p}_{n'} = 0 \quad \text{for all} \quad 1 \leq i < n', \tag{3.7}$$

$$\mathbf{v}_i^{\mathrm{T}}\,\hat{\mathbf{v}}_{n'+j} = 0 \quad \text{for all} \quad 1 \leq i \leq n',\ 1 \leq j \leq m_{\mathrm{c}}(n'). \tag{3.8}$$

We need to show that the relations (3.4)–(3.6) are satisfied.

Since $\mathbf{v}_n = \hat{\mathbf{v}}_n / \|\hat{\mathbf{v}}_n\|$, the orthogonality condition (3.4) for $\mathbf{v}_n$ is an immediate consequence of (3.8) (with $n' = n - 1$ and $j = 1$).

We now turn to (3.5). Note that at the end of the ith iteration of Algorithm 3.1, we have

$$\hat{\mathbf{v}}_{i+m_{\mathrm{c}}(i)}^{(i)} := \hat{\mathbf{v}}_{i+m_{\mathrm{c}}(i)} = \mathbf{A}\,\mathbf{p}_i - \mathbf{v}_i\,\delta_i. \tag{3.9}$$

Here, the upper index "(i)" indicates that $\hat{\mathbf{v}}_{i+m_{\mathrm{c}}(i)}^{(i)}$ is the $(i + m_{\mathrm{c}}(i))$th candidate vector at the end of the ith iteration. Note that, by (3.9) and (3.4), we have

$$\mathbf{v}_n^{\mathrm{T}}\,\mathbf{A}\,\mathbf{p}_i = \mathbf{v}_n^{\mathrm{T}}\left(\hat{\mathbf{v}}_{i+m_{\mathrm{c}}(i)}^{(i)} + \mathbf{v}_i\,\delta_i\right) = \mathbf{v}_n^{\mathrm{T}}\,\hat{\mathbf{v}}_{i+m_{\mathrm{c}}(i)}^{(i)} \quad \text{for all} \quad 1 \leq i < n. \tag{3.10}$$

Let $1 \leq i < n$ be arbitrary, but fixed, and consider the transformations that are performed on the candidate vector $\hat{\mathbf{v}}_{i+m_{\mathrm{c}}(i)}^{(i)}$ during iterations $i + 1, i + 2, \ldots, n$ of Algorithm 3.1. Note that there are only two types of transformations: shift of the (lower) index of the candidate vector by -1, and orthogonalization against previous Lanczos vectors. For the transformed vector resulting from $\hat{\mathbf{v}}_{i+m_{\mathrm{c}}(i)}^{(i)}$, there are the following three cases:

(I) The transformed vector is deflated during iteration $\tilde{n}$ for some $i < \tilde{n} \le n$.

(II) The transformed vector is one of the candidate vectors $\hat{\mathbf{v}}_n, \hat{\mathbf{v}}_{n+1}, \dots, \hat{\mathbf{v}}_{n-m_{\rm c}-1}$ before steps 2 and 3 are performed within the nth iteration of Algorithm 3.1.

(III) The transformed vector is normalized to become the kth Lanczos vector $\mathbf{v}_k$ for some $k < n$.

Case I occurs if and only if $i \in \mathcal{I}$. Indeed, let $\tilde{m}_{\rm c}$ be the value of $m_{\rm c}$ when the transformed vector $\hat{\mathbf{v}}_{\tilde{n}}$ is checked for deflation during iteration $\tilde{n}$. Since $\hat{\mathbf{v}}_{\tilde{n}}$ resulted from $\hat{\mathbf{v}}^{(i)}_{i+m_{\rm c}(i)}$, we have

$$\tilde{n} = i + m_{\rm c}(i) - (m_{\rm c}(i) - \tilde{m}_{\rm c}) = i + \tilde{m}_{\rm c},$$

and thus $i = \tilde{n} - \tilde{m}_{\rm c} > 0$. By step 1(a) of Algorithm 3.1, deflation of the vector $\hat{\mathbf{v}}_{\tilde{n}}$ is equivalent to $i = \tilde{n} - \tilde{m}_{\rm c} \in \mathcal{I}$. Moreover, note that $\hat{\mathbf{v}}^{\rm df}_i = \hat{\mathbf{v}}_{\tilde{n}}$ and that $\hat{\mathbf{v}}_{\tilde{n}}$ was obtained by orthogonalizing $\hat{\mathbf{v}}^{(i)}_{i+m_{\rm c}(i)}$ against Lanczos vectors $\mathbf{v}_k$ with $k < \tilde{n} \le n$. In view of (3.4) and step 4 of Algorithm 3.1, it follows that

$$\mathbf{v}_n^{\rm T}\, \hat{\mathbf{v}}^{(i)}_{i+m_{\rm c}(i)} = \mathbf{v}_n^{\rm T}\, \hat{\mathbf{v}}^{\rm df}_i = \delta_i\, u_{in} \quad \text{if} \quad i \in \mathcal{I}. \tag{3.11}$$

Case II occurs if and only if $i \ge j_0 := \max\{1,\, n - m_{\rm c}\}$. Indeed, the transformed vector has index $i + m_{\rm c}$ and is thus one of the candidate vectors $\hat{\mathbf{v}}_n, \dots, \hat{\mathbf{v}}_{n-m_{\rm c}-1}$ if and only if

$$n \le i + m_{\rm c} \le n - m_{\rm c} - 1. \tag{3.12}$$

Since $1 \le i < n$, the condition (3.12) is equivalent to $i \ge j_0$. Note that, in view of (3.4) and steps 2 and 3 of Algorithm 3.1, we have

$$\mathbf{v}_n^{\rm T}\, \hat{\mathbf{v}}^{(i)}_{i+m_{\rm c}(i)} = \mathbf{v}_n^{\rm T}\, \hat{\mathbf{v}}_{i+m_{\rm c}} = \delta_i\, u_{in} \quad \text{if} \quad i \ge j_0. \tag{3.13}$$

Case III complements cases I and II, and thus occurs if and only if $i < j_0$ and $i \notin \mathcal{I}$. In this case, in view of (3.4), we have

$$\mathbf{v}_n^{\rm T}\, \hat{\mathbf{v}}^{(i)}_{i+m_{\rm c}(i)} = \mathbf{v}_n^{\rm T}\, \hat{\mathbf{v}}_k = \left(\mathbf{v}_n^{\rm T}\, \mathbf{v}_k\right)\, \|\hat{\mathbf{v}}_k\| = 0. \tag{3.14}$$

Here, $k < n$ is the index of the Lanczos vector $\mathbf{v}_k$ that resulted from $\hat{\mathbf{v}}^{(i)}_{i+m_{\rm c}(i)}$.

By combining (3.10) with (3.11), (3.13), and (3.14), we get

$$\mathbf{v}_n^{\rm T}\, \mathbf{A}\, \mathbf{p}_i = \mathbf{v}_n^{\rm T}\, \hat{\mathbf{v}}^{(i)}_{i+m_{\rm c}(i)} = \begin{cases} \delta_i\, u_{in} & \text{if} \quad i \in \mathcal{I} \quad \text{or} \quad j_0 \le i < n, \\ 0 & \text{if} \quad i < j_0 \quad \text{and} \quad i \notin \mathcal{I}. \end{cases} \tag{3.15}$$

On the other hand, by transposing the relation (3.1), multiplying it from the right by $\mathbf{A}\, \mathbf{p}_i$, and using (3.3) and (3.7), it follows that

$$\mathbf{p}_n^{\rm T}\, \mathbf{A}\, \mathbf{p}_i = \mathbf{v}_n^{\rm T}\, \mathbf{A}\, \mathbf{p}_i - \begin{cases} \delta_i\, u_{in} & \text{if} \quad i \in \mathcal{I} \quad \text{or} \quad j_0 \le i < n, \\ 0 & \text{if} \quad i < j_0 \quad \text{and} \quad i \notin \mathcal{I}. \end{cases} \tag{3.16}$$

Inserting (3.15) into (3.16) gives (3.5).

Finally, we show (3.6). Note that $m_{\rm c}(i)$ is a nonincreasing function of i, and in particular, $m'_{\rm c} \ge m_{\rm c}(n) =: m_{\rm c}$. By (3.8) (for $n' = n - 1$ and $2 \le j \le m_{\rm c}$), the candidate vectors $\hat{\mathbf{v}}_{n+1}, \dots, \hat{\mathbf{v}}_{n+m_{\rm c}-1}$ are orthogonal to $\mathbf{v}_1, \dots, \mathbf{v}_{n-1}$ before step 3 is performed within the nth iteration of Algorithm 3.1. The update

$$\hat{\mathbf{v}}_{n+j} = \hat{\mathbf{v}}_{n+j} - \mathbf{v}_n\, \left(\mathbf{v}_n^{\rm T}\, \hat{\mathbf{v}}_{n+j}\right), \quad 1 \le j < m_{\rm c}, \tag{3.17}$$

in step 3 then makes these candidate vectors also orthogonal to $\mathbf{v}_n$. Moreover, in view of (3.4), the update (3.17) does not destroy the orthogonality to $\mathbf{v}_1, \ldots, \mathbf{v}_{n-1}$, and thus (3.6) is satisfied for all $1 \leq j < m_{\rm c}$. In order to prove (3.6) for $j = m_{\rm c}$, we first note that, by (3.9) (for $i = n$),

$$\hat{\mathbf{v}}_{n+m_{\rm c}} = \mathbf{A}\,\mathbf{p}_n - \mathbf{v}_n\,\delta_n.$$

Multiplying this relation from the left by $\mathbf{v}_i^{\rm T}$ and using (2.6), (3.4), and (3.5), it follows that

$$\mathbf{v}_i^{\rm T}\,\hat{\mathbf{v}}_{n+m_{\rm c}} = \mathbf{v}_i^{\rm T}\,\mathbf{A}\,\mathbf{p}_n - \left(\mathbf{v}_i^{\rm T}\,\mathbf{v}_n\right)\,\delta_n = 0 \quad \text{for all} \quad 1 \leq i < n.$$

We have thus established all three relations (3.4)–(3.6), and the proof of Theorem 3.3 is complete. □

4. Application to passive reduced-order modeling.

In this section, we discuss the application of the band Lanczos process with coupled recurrences to construct passive reduced-order models.

4.1. The problem.

Consider *symmetric* m-input m-output time-invariant linear dynamical systems of the form

$$\begin{aligned} \mathbf{C}\,\frac{d}{dt}\mathbf{x}(t) &= -\mathbf{G}\,\mathbf{x}(t) + \mathbf{B}\,\mathbf{u}(t), \\ \mathbf{y}(t) &= \mathbf{B}^{\rm T}\,\mathbf{x}(t). \end{aligned} \tag{4.1}$$

Here, $\mathbf{C} = \mathbf{C}^{\rm T}$, $\mathbf{G} = \mathbf{G}^{\rm T} \in \mathbb{R}^{N \times N}$, and $\mathbf{B} \in \mathbb{R}^{N \times m}$ are given matrices. Moreover, we assume that the matrix pencil $\mathbf{G} + s\,\mathbf{C}$ is *regular*, i.e., $\det(\mathbf{G} + s\,\mathbf{C}) = 0$ for only finitely many values of $s \in \mathbb{C}$. In (4.1), the components of the given vector-valued function $\mathbf{u}\colon [0,\infty) \mapsto \mathbb{R}^m$ are the inputs, $\mathbf{y}\colon [0,\infty) \mapsto \mathbb{R}^m$ is the unknown function of outputs, the components of the unknown vector-valued function $\mathbf{x}\colon [0,\infty) \mapsto \mathbb{R}^N$ are the state variables, and N is the state-space dimension. Systems of the form (4.1) can be used to model so-called RCL electrical networks consisting of resistors, capacitors, and inductors; see, e.g., [14] and the references given there. An important special case is RC networks consisting of only resistors and capacitors; in this case, the matrices $\mathbf{C}$ and $\mathbf{G}$ in (4.1) are sparse and positive semidefinite.

A *reduced-order model* of (4.1) is a system of the same form as (4.1) but of smaller state-space dimension $n < N$. A reduced-order model of dimension n is thus of the form

$$\begin{aligned} \mathbf{C}_n\,\frac{d}{dt}\mathbf{z}(t) &= -\mathbf{G}_n\,\mathbf{z}(t) + \mathbf{B}_n\,\mathbf{u}(t), \\ \mathbf{y}(t) &= \mathbf{B}_n^{\rm T}\,\mathbf{x}(t), \end{aligned} \tag{4.2}$$

where $\mathbf{C}_n = \mathbf{C}_n^{\rm T}$, $\mathbf{G}_n = \mathbf{G}_n^{\rm T} \in \mathbb{R}^{n \times n}$, and $\mathbf{B}_n \in \mathbb{R}^{n \times m}$. These three matrices should be chosen such that the input-output mapping $\mathbf{u}(t) \mapsto \mathbf{y}(t)$ of (4.2) somehow approximates the input-output mapping of the original system (4.1).

The input-output behavior of the original system (4.1), respectively, the reduced-order model (4.2), can be described by the associated *transfer function*

$$\mathbf{Z}(s) = \mathbf{B}^{\rm T}(\mathbf{G} + s\,\mathbf{C})^{-1}\mathbf{B}, \quad \text{respectively,} \quad \mathbf{Z}_n(s) = \mathbf{B}_n^{\rm T}(\mathbf{G}_n + s\,\mathbf{C}_n)^{-1}\mathbf{B}_n, \tag{4.3}$$

where $s \in \mathbb{C}$ is a complex variable. Note that both $\mathbf{Z}$ and $\mathbf{Z}_n$ are $(m \times m)$-matrix-valued rational functions.

Now, let $s_0 \in \mathbb{C}$ be any expansion point such that the matrix $\mathbf{G} + s_0\,\mathbf{C}$ is nonsingular. The transfer function $\mathbf{Z}_n$ is called an *nth matrix-Padé approximant* of $\mathbf{Z}$ (about the expansion point s_0) if the matrices $\mathbf{C}_n$, $\mathbf{G}_n$, and $\mathbf{B}_n$ in (4.3) are chosen such that

$$\mathbf{Z}_n(s) = \mathbf{Z}(s) + \mathcal{O}\big((s - s_0)^{q(n)}\big), \tag{4.4}$$

where the integer $q(n)$ is as large as possible; see, e.g., [7, pp. 429–466]. The condition (4.4) means that the Taylor expansions of $\mathbf{Z}_n$ and $\mathbf{Z}$ about s_0 agree in as many leading ($(m \times m)$-matrix-valued) coefficients as possible. In what follows, we call the reduced-order model (4.2) of (4.1) a *matrix-Padé model* (associated with the expansion point s_0) if its transfer function $\mathbf{Z}_n$ is an nth matrix-Padé approximant of $\mathbf{Z}$.

4.2. Reduced-order models via the symmetric band Lanczos process. In what follows, we assume that the matrices $\mathbf{C}$ and $\mathbf{G}$ in (4.1) are sparse and positive semidefinite. Moreover, let $s_0 \geq 0$ be any real expansion point such that $\mathbf{G}+s_0\,\mathbf{C} > 0$, and let

$$\mathbf{G} + s_0\,\mathbf{C} = \mathbf{M}\,\mathbf{M}^{\mathrm{T}} \tag{4.5}$$

be a sparse Cholesky factorization of $\mathbf{G} + s_0\,\mathbf{C}$. Note that, in general, $\mathbf{M}$ is a product of a permutation matrix, which records any pivoting for sparsity, and a sparse lower triangular matrix. Using (4.5), the transfer function $\mathbf{Z}$ in (4.3) can be recast as follows:

$$\begin{gathered} \mathbf{Z}(s) = \mathbf{R}^{\mathrm{T}}\,(\mathbf{I} + (s - s_0)\,\mathbf{A})^{-1}\,\mathbf{R}, \\ \text{where} \quad \mathbf{A} := \mathbf{M}^{-1}\,\mathbf{C}\,\mathbf{M}^{-\mathrm{T}} \geq 0 \quad \text{and} \quad \mathbf{R} := \mathbf{M}^{-1}\,\mathbf{B}. \end{gathered} \tag{4.6}$$

In [17, 18], it was proposed to obtain reduced-order models via the symmetric band Lanczos process applied to the matrix $\mathbf{A}$ and the block of starting vectors $\mathbf{R}$ given in (4.6). More precisely, after $n\,(\geq m_1)$ iterations of the algorithm, a reduced-order transfer function of dimension n is defined as follows:

$$\mathbf{Z}_n(s) = \boldsymbol{\rho}_n^{\mathrm{T}}\,(\mathbf{I}_n + (s - s_0)\,\mathbf{T}_n))^{-1}\,\boldsymbol{\rho}_n, \quad \text{where} \quad \boldsymbol{\rho}_n = \begin{bmatrix} \boldsymbol{\rho}_{m_1} \\ \mathbf{0}_{n-m_1\times m} \end{bmatrix}. \tag{4.7}$$

Here, $\mathbf{T}_n$ is the $n\times n$ projected Lanczos matrix and $\boldsymbol{\rho}_n$ is the $m_1\times m$ matrix containing the coefficients used to orthogonalize the starting block $\mathbf{R}$. In [17, 18], the standard symmetric band Lanczos process was used to generate $\mathbf{T}_n$ and $\boldsymbol{\rho}_n$, and the resulting algorithm was termed SyMPVL.

Here, we propose to employ the symmetric band Lanczos process based on coupled recurrences, instead of the standard algorithm, and we call the resulting computational procedure SyMPVL2.

Algorithm 4.1 (SyMPVL2).

```
INPUT: Matrices $\mathbf{C}$, $\mathbf{G} \in \mathbb{R}^{N\times N}$ such that $\mathbf{C}$, $\mathbf{G} \geq \mathbf{0}$ and $\mathbf{G}+s\,\mathbf{C}$ is a regular pencil.
       A matrix $\mathbf{B} \in \mathbb{R}^{N\times m}$.
OUTPUT: Transfer function $\mathbf{Z}_n$ of a reduced-order model of dimension $n$.
    (1) Select expansion point $s_0 \geq 0$ with $\mathbf{G} + s_0\,\mathbf{C} > \mathbf{0}$.
    (2) Compute Cholesky decomposition $\mathbf{G} + s_0\,\mathbf{C} = \mathbf{M}\,\mathbf{M}^{\mathrm{T}}$.
    (3) Run $n\,(\geq m_1)$ iterations of Algorithm 3.1 (applied to $\mathbf{A} := \mathbf{M}^{-1}\,\mathbf{C}\,\mathbf{M}^{-\mathrm{T}}$
        and $\mathbf{R} := \mathbf{M}^{-1}\,\mathbf{B}$) to generate matrices $\mathbf{U}_n$, $\boldsymbol{\Delta}_n$, and $\boldsymbol{\rho}_{m_1}$.
```

(4) *Set*

$$\mathbf{Z}_n(s) = \boldsymbol{\rho}_n^{\mathrm{T}} \left(\mathbf{I}_n + (s - s_0)\, \mathbf{U}_n^{\mathrm{T}}\, \boldsymbol{\Delta}_n\, \mathbf{U}_n\right)^{-1} \boldsymbol{\rho}_n, \quad \textit{where} \quad \boldsymbol{\rho}_n = \begin{bmatrix} \boldsymbol{\rho}_{m_1} \\ \mathbf{0}_{n-m_1 \times m} \end{bmatrix}.$$

Remark 4.1. As we mentioned in section 2.7, the standard band Lanczos process and the algorithm based on coupled recurrences are mathematically equivalent in the case that no deflation occurs or that only exact deflation is performed. Consequently, in this case, SyMPVL and SyMPVL2 are mathematically equivalent. Furthermore, in [13, 18], it was shown that the reduced-order model defined by the transfer function (4.7) is a matrix-Padé model.

4.3. Passivity. Linear dynamical systems of the form (4.1) with $\mathbf{C}$, $\mathbf{G} \geq \mathbf{0}$ are *passive*. Roughly speaking, a system (4.1) is passive if it does not generate energy. In terms of the transfer function (4.3), $\mathbf{Z}$, passivity of (4.1) is equivalent to the *positive realness* of $\mathbf{Z}$; see, e.g., [2, 27]. A precise definition of positive realness is as follows.

Definition 4.2 (positive realness). *An* $(m \times m)$*-matrix-valued rational function* $\mathbf{Z}$ *is called positive real if*

(i) $\mathbf{Z}$ *has no poles in* $\mathbb{C}_+ := \{\, s \in \mathbb{C} \mid \mathrm{Re}(s) > 0 \,\}$*;*

(ii) $\mathbf{Z}(\bar{s}) = \overline{\mathbf{Z}(s)}$ *for all* $s \in \mathbb{C}$*;*

(iii) $\mathrm{Re}\left(\mathbf{x}^{\mathrm{H}}\, \mathbf{Z}(s)\, \mathbf{x}\right) \geq 0$ *for all* $\mathbf{x} \in \mathbb{C}^m$ *and all* $s \in \mathbb{C}_+$*.*

Passivity is a stronger condition than stability of a linear dynamical system. When reduced-order models of passive linear subsystems are used within a simulation of a (not necessarily linear) system, then passivity of the reduced-order models is needed to guarantee stability of the overall simulation; see the references given in [16].

By [16, Theorem 13], a reduced-order model (4.2) with transfer function $\mathbf{Z}_n$ given by (4.3) is passive if $\mathbf{C}_n \geq \mathbf{0}$ and $\mathbf{G}_n \geq \mathbf{0}$. By applying this result to the reduced-order transfer function (4.7), it follows that passivity is guaranteed if

$$\mathbf{T}_n \geq \mathbf{0} \quad \text{and} \quad \mathbf{I}_n - s_0\, \mathbf{T}_n \geq \mathbf{0}. \tag{4.8}$$

In exact arithmetic, the two conditions (4.8) are always fulfilled. Indeed, the first relation in (4.8) follows from $\mathbf{T}_n = \mathbf{V}_n^{\mathrm{T}}\, \mathbf{A}\, \mathbf{V}_n$ and $\mathbf{A} \geq \mathbf{0}$, and the second relation in (4.8) follows from

$$\mathbf{M}\ (\mathbf{I}_N - s_0\, \mathbf{A})\ \mathbf{M}^{\mathrm{T}} = \mathbf{G} \geq \mathbf{0}$$

and

$$\mathbf{I}_n - s_0\, \mathbf{T}_n = \mathbf{V}_n^{\mathrm{T}}\ (\mathbf{I}_N - s_0\, \mathbf{A})\ \mathbf{V}_n \geq \mathbf{0}.$$

Unfortunately, in finite-precision arithmetic, round-off may cause the matrix $\mathbf{T}_n$ generated by the standard band Lanczos process to be indefinite. The negative eigenvalues of $\mathbf{T}_n$ translate into positive poles of $\mathbf{Z}_n$, causing the reduced-order model given by (4.7) to be nonpassive; see the examples given in section 4.4 below. This problem can easily be remedied by employing the SyMPVL2 Algorithm 4.1. Instead of $\mathbf{T}_n$, it generates an $\mathrm{LDL}^{\mathrm{T}}$ factorization, $\mathbf{U}_n^{\mathrm{T}}\, \boldsymbol{\Delta}_n\, \mathbf{U}_n$, of $\mathbf{T}_n$. In view of (2.9) and (2.19), we have $\boldsymbol{\Delta}_n \geq \mathbf{0}$ and thus the first relation in (4.8) is satisfied.

An important practical issue for the SyMPVL2 Algorithm 4.1 is to determine if the reduced-order model $\mathbf{Z}_n$ is a sufficiently good approximation to $\mathbf{Z}$ for some prescribed range of s. By applying the technique in [6] to the reduced-order transfer function $\mathbf{Z}_n$ generated by Algorithm 4.1, it is easy to verify the following result; see [5] for more details. For all s with $|s - s_0| < 1/\,\|\mathbf{A}\|$, we have

$$(4.9)\quad \|\mathbf{Z}(s) - \mathbf{Z}_n(s)\| \;\le\; |s - s_0|^2\, \|\mathbf{F}_m(s - s_0)\|^2\, \frac{\|\widehat{\mathbf{V}}_{m_{\mathrm{c}}}\|^2}{1 - |s - s_0|\;\|\mathbf{A}\|} + \mathcal{O}\left(\mathtt{dtol}_n\right).$$

Here, we set

$$\begin{aligned} \mathbf{F}_n(s - s_0) &:= \left[\, \mathbf{0}_{m_{\mathrm{c}} \times n - m_{\mathrm{c}}} \quad \mathbf{I}_{m_{\mathrm{c}}} \,\right] \left(\mathbf{I}_n + (s - s_0)\, \mathbf{U}_n^{\mathrm{T}}\, \boldsymbol{\Delta}_n\, \mathbf{U}_n\right)^{-1} \begin{bmatrix} \boldsymbol{\rho}_m^{\mathrm{T}} \\ \mathbf{0}_{n-m\times m} \end{bmatrix}, \\ \widehat{\mathbf{V}}_{m_{\mathrm{c}}} &:= \left[\, \hat{\mathbf{v}}_{n+1} \quad \hat{\mathbf{v}}_{n+2} \quad \cdots \quad \hat{\mathbf{v}}_{n+m_{\mathrm{c}}} \,\right]. \end{aligned}$$

In practice, we drop the last term, $\mathcal{O}\left(\mathtt{dtol}_n\right)$, in the error bound (4.9), and evaluate only the remaining terms. The $(m_{\mathrm{c}} \times m)$-matrix-valued function $\mathbf{F}_m(s - s_0)$ can be computed explicitly, and sufficiently good approximations of the norms $\|\mathbf{F}_m(s - s_0)\|$ and $\|\widehat{\mathbf{V}}_{m_{\mathrm{c}}}\|$ can easily be obtained, for instance, using the Matlab function `normest`. The norm $\|\mathbf{A}\|$ can be estimated by using the largest eigenvalue of the projected Lanczos matrix $\mathbf{U}_n^{\mathrm{T}}\, \boldsymbol{\Delta}_n\, \mathbf{U}_n$; see, e.g., [20, section 9.1.4].

We have performed extensive numerical tests with SyMPVL2 using the convergence check based on evaluating the error bound (4.9). We found that, typically, the computational costs for this convergence check is about 5% to 8% of the cost of the underlying symmetric band Lanczos process.

4.4. Numerical examples. In this subsection, we present results of three numerical examples that demonstrate the superiority, in terms of both robustness and accuracy, of the SyMPVL2 Algorithm 4.1 based on coupled recurrences over the original SyMPVL algorithm [17, 18] based on the standard symmetric band Lanczos process.

All three examples are passive linear dynamical systems (4.1) describing RC networks arising in VLSI circuit simulation. The frequency range of interest is always $s = 2\pi\,\mathtt{i}\,\omega$, where $1 \le \omega \le 10^9$. The goal is to compute a reduced-order transfer function $\mathbf{Z}_n$ that approximates the transfer function $\mathbf{Z}$ of (4.1) well in this frequency range. In all three examples, $\mathbf{C} \ge \mathbf{0}$, $\mathbf{G} > 0$, and the expansion point $s_0 = 0$ is used. All experiments were performed in Matlab. The symmetric band Lanczos process was run with deflation tolerance (3.2), where $\mathtt{dtol} = \sqrt{\mathtt{eps}}$ and `eps` is the machine precision.

Example 4.1. In this example, $\mathbf{C}$ and $\mathbf{G}$ are matrices of order $N = 1346$, and $m = 10$. Both SyMPVL and SyMPVL2 required $n = 60$ iterations. However, the reduced-order model generated via SyMPVL is not passive, and not even stable. The reason is that the computed matrix $\mathbf{T}_{60}$ is indefinite, causing some positive poles of the transfer function $\mathbf{Z}_{60}$. On the other hand, SyMPVL2 generates a diagonal matrix $\boldsymbol{\Delta}_{60}$ with only positive diagonal entries, and the matrix $\mathbf{U}_{60}^{\mathrm{T}}\, \boldsymbol{\Delta}_{60}\, \mathbf{U}_{60}$ is positive definite. In particular, the transfer function $\mathbf{Z}_{60}$ does not have any positive poles. In Figure 1, we show the dominant poles of the reduced-order models obtained from SyMPVL and SyMPVL2, as well as the dominant poles of the transfer function $\mathbf{Z}$ of the original linear dynamical system. Note that SyMPVL not only generated two unstable poles, but also that one of the negative poles close to zero is wrong. In Figure 2, we plot the SyMPVL and SyMPVL2 errors $\|\mathbf{Z}(s) - \mathbf{Z}_{60}(s)\|$ for all $s = 2\pi\,\mathtt{i}\,\omega$, $1 \le \omega \le 10^9$. Clearly, the SyMPVL2 model is also more accurate than the SyMPVL model.

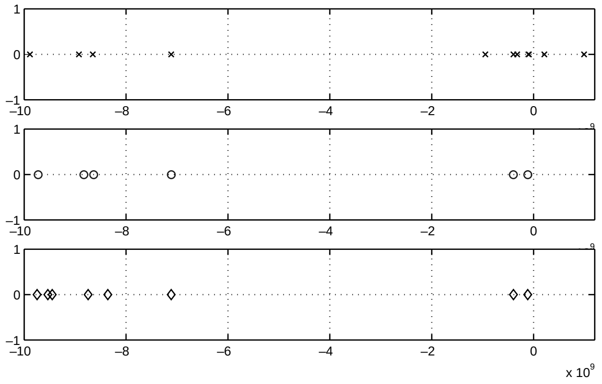


FIG. 1. *Dominant poles of SyMPVL reduced-order model* (*top*), *SyMPVL*2 *reduced-order model* (*middle*), *and the exact dominant poles* (*bottom*) *for Example* 4.1.

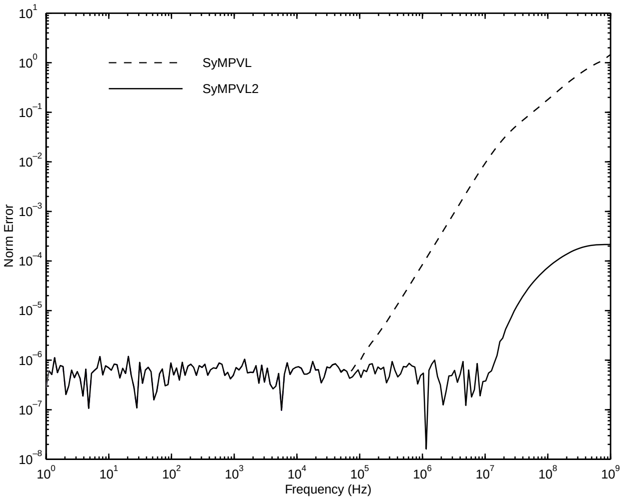


FIG. 2. *Error* $\|\mathbf{Z}(s) - \mathbf{Z}_n(s)\|$ *of SyMPVL and SyMPVL*2 *models for Example* 4.1.

Example 4.2. In this example, $\mathbf{C}$ and $\mathbf{G}$ are matrices of order $N = 13875$, and $m = 150$. A reduced-order model of dimension $n = 300$ is needed to achieve convergence in the frequency range of interest. Again, SyMPVL produced an indefinite matrix $\mathbf{T}_{300}$, resulting in some positive poles of $\mathbf{Z}_{300}$, while the reduced-order model generated via SyMPVL2 has no positive poles and is indeed passive. In Figure 3, we show the dominant poles of the reduced-order models obtained from SyMPVL and SyMPVL2.

Example 4.3. This example illustrates the behavior of the error bound (4.9). We use the same matrices $\mathbf{C}$ and $\mathbf{G}$ as in Example 4.2, but now $\mathbf{B}$ has only $m = 50$ columns. The left plot in Figure 4 shows the norm of the reduced-order transfer function $\mathbf{Z}_{300}(2\pi\, \mathtt{i}\, \omega)$ for the frequency range $1 \leq \omega \leq 10^9$. The right plot in Figure 4 shows the upper bound (4.9) for the corresponding error norm $\|\mathbf{Z}(s) - \mathbf{Z}_n(s)\|$ obtained after $n = 100$, 200, and 300 iterations of Algorithm 3.1.

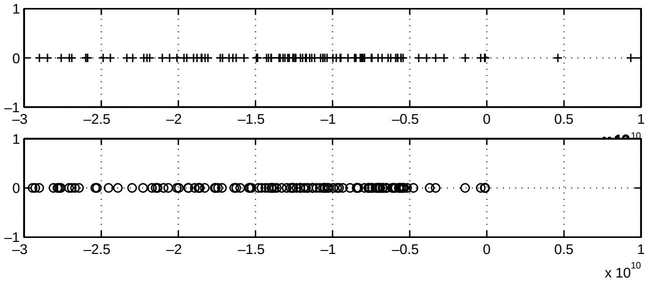


FIG. 3. *Dominant poles of SyMPVL reduced-order model* (*top*) *and SyMPVL*2 *reduced-order model* (*bottom*) *for Example* 4.2.

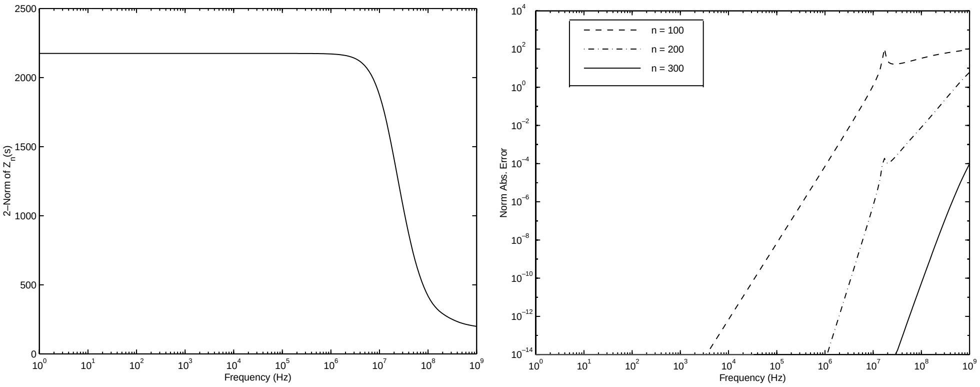


FIG. 4. $\|\mathbf{Z}_{300}(s)\|$ (*left*) *and upper bounds for* $\|\mathbf{Z}(s) - \mathbf{Z}_n(s)\|$ (*right*) *for Example* 4.3.

5. Application to eigenvalue computations. In this section, we present some preliminary numerical examples to illustrate the application of Algorithm 3.1 to the solution of generalized symmetric definite eigenvalue problems of the form

$$\mathbf{K}\,\mathbf{x} = \lambda\,\mathbf{M}\,\mathbf{x}, \tag{5.1}$$

where $\mathbf{K} = \mathbf{K}^{\mathrm{T}} \in \mathbb{R}^{N\times N}$, $\mathbf{M} = \mathbf{M}^{\mathrm{T}} \in \mathbb{R}^{N\times N}$, and $\mathbf{M} > \mathbf{0}$.

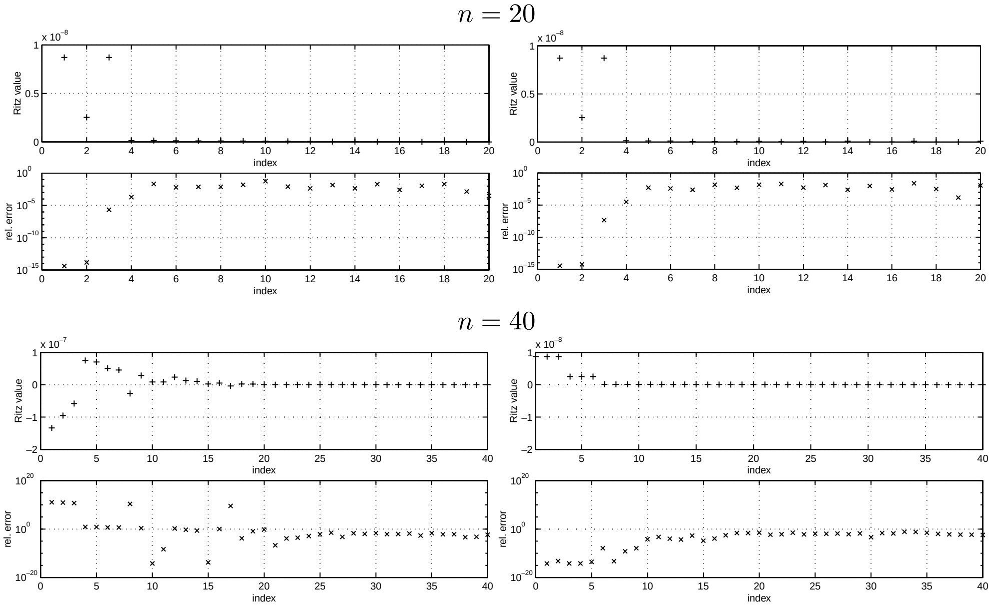
The usual approach is first to compute a Cholesky decomposition $\mathbf{M} = \mathbf{L}\,\mathbf{L}^{\mathrm{T}}$ of $\mathbf{M}$, and then convert (5.1) to the standard eigenvalue problem

$$\left(\mathbf{L}^{-1}\,\mathbf{K}\,\mathbf{L}^{-\mathrm{T}}\right)\,\mathbf{y} = \lambda\,\mathbf{y}, \quad \text{where} \quad \mathbf{y} := \mathbf{L}^{\mathrm{T}}\,\mathbf{x}. \tag{5.2}$$

The band Lanczos process is then applied to the symmetric matrix $\mathbf{A} := \mathbf{L}^{-1}\,\mathbf{K}\,\mathbf{L}^{-\mathrm{T}}$ and a random block $\mathbf{B} \in \mathbb{R}^{N\times m}$ of m starting vectors. Note that the Lanczos process requires only matrix-vector products with $\mathbf{A}$. These can be computed by means of multiplications with $\mathbf{K}$ and backsolves with $\mathbf{L}$ and $\mathbf{L}^{\mathrm{T}}$, without explicitly forming $\mathbf{A}$.

Next, we present two numerical examples. These experiments were performed in Matlab, and in both cases $m = 2$ starting vectors were used. In both examples, $\mathbf{K} \geq \mathbf{0}$, and hence the eigenvalues λ_k, $1 \leq k \leq N$, of (5.2) are all real and nonnegative.

Example 5.1. The matrices $\mathbf{K}$ and $\mathbf{M}$ in this example are of order $N = 1346$, and are taken from an application in circuit simulation. We ran both the standard symmetric band Lanczos process and the coupled Algorithm 3.1 for $n = 20$ and $n = 40$



$n = 20$

$n = 40$

FIG. 5. *The standard* (*left*) *and the coupled* (*right*) *band Lanczos process for Example* 5.1.

iterations. The Ritz values are then computed as the eigenvalues of the matrices $\mathbf{T}_n$ (for the standard algorithm) and $\mathbf{U}_n^{\mathrm{T}}\,\boldsymbol{\Delta}_n\,\mathbf{U}_n$ (for Algorithm 3.1). In Figure 5, we plot the computed Ritz values θ_i and their relative errors vs. their index i, $i = 1, 2, \ldots, n$, for both variants of the Lanczos process, and for $n = 20$ and 40. Here, the relative error of a computed Ritz value θ_i is defined as

$$\frac{\min_{1\le k\le N} |\theta_i - \lambda_k|}{|\lambda_{k_i}|}, \quad \text{where} \quad k_i := \operatorname{argmin}_{1\le k\le N} |\theta_i - \lambda_k|. \tag{5.3}$$

Note that for $n = 20$, the computed Ritz values and their relative errors for both variants of the band Lanczos process are essentially the same. However, for $n = 40$, the standard algorithm is significantly worse than the coupled algorithm, and even has generated negative Ritz values. On the other hand, all the Ritz values from the coupled Algorithm 3.1 are positive.

Example 5.2. In this example, **K** and **M** are 834 × 834 stiffness and mass matrices arising in a structural analysis within MSC's NASTRAN application. In Figure 6, we show the computed Ritz values, as well as their relative errors (5.3), that were obtained after $n = 40$ iterations of both variants of the band Lanczos process. Note that the standard algorithm produced one negative Ritz value, namely, θ_{30}, while all Ritz values from the coupled Algorithm 3.1 are positive.

6. Concluding remarks. We proposed a variant of the band Lanczos process for symmetric matrices and multiple starting vectors that uses coupled recurrences involving two sets of basis vectors, instead of the recurrences involving only one set in the standard algorithm. The new variant generates the factors of an LDL$^{\mathrm{T}}$ factorization of the Lanczos matrix, rather than the Lanczos matrix directly. Numerical experiments suggest that the coupled algorithm is superior to the standard algorithm both in terms of accuracy and robustness. However, a precise round-off error analysis

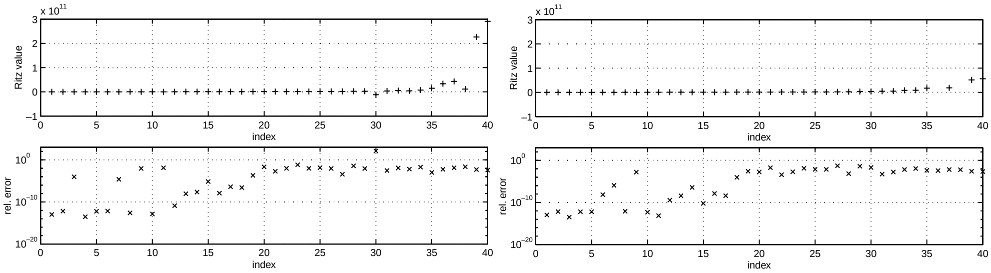


FIG. 6. *The standard* (*left*) *and the coupled* (*right*) *band Lanczos process for Example* 5.2.

that would provide a theoretical basis for the superiority of the coupled algorithm still remains to be done.

For a single starting vector, the symmetric band Lanczos process reduces to the classical symmetric Lanczos algorithm [23] and the Lanczos matrix is tridiagonal. In the last few years, it has gradually become clear that the standard representation of a tridiagonal matrix via its entries is an unfortunate one, and that it is better, for both accuracy and efficiency, to represent the matrix as a product of bidiagonals; see, e.g., [11, 24, 25]. This paper has demonstrated that the same is true for the Lanczos matrix associated with the symmetric band Lanczos process. Instead of representing that matrix via its entries, it is preferable to present it via the entries of an $\mathrm{LDL}^{\mathrm{T}}$ factorization.

Acknowledgments. We would like to thank Peter Feldmann for his help with some of the numerical examples. We are also grateful to an anonymous referee, whose constructive comments helped us to improve the presentation of the paper.

REFERENCES

[1] J. I. Aliaga, D. L. Boley, R. W. Freund, and V. Hernández, *A Lanczos-type method for multiple starting vectors*, Math. Comp., 69 (2000), pp. 1577–1601.

[2] B. Anderson and S. Vongpanitlerd, *Network Analysis and Synthesis*, Prentice-Hall, Englewood Cliffs, NJ, 1973.

[3] B. D. O. Anderson and J. B. Moore, *Optimal Filtering*, Prentice-Hall, Englewood Cliffs, NJ, 1979.

[4] Z. Bai, P. Feldmann, and R. W. Freund, *How to make theoretically passive reduced-order models passive in practice*, in Proceedings of the IEEE Custom Integrated Circuits Conference, Piscataway, NJ, 1998, pp. 207–210.

[5] Z. Bai and R. W. Freund, *A Symmetric Band Lanczos Process Based on Coupled Recurrences and Some Applications*, Numerical Analysis Manuscript 00–3–04, Bell Laboratories, Murray Hill, NJ, 2000. Available online from http://cm.bell-labs.com/cs/doc/00.

[6] Z. Bai and Q. Ye, *Error estimation of the Padé approximation of transfer functions via the Lanczos process*, Electron. Trans. Numer. Anal., 7 (1998), pp. 1–17.

[7] G. A. Baker, Jr., and P. Graves-Morris, *Padé Approximants*, 2nd ed., Cambridge University Press, New York, 1996.

[8] R. H. Battin, *An Introduction to The Mathematics and Methods of Astrodynamics*, American Institute of Aeronautics and Astronautics, New York, 1987.

[9] J. K. Cullum and W. E. Donath, *A block Lanczos algorithm for computing the q algebraically largest eigenvalues and a corresponding eigenspace for large, sparse symmetric matrices*, in Proceedings of the 1974 IEEE Conference on Decision and Control, New York, 1974, IEEE Computer Society Press, Los Alamitos, CA, 1974, pp. 505–509.

[10] J. K. Cullum and R. A. Willoughby, *Lanczos Algorithms for Large Symmetric Eigenvalue Computations, Volume* 1, *Theory*, Birkhäuser, Basel, Switzerland, 1985.

[11] I. S. Dhillon, *A New $O(n^2)$ Algorithm for the Symmetric Tridiagonal Eigenvalue/Eigenvector Problem*, Ph.D. thesis, Department of Mathematics, University of California, Berkeley, CA, 1997.

[12] P. Feldmann and R. W. Freund, *Reduced-order modeling of large linear subcircuits via a block Lanczos algorithm*, in Proceedings of the 32nd ACM/IEEE Design Automation Conference, New York, 1995, pp. 474–479.

[13] R. W. Freund, *Computation of matrix Padé approximations of transfer functions via a Lanczos-type process*, in Approximation Theory VIII, Vol. 1: Approximation and Interpolation, C. Chui and L. Schumaker, eds., World Scientific, Singapore, 1995, pp. 215–222.

[14] R. W. Freund, *Reduced-order modeling techniques based on Krylov subspaces and their use in circuit simulation*, in Applied and Computational Control, Signals, and Circuits, Vol. 1, B. N. Datta, ed., Birkhäuser, Boston, 1999, pp. 435–498.

[15] R. W. Freund, *Band Lanczos method*, in Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide, Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, eds., SIAM, Philadelphia, 2000, pp. 80–88.

[16] R. W. Freund, *Krylov-subspace methods for reduced-order modeling in circuit simulation*, J. Comput. Appl. Math., 123 (2000), pp. 395–421.

[17] R. W. Freund and P. Feldmann, *The SyMPVL algorithm and its applications to interconnect simulation*, in Proceedings of the IEEE International Conference on Simulation of Semiconductor Processes and Devices, Piscataway, NJ, 1997, pp. 113–116.

[18] R. W. Freund and P. Feldmann, *Reduced-order modeling of large linear passive multi-terminal circuits using matrix-Padé approximation*, in Proceedings of the Design, Automation, and Test in Europe Conference, IEEE Computer Society Press, Los Alamitos, CA, 1998, pp. 530–537.

[19] R. W. Freund and N. M. Nachtigal, *An implementation of the QMR method based on coupled two-term recurrences*, SIAM J. Sci. Comput., 15 (1994), pp. 313–337.

[20] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed., The Johns Hopkins University Press, Baltimore, MD, 1996.

[21] M. H. Gutknecht and Z. Strakoš, *Accuracy of two three-term and three two-term recurrences for Krylov space solvers*, SIAM J. Matrix Anal. Appl., 22 (2000), pp. 213–229.

[22] S. Hammarling, *Numerical solution of the stable, non-negative definite Lyapunov equation*, IMA J. Numer. Anal., 2 (1982), pp. 303–323.

[23] C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, J. Research Nat. Bur. Standards, 45 (1950), pp. 255–282.

[24] B. N. Parlett, *The new QD algorithm*, Acta Numer., 4 (1995), pp. 459–491.

[25] B. N. Parlett and K. Fernando, *Accurate singular values and differential QD algorithms*, Numer. Math., 67 (1994), pp. 191–229.

[26] A. Ruhe, *Implementation aspects of band Lanczos algorithms for computation of eigenvalues of large sparse symmetric matrices*, Math. Comp., 33 (1979), pp. 680–687.

[27] M. Wohlers, *Lumped and Distributed Passive Networks*, Academic Press, New York, 1969.

SIAM J. Sci. Comput.
Vol. 23, No. 2, pp. 563–582

LARGE-SCALE NORMAL COORDINATE ANALYSIS FOR MOLECULAR STRUCTURES*

CHAO YANG[†], BARRY W. PEYTON[‡], DONALD W. NOID[§], BOBBY G. SUMPTER[§], AND ROBERT E. TUZUN[¶]

Abstract. We apply truncated RQ-iteration (TRQ) and the Jacobi–Davidson (JD) method to perform vibrational (eigenvalue) analysis for large-scale molecular systems. Both algorithms employ a preconditioned iterative solver to construct a low-dimensional subspace that contains desired vibrational modes. We discuss several strategies for speeding up the eigenvalue calculation. In particular, we illustrate how to construct effective preconditioners and analyze the quality of these preconditioners. We show that convergence can be improved by choosing appropriate shifts and deflating the translational and rotational modes. Numerical examples are provided to demonstrate the efficiency of our computation.

Key words. normal coordinate analysis, eigenvalue computation, preconditioner

AMS subject classifications. Primary, 65F15; Secondary, 65G05

PII. S1064827500373668

1. Introduction.

Normal coordinate analysis (NCA) [17, 26] plays an important role in the study of vibrational and thermal properties of various molecular structures at the atomic level. The molecular vibration at low temperatures can be analyzed by examining a collection of fundamental (normal) vibrational modes. These modes are essentially eigenvectors of a matrix that describes the atomic force interaction within the molecular system under study. Once NCA information is obtained, it becomes easier to link theoretical models to experimentally accessible macroscopic data to provide a straightforward interpretation of molecular events in the absence of classical chaos. The immediate applications of NCA include characterizing thermal stability of polymer materials [7] and assessing the dynamic role protein plays in the photosynthetic center of green plants [19]. The long-term benefits include the development of new materials that can be used for nano-manufacturing and bioengineering [8].

Until recently, the use of NCA has been limited to relatively small systems. This is partly due to the lack of efficient numerical algorithms and the limited computing

*Received by the editors June 13, 2000; accepted for publication (in revised form) November 5, 2000; published electronically July 10, 2001. This research was sponsored by Applied Mathematical Science Program, Office of Science, and Division of Material Sciences, Office of Basic Energy Sciences, U.S. Department of Energy under contract DE-AC05-00OR22725 with UT-Battelle, LLC. The U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes. Copyright is owned by SIAM to the extent not limited by these rights.

http://www.siam.org/journals/sisc/23-2/37366.html

[†]National Energy Research Scientific Computing Center, Lawrence Berkeley National Laboratory, Building 50F, 1 Cyclotron Road, Berkeley, CA 94720 (cyang@lbl.gov). This research was carried out while the author was employed by Oak Ridge National Laboratory and supported by an Alston S. Householder postdoctoral fellowship.

[‡]Computer Science & Mathematics Division, Oak Ridge National Laboratory, P.O. Box 2008, Oak Ridge, TN 37381-6367 (peytonbw@ornl.gov).

[§]Chemical & Analytical Sciences Division, Oak Ridge National Laboratory, P.O. Box 2008, Oak Ridge, TN 37381-6367 (noiddw@ornl.gov, sumpterbg@ornl.gov).

[¶]Department of Computational Science, State University of New York at Brockport, Brockport, NY 14420-2914 (rtuzun@brockport.edu). This author is supported by a Petroleum Research Fund administered by the American Chemical Society.

power on traditional sequential and parallel machines. In [16], we presented a numerical scheme that was based on a shift-invert Lanczos iteration. We reported the success of using this scheme to compute the lowest 100 vibrational modes for a 6000-atom (18000 degrees of freedom) polyethylene particle. This computation was performed on a single processor of NEC SX-4, a powerful vector machine with 2 Gflops peak performance and 2 GB of main memory. Although the performance we achieved on the SX-4 is quite satisfactory, the large amount of memory (450 MB) required to carry out this calculation makes it difficult to extend the current computation to larger molecular systems. To solve problems that are 10 or 100 times larger on existing machines, we must modify the algorithm to reduce memory usage.

Although force field modeling and calculation are crucial components of NCA, extracting eigenvalues and eigenvectors from the force interaction matrix constitutes the majority of the computational work. Therefore, the efficiency of NCA relies on fast algorithms for solving matrix eigenvalue problems. As will be shown in section 2, the force interaction matrix encountered in NCA is sparse, symmetric, and positive (semi-)definite.

In this paper we investigate the possibility of applying two recently developed algorithms—the truncated RQ-iteration (TRQ) [23, 27] and the Jacobi–Davidson (JD) method [20, 4]—to perform large-scale NCA. Both algorithms require far less memory than a shift-inverted Lanczos iteration. The experiments shown in section 6 demonstrate that these methods are quite effective in capturing the desired vibrational modes of a large-scale polymer system.

The paper is organized as follows. In the next section, we provide some mathematical background for NCA. Various approaches to the solution of large-scale eigenvalue problems are discussed in section 3. We point out the limitation of the Lanczos-type algorithms for this particular problem. We review TRQ and JD in section 4. The efficiency of TRQ and JD depends largely on the quality of preconditioners used to solve a linear system with a special structure. We discuss the construction of effective preconditioners and the use of deflation techniques in section 5. We show numerical results in section 6 to demonstrate the efficiency of these methods on NCA problems and point out additional techniques that can be used to improve the convergence rate. Finally, we give a few concluding remarks in section 7.

2. Problem formulation. The dynamics of a molecular system consisting of m atoms can be described by

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_j}\right) + \left(\frac{\partial V}{\partial q_j}\right) = 0,$$

where T and V are total kinetic and potential energy of the system, respectively, and q_j is the mass adjusted displacement of an atom in one of the Cartesian coordinate directions. If we assume molecules vibrate near an equilibrium configuration, the equation of motion can be simplified to yield

$$\ddot{q} + Fq = 0,$$

where F is obtained by taking the second derivative of the potential with respect to the Cartesian coordinates, i.e.,

$$F_{i,j} = \frac{\partial^2 V}{\partial q_i \partial q_j}.$$

The standard technique for solving this second-order ODE is first to diagonalize F by computing its eigenvalues and eigenvectors. This allows us to decouple the system and solve each equation independently. Each eigenvector of F is called a normal (vibrational) mode and the corresponding eigenvalue is proportional to the vibrational frequency associated with that particular mode. This procedure of decomposing molecular vibration into a number of linearly independent modes is thus called normal coordinate (mode) analysis. It is well known that the near-equilibrium motion can often be captured by a linear combination of several low-frequency modes. Thus, our primary interest is to compute a number of smallest eigenvalues and corresponding eigenvectors of F.

Throughout this paper, we will focus on performing NCA on computer-generated polyethylene (PE) particles of various sizes. These particles are named as *pe*3*k*, *pe*6*k*, *pe*12*k*, and *pe*24*k*, where the trailing number represents the number of atoms in the particle. A 6000-atom particle model (*pe*6*k*) is shown in Figure 2.1. The structure of this model closely resembles droplet streams that can be generated experimentally [1, 11]. An m-atom particle involves $n = 3m$ degrees of freedom. Thus the dimension of the force interaction matrix F for the 6000-atom model is 18000.

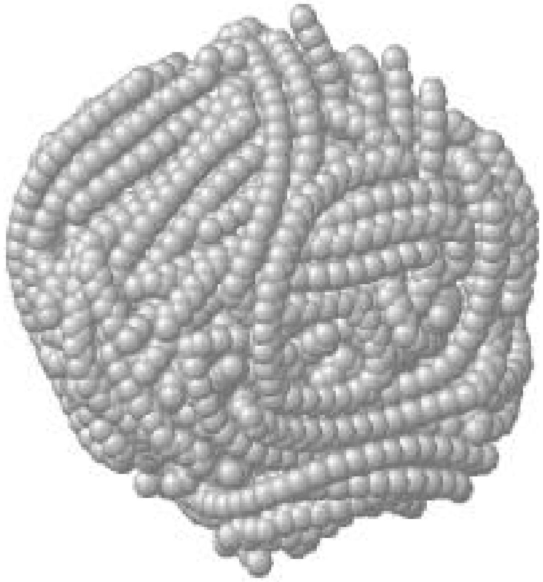


FIG. 2.1. *The structure of the pe6k particle.*

The potential field of PE particles is well understood. The interactions between atoms are categorized as either bonded or nonbonded. These interactions are often measured in terms of various angles and internal stretches. For example, the bonded interaction contributed by the ith atom can be expressed by

$$\begin{aligned} V_b^i(r_{i,i+1}, \theta_{i,i+1,i+2}, \tau_{i,i+1,i+2,i+3}) &= \frac{1}{2}k_r(r_{i+1,i} - r_0)^2 \\ &+ \frac{1}{2}k_\theta(\theta_{i,i+1,i+2} - \theta_0)^2 \\ &+ 8.77 + \alpha \cos \tau_{i,i+1,i+2,i+3} + \beta \cos^3 \tau_{i,i+1,i+2,i+3}, \end{aligned}$$

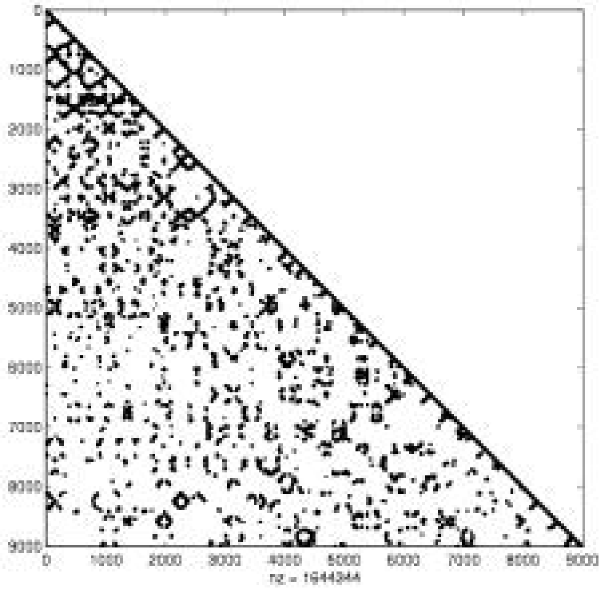


FIG. 2.2. *The nonzero pattern of the lower triangular part of F for pe3k.*

where k_r, k_θ, α, and β are force constant parameters and r_0 and θ_0 are equilibrium values of the bond length and angle. The 2-atom bonded stretch $r_{i,i+1}$ measures the displacement along the line segment connected by atom i and $i+1$. Atoms i, $i+1$, and $i+2$ form a bending angle $\theta_{i,i+1,i+2}$ centered at atom $i+1$. The quantity $\tau_{i,i+1,i+2,i+3}$ represents a 4-atom torsional angle.

The interaction between two atoms not directly bonded is represented by the following Lennard–Jones type of potential function:

$$\hat{V}_{nb} = 4\epsilon\left[\left(\frac{\sigma}{r_{ij}}\right)^{12} - \left(\frac{\sigma}{r_{ij}}\right)^{6}\right],$$

where r_{ij} is the distance between atom i and j, and σ and ϵ are Lennard–Jones parameters. The nonbonded interaction is often neglected when the distance between two atoms is beyond a certain threshold. The cutoff distance we used for PE particle in our experiment is 10 angstrom.

To form the force interaction matrix, one must calculate the derivatives of V with respect to the Cartesian coordinates (x,y,z). The brute force approach of expressing V in terms of x, y, and z before taking derivatives is very time-consuming. A more efficient computational scheme is presented in [24] to speed up the calculation. The details on efficient derivative calculation is beyond the scope of this paper. We refer readers to [24, 25] for more information.

Because the long-range nonbonded interactions are neglected, the force constant matrix is sparse. Since the second derivative of our potential is independent of the order in which it is taken, F is symmetric. The nonzero structure of the lower triangular portion of F for a 3000-atom PE particle is shown in Figure 2.2. The sparsity of F allows us to perform the matrix-vector mulitplication $y \leftarrow Fx$ efficiently.

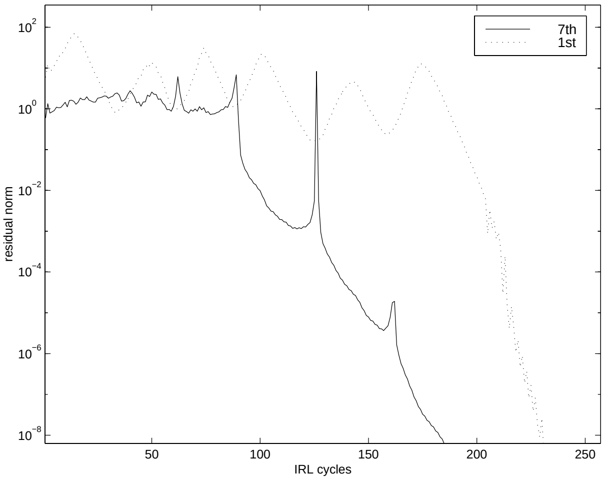


FIG. 3.1. *The relative residual history of the* 1*st and* 7*th approximate eigenpairs of* F *computed by IRL.*

3. Implicitly restarted Lanczos and shift-invert. Because F is fairly sparse, and because we are only interested in a small number of eigenpairs of F (around 100), it is natural to apply the Lanczos algorithm [12] to calculate the desired vibrational modes. The implicit restarting mechanism [21] implemented in the ARPACK software [13] allows us to further reduce memory usage by maintaining and repeatedly updating a *Krylov subspace* of low dimension. The approximate eigenpairs of F are obtained by projecting F into this low-dimensional subspace and solving a smaller eigenvalue problem.

Our first attempt at using ARPACK to perform large-scale NCA turned out to be somewhat disappointing. In Figure 3.1, we plot the convergence history of the first and the seventh approximate eigenpairs (θ_1, u_1) and (θ_7, u_7). Each curve records relative residual norms

$$\frac{\|Fu_i - \theta_i u_i\|}{|\theta_i|}$$

measured at the end of each implicit restarting cycle. The dimension of the Krylov subspace constructed in the implicitly restarted Lanczos (IRL) process is $\mathtt{ncv} = 100$. For this particular run, we requested $\mathtt{nev} = 50$ eigenpairs. The convergence tolerance is set to $\mathtt{etol} = 10^{-8}$. Thus, the number of matrix-vector products used in each IRL cycle is roughly 50. At the end of 300 IRL cycles, only 20 eigenpairs converged. The residual curves show some irregular bumps during the course of convergence. This phenomenon usually indicates that IRL detects nearby eigenpairs first before it converges to the desired one. It is not difficult to see, after examining the full spectrum of F, why IRL converges slowly. We observe from Figure 3.2 that the relative gaps between adjacent eigenvalues at the low end of the spectrum are tiny. The lowest six eigenvalues are nearly zero. The small magnitude of these eigenvalues is expected. In fact, if F is formed exactly, these eigenvalues should be exactly zero. The corresponding eigenvectors represent the pure translational and rotational motion of the entire molecular system. It is well known that small relative gaps between eigenvalues at the low end of the spectrum make it harder for IRL to construct a

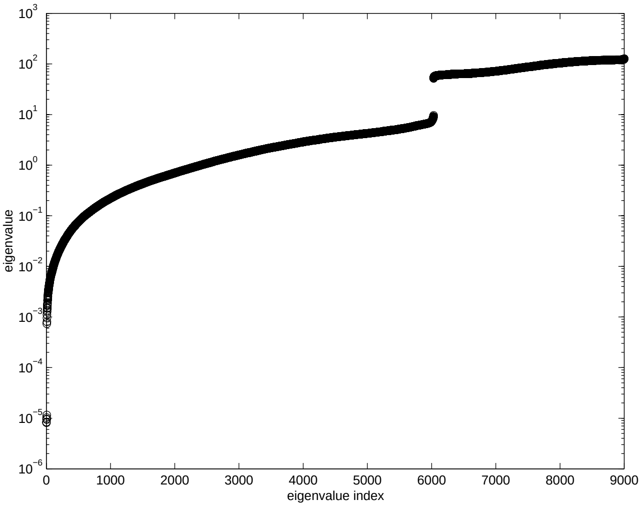


FIG. 3.2. *The full spectrum of pe3k.*

polynomial filter that can effectively separate the desired vibrational modes from the unwanted ones [13].

A commonly used strategy for speeding up the calculation of clustered or interior eigenvalues is to apply IRL to the shifted and inverted matrix $(F - \sigma I)^{-1}$, where σ is a target shift near the eigenvalues of interest. Instead of performing a sparse matrix-vector multiplication, we must solve sparse triangular systems at each Lanczos step using the sparse triangular factors computed in advance. The shifted matrix is often decomposed as $F - \sigma I = LDL^T$, where L is unit lower triangular and D is diagonal. The matrix L contains nonzero "fill-ins" produced by Gaussian elimination and thus requires more memory than that for F. Although significant progress has been made in the past few years on direct algorithms for solving large sparse linear systems [3, 9], decomposing $F - \sigma I$ into sparse triangular factors and solving sparse triangular systems still constitute a large portion of the computational effort in shift-invert Lanczos. However, our experiments show that, at least for problems of moderate size, shift-invert Lanczos is quite efficient compared to the straightforward application of IRL to F. In Table 3.1, we list CPU seconds and memory usage required to compute the lowest 100 vibrational modes of *pe*6*k*. The computation is carried out on a NEC SX-4 vector processor with 2 Gflop peak performance and 8 GB of shared memory. We observe that shift-invert is almost 20 times faster than the direct usage of IRL. Unfortunately, the large amount of memory required to store L makes it more difficult to perform NCA for larger molecular systems on current high-performance computers. Table 3.2 shows the CPU time and memory required to perform NCA on PE particles with various sizes. The first column of the table represents the dimension of F. A multilevel nested dissection matrix ordering scheme is used to reduce the amount of fill-ins in the sparse matrix factorization [10]. We observe that more than 3 GB of memory is needed to carry out NCA for a PE particle with 24000 atoms. Multifrontal codes require even more memory for intermediate stacks. This is quite demanding for a problem that is still relatively small compared to what we would like to tackle (molecular systems with 300,000 atoms).

An alternative to shift-invert is to apply IRL to $p(F)$, where $p(\lambda)$ is a polynomial

Table 3.1

The CPU time and memory required to compute the lowest 100 *vibrational modes of a* 6000-*atom PE particle.*

Method	CPU time (seconds)	Memory (MB)
IRL	9204	116
Shift-invert	515	459
Polynomial	2850	116

Table 3.2

The CPU time and memory required to compute the lowest 100 *vibrational modes of PE particles of various sizes.*

n	CPU time (seconds)	Memory (MB)
9000	145	224
18000	515	459
36000	1472	1333
72000	3432	3213

constructed to enhance the eigenvalue distribution. Typically, $p(\lambda)$ is chosen to map the small eigenvalues F to dominant eigenvalues of $p(F)$ [22]. However, when the condition number of F is large, it is difficult to find a low degree polynomial that can effectively separate the desired eigenvalues from the unwanted ones. Nevertheless, Table 3.1 indicates that polynomial transformation appears to be a reasonable compromise between the direct-use IRL and shift-invert IRL.

Another potential way to overcome the memory limitation is to parallelize the eigenvalue calculation on distributed-memory multiprocessors or a network of workstations [28]. We will not discuss parallelization issues in this paper.

4. JD and TRQ.

In the last few years, several algorithms [4, 20, 23] have been developed to address the computation of clustered or interior eigenvalues of large-scale problems without performing complete sparse matrix factorizations. The JD algorithm [4, 20] and the inexact TRQ (ITRQ) iteration [23, 27] are among the most successful ones. Both algorithms use preconditioned iterative solvers to generate and update a subspace similar to the one that is produced by a shift-invert Lanczos iteration. Neither method requires linear systems to be solved to high accuracy. In JD, the approximation subspace is no longer a Krylov subspace, and basis vectors are generated through a successive eigenvector correction process. The ITRQ iteration uses a preconditioned iterative solver to modify the starting vector of a restarted Lanczos process so that the leading columns of the Lanczos basis matrix converge quickly to desired eigenvectors. We review some of the basic concepts of these methods in this section.

4.1. The JD algorithm.

Suppose an approximate eigenvector $u \in \mathbb{R}^{n\times 1}$, $\|u\| = 1$, and its associated Rayleigh quotient

$$\theta = u^T F u$$

have been computed. One way to improve the current approximation is to seek a correction pair (γ, z), where $\gamma \in \mathbb{R}$ and $z \in \mathbb{R}^{n\times 1}$, such that

$$F(u+z) = (\theta+\gamma)(u+z) \quad \text{and} \quad u^T z = 0. \tag{4.1}$$

It follows that

$$(F - \theta I)z - \gamma u = -r + \gamma z,$$

```
Algorithm. Jacobi–Davidson.

Input: jmin, jmax, matrix $F$, initial eigenvector approximation $u$.
Output: approximate eigenpair $(\theta, u)$ such that $\|Fu - \theta u\|$ is small.

1. $u \leftarrow u/\|u\|$, $\theta \leftarrow u^H Fu$; converged $\leftarrow$ FALSE;
2. $V \leftarrow (u)$; $H \leftarrow (\theta)$; jstart $\leftarrow 1$;
3. while not converged do
  3.1 for $j =$ jstart, jstart + 1, ..., jmax
    3.1.1 Solve the correction equation (4.2) to obtain $z$;
    3.1.2 $z \leftarrow (I - VV^H)z$;
    3.1.3 $V \leftarrow (V, z)$; $H \leftarrow V^H FV$;
    3.1.4 Compute all eigenpairs of $H$, and select a desired pair $(\theta, s)$;
    3.1.5 Put $u \leftarrow Vs$;
    3.1.6 if $\|Fu - \theta u\| <$ some tolerance, converged $\leftarrow$ TRUE, goto 4;
  3.2 end for;
  3.3 Restart: replace $V$ with jmin desired Ritz vectors;
  3.4 $H \leftarrow V^H FV$; jstart $\leftarrow$ jmin $+ 1$; goto 3;
4. end while;
```

Fig. 4.1. *The JD iteration.*

where $r = Fu - \theta u$. If we drop the second order correction γz, z and γ can be determined by solving the following linear equation:

$$\begin{pmatrix} F - \theta I & u \\ u^T & 0 \end{pmatrix} \begin{pmatrix} z \\ -\gamma \end{pmatrix} = \begin{pmatrix} -r \\ 0 \end{pmatrix}. \tag{4.2}$$

The solution to this equation may be viewed as a Newton correction to the initial approximation to the solution of the nonlinear equation (4.1). Once z is computed, it can be adjoined to the approximate eigenvector u to form an augmented subspace

$$\mathcal{S} = \text{span}\{u, z\},$$

from which a better approximation, (θ^+, u^+), to the desired eigenpair of F can be extracted. If the new residual $r^+ = Fu^+ - \theta^+ u^+$ remains large, we can repeat the correction procedure and further expand $\mathcal{S}$ with an additional correction vector. To promote numerical stability, the new correction vector is orthogonalized against all previously generated basis vectors of $\mathcal{S}$. As the dimension of $\mathcal{S}$ becomes large, it may be necessary to truncate the correction sequence and retain only a few basis vectors of $\mathcal{S}$. The correction process then continues, and new correction vectors are appended to the truncated subspace $\mathcal{S}$. The details of this algorithm are described in [4, 20]. For completeness, we list the main steps of this subspace correction procedure in Figure 4.1.

Clearly, the most time-consuming part of this correction scheme is to solve the bordered linear system (4.2). One can easily show that if (4.2) is solved exactly, then

$$z = -u + \gamma(F - \theta I)^{-1}u,$$

```
ALGORITHM. IMPLICITLY SHIFTED RQ-ITERATION.

Input: $(F, V, H)$ with $FV = VH$, $V^H V = I$, and $H$ is upper
          Hessenberg.
Output: $(V, H)$ such that $FV = VH, V^T V = I$ and $H$ is upper triangular.

1. for $j = 1, 2, 3, \ldots$ until converged,
   1.1. Select a shift $\mu \leftarrow \mu_j$;
   1.2. Factor $H - \mu I = RQ$; ($R$ is triangular and $Q$ is orthogonal)
   1.3. $H \leftarrow QHQ^T$ ; $V \leftarrow VQ^T$;
2. end;
```

Fig. 4.2. *Implicitly shifted RQ-iteration.*

where $\gamma = \frac{u^T(F-\theta I)^{-1}r}{u^T(F-\theta I)^{-1}u}$. If we simply add the correction vector to u, it follows that

$$u \leftarrow \gamma(F - \theta I)^{-1}u,$$

which indicates the close relationship between the above correction procedure and an inverse iteration.

To reduce the cost associated with solving (4.2), Sleijpen and Van der Vorst [20] suggested solving the correction equation by a preconditioned iterative method. They expressed z as the solution to the projected equation

$$(I - uu^T)(F - \theta I)(I - uu^T)z = -(I - uu^T)r. \tag{4.3}$$

Since z is merely used as a basis vector of $\mathcal{S} = \operatorname{span}\{V\}$, it is not necessary to solve (4.3) to full accuracy. Thus JD can be viewed as an inner-outer iteration. The outer iteration provides eigenvalue and eigenvector approximations through the standard Rayleigh–Ritz procedure [18]. Each inner iteration provides a new basis vector of $\mathcal{S}$ by solving (4.3).

4.2. The ITRQ iteration. The TRQ iteration brings in the flavor of inverse iteration in a different way. A full RQ-iteration is similar to the familiar QR algorithm [5, 6]. The algorithm begins with a Hessenberg reduction

$$FV = VH, \tag{4.4}$$

where $V^T V = I$ and H is upper Hessenberg. This reduction is followed by a procedure described in Figure 4.2, which eventually drives H into an upper triangular form with eigenvalues exposed on the diagonal. Since our matrix F is symmetric, H is symmetric and tridiagonal. As convergence takes place, H becomes diagonal. If we let $V_+ = VQ^T$, $H_+ = QHQ^T$, $v_1^+ = V_+e_1$, and $v_1 = Ve_1$, it is easy to verify that in a single RQ iterate

$$(F - \mu I)v_1^+ = v_1\rho_{1,1},$$

where $\rho_{1,1} = e_1^T Re_1$. This implies that the first column V_+ is what one would have obtained by applying one step of inverse iteration to v_1 with the shift μ. This property is preserved in all subsequent RQ iterates. Thus, one would expect very rapid convergence of leading columns of V to an invariant subspace of A.

To avoid carrying out the full RQ-iteration involving $n \times n$ orthogonal similarity transformations on large-scale problems, it is desirable to truncate this update procedure after k steps to maintain and update only the leading portion of the factorizations occurring in this sequence.

The truncation scheme developed in [23] leads to a procedure that maintains and updates the following set of equations:

$$(F - \mu I)(V_k, v_+) = (V_k, v) \begin{pmatrix} T_k - \mu I_k & h \\ \beta_k e_k^T & \alpha \end{pmatrix}, \tag{4.5}$$

where T_k is symmetric and tridiagonal, $V_k^T V_k = I_k$, $V_k^T v = 0$, $V_k^T v_+ = 0$, and $\|v\| = \|v_+\| = 1$.

Clearly, the first k columns of the above equation satisfy a Lanczos reduction

$$FV_k = V_k T_k + \beta_k v e_k^T.$$

The vectors v_+, h, and the scalar variable α can be determined by solving the *TRQ equation*

$$\begin{pmatrix} F - \mu I & V_k \\ V_k^T & 0 \end{pmatrix} \begin{pmatrix} v_+ \\ h \end{pmatrix} = \begin{pmatrix} v\alpha \\ 0 \end{pmatrix}, \quad \|v_+\| = 1. \tag{4.6}$$

Once the solution of the above equation is obtained, the TRQ algorithm proceeds by performing an RQ decomposition

$$\begin{pmatrix} T_k - \mu I_k & h \\ \beta_k e_k^T & \alpha \end{pmatrix} = \begin{pmatrix} R_k & r \\ 0 & \rho \end{pmatrix} \begin{pmatrix} Q_k & q \\ \sigma e_k^T & \gamma \end{pmatrix}$$

and updating V_k, T_k, and v by applying appropriate orthogonal transformations. We refer readers to [23] for a detailed derivation of the algorithm.

Again, the major cost for carrying out a TRQ iteration is in solving the TRQ equation. To avoid enormous memory usage, we prefer to solve the TRQ equation by a preconditioned iterative solver. Just as it was for the correction equation that appeared in JD, it is easy to verify that v_+ also satisfies the following equation:

$$(I - V_k V_k^T)(F - \mu I)(I - V_k V_k^T) v_+ = v\alpha. \tag{4.7}$$

If the equation is not solved to full accuracy, the residual error will be mixed into all columns of V_k so that they will no longer form a basis of a Krylov subspace. However, it is shown in [27] that the error is damped by the orthogonal transformation applied to columns of V_k. After the RQ update is complete, the first column of the updated V_k, v_1^+ satisfies the equation

$$(F - \mu I) v_1^+ = \rho_{11} v_1 + z\delta,$$

where z is the residual error produced by the iterative solver used to solve (4.7), and δ is the product of sines generated from the Givens rotations used in the RQ update. Therefore the error associated with the above inexact inverse iteration is likely to be considerably smaller than $\|z\|$. Consequently, we expect v_1^+ to converge rapidly to a desired eigenvector of F. However, to continue the next cycle of ITRQ, we must restart a k-step Lanczos iteration from v_1^+ to generate a new set of T_k, V_k, v, β_k. We outline the basic steps of ITRQ in Figure 4.3. A convergence analysis for ITRQ is provided in [27].

```
ALGORITHM. ITRQ ITERATION.

Input: $(F, V_k, T_k, f_k)$ with $FV_k = V_k T_k + f_k e_k^T, V_k^T V_k = I$,
        $T_k$ is tridiagonal.
Output: $(V_k, T_k)$ such that $FV_k = V_k T_k, V_k^T V_k = I$ and $T_k$ is
        diagonal.

1. Put $\beta_k = \|f_k\|$ and put $v = f_k/\beta_k$;
2. for $j = 1, 2, 3, ...$ until convergence,
    2.1. Select a shift $\mu \leftarrow \mu_j$;
    2.2. Solve $(I - V_k V_k^T)(F - \mu I)(I - V_k V_k^T)w = v$ approximately;
    2.3. $w \leftarrow (I - V_k V_k^T)w$, $v_+ \leftarrow w/\|w\|$;
    2.4. $h \leftarrow V_k^T F v_+$,  $\alpha \leftarrow v^T(F - \mu I)v_+$ ;
    2.5. Factor $\begin{pmatrix} T_k - \mu I_k & h \\ \beta_k e_k^T & \alpha \end{pmatrix} = \begin{pmatrix} R_k & r \\ 0 & \rho \end{pmatrix} \begin{pmatrix} Q_k & q \\ \sigma e_k^T & \gamma \end{pmatrix}$;
    2.6. $v_1 \leftarrow V_k Q_k^T e_1 + v_+ q^T e_1$;
    2.7. $(T_k, V_k, v, \beta_k) \leftarrow$ Lanczos$(F, v_1, k)$;
3. end;
```

FIG. 4.3. *ITRQ iteration.*

5. Deflation and preconditioning.

The overall performance of TRQ and JD depends largely on the efficiency of the preconditioned iterative solver used to solve a sequence of projected linear systems of the form (4.7) (inner iteration). Although these equations do not need to be solved to full accuracy, some level of convergence is needed to ensure improvement of the approximate eigenpair in the outer iteration. In this section, we will discuss techniques for accelerating the convergence of the inner iteration.

5.1. Deflation.

Because the eigenvectors corresponding to the lowest six eigenvalues of F represent translational and rotational modes, the invariant subspace associated with these modes can be easily constructed. For example, a translational mode can be fixed by specifying 1 in the x direction and 0 in the other two directions. Deflating these modes from JD and ITRQ iteration helps improve the convergence of the eigenvalue calculation. In JD, deflation essentially amounts to solving a correction equation

$$(I - QQ^T)(F - \theta I)(I - QQ^T)z = -r,$$

where $Q = (U, u)$ consists of both the converged eigenvectors (including the translational and rotational modes) U and the approximation to a desired vibrational mode u. Deflation in ITRQ can be achieved by *locking* the converged eigenvectors in the leading columns of V_k. We refer readers to [4, 23] for implementation details.

5.2. Preconditioner for *F*.

Recall that each nonzero entry of the matrix F corresponds to either a bonded or a nonbonded force interaction between a pair of atoms. Because the bonded interaction is typically stronger than the nonbonded interaction, a natural candidate for a preconditioner is a matrix B that contains only the bonded entries of F. For PE particles, bonded interaction occurs among four adjacent atoms. Since each atom has three degrees of freedom (x, y, z), the full bonded interaction can be represented by a 12×12 dense matrix block. If atoms are numbered appropriately, these dense blocks will appear (with some overlap) on the diagonal of F. If we also include some of the nonbonded interactions between atoms that belong to different but nearby molecules, B can be made into a banded matrix. One of the preconditioners used in our experiments is a banded matrix formed by extracting nonzero entries between the 12th super- and subdiagonals of F.

A precise analysis of the effects of this preconditioner appears to be difficult. The following observation provides an intuitive explanation on why this preconditioner works well for PE particles.

It follows from our construction that

$$F = B + E.$$

The norm of E is relatively small compared to that of F and B. For the 3000-atom PE particle (*pe*3*k*) used in our experiments, $\|F\|_2 = 127.6$, $\|B\|_2 = 127.6$, and $\|E\|_2 = 1.3$. Thus, one may view E as a perturbation. Since $B^{-1}F = I + B^{-1}E$, the spectrum of the preconditioned system depends solely on the eigenvalue distribution of $B^{-1}E$. Because the removal of nonbonded interaction essentially eliminates the possibility of rigid body motion (i.e., translations and rotations), B is unlikely to contain eigenvalues that are extremely close to zero. Therefore, we expect eigenvalues of $B^{-1}E$ to be small. We observed from numerical experiments that the eigenvalues of $B^{-1}E$ lie within the interval $(-1.0, 1.5)$ for *pe*3*k*.

In Figure 5.1, we plotted the estimated spectrum of $B^{-1}F$ (computed by a simple Lanczos run) and compared it with that of F. Clearly, the preconditioner is quite effective in reducing condition number by two orders of magnitude.

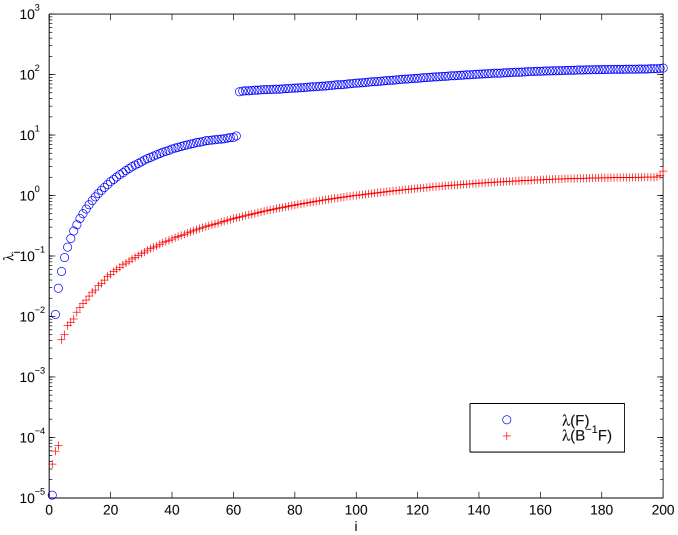


FIG. 5.1. *The estimated spectrum of F and $B^{-1}F$.*

To use the preconditioner in a symmetric iterative solver such as the conjugate gradient (CG) or the minimum residual (MINRES) method, B is required to be positive definite so that the Cholesky factors of B can be applied to preserve symmetry.

When this fails to be true, we add a small shift to the diagonal of B to make it positive definite.

Another technique we use to construct a preconditioner for F is *incomplete Cholesky factorization* [14, 15]. The factorization produces a lower triangular matrix L such that

$$E = F - LL^T$$

is small componentwise. If F is partitioned as

$$F = \begin{pmatrix} \alpha & \ell^T \\ \ell & \hat{F} \end{pmatrix},$$

the first step of the factorization leads to

$$\begin{pmatrix} \alpha & \ell^T \\ \ell & \hat{F} \end{pmatrix} \approx \begin{pmatrix} \sqrt{\alpha} & \\ \tilde{\ell}/\sqrt{\alpha} & I \end{pmatrix} \begin{pmatrix} \sqrt{\alpha} & \tilde{\ell}^T/\sqrt{\alpha} \\ & \hat{F} - \tilde{\ell}\tilde{\ell}^T/\alpha \end{pmatrix},$$

where $\tilde{\ell}$ is obtained by setting some small entries in ℓ to zero. Because most of the nonzero entries outside of the 12th subdiagonal element in ℓ correspond to weaker nonbonded interactions between atoms, their magnitude can be significantly smaller than $\sqrt{\alpha}$. Therefore it is reasonable to set some of these entries to zero to reduce the amount of fill-ins in $\hat{F} - \tilde{\ell}\tilde{\ell}^T/\alpha$. A heuristic threshold is often used to determine whether to keep or drop the nonzero entries in ℓ. When a fill-in is created in $\hat{F} - \tilde{\ell}\tilde{\ell}^T/\alpha$ its magnitude is likely to be small compared to the diagonal entry. Therefore, it is likely to be dropped from L as the factorization proceeds. As we will show in section 6, with a proper choice of drop tolerance, the number of nonzeros in L can be less than that in the lower triangular part of F.

5.3. Preconditioner for the projected system. We discussed how to construct preconditioners for F in section 5.2. However, in both JD and ITRQ, preconditioners must be applied to the projected system

$$(I - V_kV_k^T)(F - \theta I)(I - V_kV_k^T)x = \hat{b}, \tag{5.1}$$

where $V_k^T\hat{b} = 0$ and $V_k^Tx = 0$. In this section, we will provide a procedure for applying a pair of symmetric preconditioners to the projected system.

Suppose $B = LL^T$ is a good preconditioner for $F - \theta I$. It is reasonable to expect that

$$\hat{L} = (I - V_kV_k^T)L(I - V_kV_k^T),$$

together with $\hat{L}^T$, can be used to enhance the eigenvalue distribution of

$$C = (I - V_kV_k^T)(F - \theta I)(I - V_kV_k^T).$$

To apply these preconditioners in an iterative solver, we must find a convenient way to solve

$$\hat{L}\hat{x} = \hat{b} \quad \text{and} \quad \hat{L}^T\hat{y} = \hat{b},$$

where $V_k^T\hat{x} = 0$, $V_k^T\hat{y} = 0$, and $V_k^T\hat{b} = 0$. Because the solution to $\hat{L}\hat{x} = \hat{b}$ also satisfies

$$\begin{pmatrix} L & V_k \\ V_k^T & 0 \end{pmatrix} \begin{pmatrix} \hat{x} \\ -g \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}$$

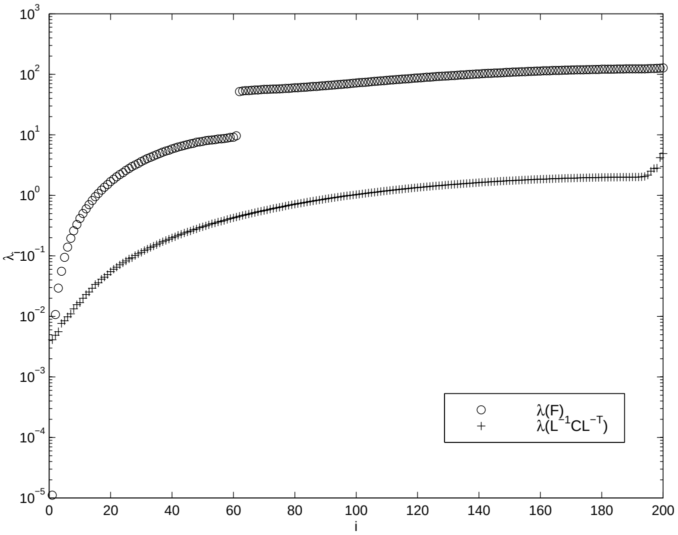


FIG. 5.2. *The estimated spectrum of F and $\hat{L}^{-1}(I - V_6V_6^T)F(I - V_6V_6^T)\hat{L}^{-T}$.*

for some $g \in \mathbb{R}^{k\times 1}$, block Gaussian elimination yields

$$\hat{x} = \hat{L}^{-1}\hat{b} = (I - Y_kG^{-1}V_k^T)L^{-1}\hat{b},$$

where $Y_k = L^{-1}V_k$ and $G = V_k^TY_k$. It follows that

$$\hat{L}^{-1}C\hat{L}^{-T} = \left(I - Y_kG^{-1}V_k^T\right)L^{-1}(F - \theta I)L^{-T}\left(I - V_kG^{-T}Y_k^T\right). \tag{5.2}$$

In Figure 5.2, we plotted the estimated spectrum of $\hat{L}^{-1}(I-V_6V_6^T)F(I-V_6V_6^T)\hat{L}^{-T}$ restricted to the subspace $V_6^{\perp}$, where V_6 consists of six vectors that describe the translational and rotational motions of *pe*3*k*. The estimated spectrum of the preconditioned and deflated matrix is compared to that of F. We observed that the combination of preconditioning and deflation effectively reduces the condition number of the linear system (in the inner iteration) from 10^8 to 10^2. Most eigenvalues of the preconditioned system have been clustered around 1.0.

Equation (5.2) defines a sequence of `matvec` operations that must be performed in one step of CG or MINRES iteration. One may follow the procedure provided below to modify a standard preconditioned iterative solver to compute an approximate solution of (5.1).

1. Solve $LY_k = V_k$.
2. Compute $G_k = V_k^TY_k$.
3. $\hat{b} \leftarrow L^{-1}\hat{b}$.
4. $\hat{b} \leftarrow \hat{b} - Y_kG_k^{-1}V_k^T\hat{b}$.
5. Apply an iterative solver to $(I-Y_kG^{-1}V_k^T)L^{-1}(F-\theta I)L^{-T}(I-V_kG^{-T}Y_k^T)y = \hat{b}$ to obtain an approximate solution y.
6. Compute $y \leftarrow y - V_kG^{-T}Y_k^Ty$.
7. Solve $L^T\hat{x} = y$.

6. Performance. In this section we present our computational experience applying JD and ITRQ to PE particles of various sizes. We found both algorithms are quite efficient at extracting a small number of low-frequency vibrational modes. The total number of floating point operations (flops) required to obtain the lowest 100 vibrational modes is kn^2 for some $k \ll n$. The JD algorithm appears to be more efficient than ITRQ when a set of optimal parameters is identified. These parameters include the following:

1. Preconditioner. We found that it is extremely expensive to solve the linear systems (4.2) and (4.7) without use of a preconditioner. We choose between two preconditioners of the linear systems. The first is a standard incomplete Cholesky preconditioner, which we call `IC`. The second is the banded submatrix of the original matrix obtained by ignoring the nonbonded interactions; we call this preconditioner `band`.
2. The convergence tolerance `tol` used to control the accuracy of the linear systems solved. The values of `tol` looked at here are $\sqrt{10^{-1}}$, 10^{-1}, 10^{-2}, 10^{-4}, and 10^{-6}. We also investigated the possibility of using a dynamic tolerance in which the `tol` value is gradually tightened in the outer iteration as the approximate eigenvector becomes more accurate.
3. The maximum dimension (`jmax`) of the approximating subspace constructed in the outer iteration and the minimum number (`jmin`) of vectors retained when restarting takes place.
4. Choice of shift θ. In the JD algorithm the linear systems are shifted using the most recent estimate of the eigenvalue. We can choose, alternatively, to refrain from shifting the linear system, and work directly with F. Shifting produces linear systems that are generally more ill-conditioned; however, it potentially obtains greater reduction in the eigenresidual per solve because shifting preserves the Newton-like solution process.

We ran our implementation of JD on three PE particle models *pe*3*k*, *pe*6*k*, and *pe*12*k* of orders 9000, 18000, and 36000, respectively. Problem statistics are given in Table 6.1, which includes statistics for the two preconditioners.

TABLE 6.1
Problem statistics.

Problem	n	nnz(F)	`nonzero` percentage(F)	`IC` nnz(L)	`band` nnz(L)
*pe*3*k*	9000	3,279,690	3.7%	903,510	106,161
*pe*6*k*	18000	6,897,316	1.8%	1,876,009	211,946
*pe*12*k*	36000	14,220,946	1.1%	3,540,453	423,517

Throughout we will use approximate counts of flops to measure performance. We count just the flops required by matrix-vector products and application of the preconditioners. We ignore the relatively small number of flops required by BLAS operations used in deflation, projection, and other MINRES operations. A flop refers to a multiply-add pair. An approximate eigenpair is declared converged when its residual norm drops below 10^{-6}.

6.1. `IC` and `band` preconditioners. To compute the `IC` preconditioner we used a drop-tolerance-based code provided to us by Edmond Chow [2], which provides a number of options that we found useful or even necessary. We used the option that scales the matrix before factorization so that it has unit main diagonal. We needed to shift the main diagonal to preserve positive definiteness. We tried a number of combinations of shifts and drop tolerances, and found that a drop tolerance of 10^{-4} and shift of 2×10^{-4} are very near optimal for all three problems under consideration.

Table 6.2

Average iterations per solve, flops per iterations, and flops per solve for MINRES using two different preconditioners during JD runs to calculate leading twenty eigenvalues. Linear systems are shifted; `tol` $= 10^{-1}$*; and* `jmin` $= 20$ *and* `jmax` $= 40$.

Problem	Iter/solve		Flops/iter		Flops/solve	
	`IC`	`band`	`IC`	`band`	`IC`	`band`
*pe*3*k*	10.5	23.6	1.02e7	6.84e6	1.08e8	1.61e8
*pe*6*k*	11.8	32.0	2.12e7	1.44e7	2.50e8	4.61e8
*pe*12*k*	12.4	34.7	4.21e7	2.96e7	5.21e8	1.03e9

Table 6.3

Approximate total flops during JD runs to calculate leading twenty eigenvalues using two different preconditioners. Linear systems are shifted; `tol` $= 10^{-1}$*; and* `jmin` $= 20$ *and* `jmax` $= 40$.

Problem	`IC`	`band`
*pe*3*k*	1.05e10	1.58e10
*pe*6*k*	2.38e10	4.27e10
*pe*12*k*	4.96e10	9.28e10

In Table 6.1 notice that each `IC` factor is considerably sparser than the lower triangle of the corresponding coefficient matrix. There are many very small entries in the coefficient matrices, and it is vital that the `IC` algorithm allows many of these small original entries to be dropped from the IC factors.

Observe in Table 6.1 that the `band` factors are much sparser than their `IC` counterparts, and hence much cheaper to apply. In Table 6.2 the number of flops per iteration is significantly smaller for `band`. But there are significant entries outside the band that need to play a role in the preconditioner.

`IC` permits entries to enter the preconditioner based on size alone and not position in the matrix (i.e., physical classification). While `IC` produces a much denser factor L, it is nonetheless much sparser than the lower triangle of the originating coefficient matrix. The number of iterations per solve in Table 6.2 shows that `IC` is a much more powerful preconditioner; the number of flops per solve in the same table show that it is much more efficient.

Finally, Table 6.3 shows approximate total flops for JD runs using `IC` and `band`. `IC` is clearly superior and has been in all the tests we have run. Henceforth we drop `band` and work with `IC` alone.

6.2. Accuracy in the linear equations. Using a “factorization-free” approach, we can choose how accurately to solve the linear systems. This turns out to be important. We tried various tolerances `tol` to control the relative reduction in the MINRES residuals. Table 6.4 shows approximate total work for the various values of `tol`.

Table 6.4

Approximate total flops during JD runs to calculate leading twenty eigenvalues using various settings of `tol`. `IC` *is used; linear systems are shifted; and* `jmin` $= 20$ *and* `jmax` $= 40$.

Problem	Tolerance					
	$\sqrt{10^{-1}}$	10^{-1}	10^{-2}	10^{-4}	10^{-6}	Dynamic
*pe*3*k*	1.22e10	**1.05e10**	1.45e10	1.81e10	2.14e10	**1.09e10**
*pd*6*k*	2.73e10	**2.38e10**	3.43e10	4.53e10	5.37e10	**2.36e10**
*pe*12*k*	5.50e10	**4.96e10**	7.28e10	1.01e11	1.16e11	**4.95e10**

Roughly one digit of accuracy (`tol` $= 10^{-1}$) is preferable for all three problems. The superiority of `tol` $= 10^{-1}$ manifested itself in all the tests we ran, which includes many runs not recorded in the table. One pays a fairly high price for solving the linear systems too accurately. In general, our experience has been that high accuracy in the linear system does not lead, with sufficient consistency, to commensurate reductions in the eigenresiduals in the JD algorithm. We have also observed that the cost of an average linear solve increases faster than accuracy as shift becomes closer to an eigenvalue.

Following the strategy proposed in [4], we also experimented with a dynamic tolerance $\texttt{tol}_j = \texttt{tol}_0/2^j$, where j is the outer iterate number, and $\texttt{tol}_0$ is the initial residual. This is motivated by the fact that JD can be viewed as an inexact Newton iteration in which a higher accuracy is required near the solution of the correction equation. However, for this scheme to be competitive with the fixed tolerance approach, we must limit the maximum number of MINRES iterations to a small number ($10 \sim 15$ in most cases). We observed that this limit is almost always reached before the residual error of the approximate solution falls below the dynamic tolerance. Henceforth the following results are obtained by using $\texttt{tol} = 10^{-1}$ in our runs.

6.3. Projection subspace size in JD. Our experience verified our expectation that the larger the space from which approximate eigenvectors are taken, the more efficiently the algorithm runs. Table 6.5 shows that increasing the maximum size jmax of the basis from 10 to 20 to 40 (along with the minimum size jmin from 5 to 10 to 20) reduces approximate total work when either shifted or unshifted linear systems are used.

Table 6.5
Approximate total flops during JD runs to calculate leading twenty eigenvalues using various settings of jmax *and* jmin *and shifted and unshifted linear systems.* IC *and* $\texttt{tol} = 10^{-1}$ *are used.*

	Shifted			Unshifted		
	(jmin,jmax)			(jmin,jmax)		
Problem	(5,10)	(10,20)	(20,40)	(5,10)	(10,20)	(20,40)
*pe*3*k*	1.26e10	1.13e10	1.05e10	1.88e10	1.47e10	1.26e10
*pe*6*k*	2.89e10	2.50e10	2.38e10	4.42e10	3.31e10	2.79e10
*pe*12*k*	6.05e10	5.32e10	4.96e10	9.10e10	7.14e10	5.90e10

It is interesting to note that the effect of increasing jmax is far more pronounced when the linear systems are unshifted rather than shifted. Unshifted is far more competitive with shifted when jmax = 40 rather than jmax = 10. In all our tests jmax = 40 outperformed the lower alternative values. The primary cost in increasing jmax is extra space for the basis vectors. The cost in storage of using jmax = 40 is not too large, and in the table it appears to be a reasonable point of diminishing returns. Hence, we choose jmax = 40.

6.4. Shifted versus unshifted linear systems. Table 6.5 also shows that shifting the linear systems is superior to the alternative of using unshifted linear systems. When jmax is large, the advantage of shifting is modest but significant; when jmax is small, the advantage of shifting can be quite large. Our experience with other runs not shown in the table also corroborate the conclusions we draw here.

6.5. A closer look at an optimal run. Let us now consider in more detail a single run of JD to compute the leading 20 eigenpairs for the smallest problem (*pe*3*k*) using the optimal parameters IC, $\texttt{tol} = 10^{-1}$, jmax = 40, jmin = 20, and shifted linear systems. Figure 6.1 displays the convergence history of the leading 20 eigenpairs. The residual norm of each approximate eigenpair is plotted against the flops mea-

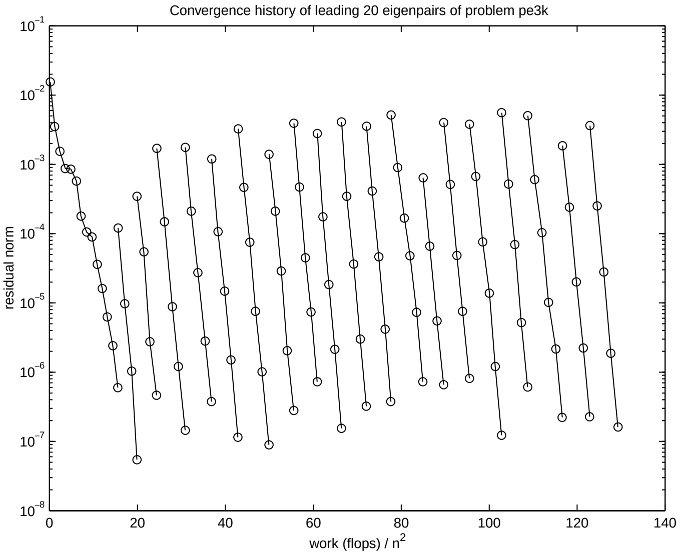


FIG. 6.1. *The residual history of the first* 20 *eigenpairs in a JD run.*

sured at each outer iteration of JD. The first eigenpair is computed without shifting because in our experience shifting is always completely ineffective for this eigenpair; moreover, shifting is always considerably more expensive for this eigenpair. The convergence histories for the other 19 eigenpairs are remarkably regular. The thing we would like to emphasize here is that with great regularity reducing the residual in the linear system by one order of magnitude results in reducing the eigenresidual by close to one order of magnitude. The average number of iterations per linear system solve is 10.5 iterations. The average number of MINRES iterations per eigenpair is 47.5 iterations.

It is important that some attention is paid to tuning the linear equation solver and JD for this application. Using `IC`, unshifted linear systems, `tol` $= 10^{-6}$, `jmax` $= 10$, and `jmin` $= 5$ it takes approximately 4.24×10^{10} flops to compute the leading 20 eigenpairs for problem *pe*3*k*. This is 4.0 times more flops than required by the optimal run. Using `band`, unshifted linear systems, `tol` $= 10^{-6}$, `jmax` $= 10$, and `jmin` $= 5$ it takes approximately 1.04e11 flops to compute the leading 20 eigenpairs for problem *pe*3*k*. This is 9.9 times more flops than required by the optimal run.

6.6. Comparison with ITRQ. Our preliminary results show that ITRQ is less efficient than JDQR for the PE problems. The convergence history of the first 20 eigenpairs is displayed in Figure 6.2. The residual norm of each approximate eigenpair is plotted against the flops measured at each RQ update step. We observe that the total number of flops required to capture the lowest 20 eigenpairs is significantly larger than that required by an optimal JD run. Part of the reason why ITRQ takes more flops is that our current implementation solves the full bordered system (4.2) by preconditioned MINRES. This may not be necessary because (4.2) can be rearranged into an equation that involves no V_k by exploiting the Lanczos relation $FV_k = V_k T_k + f e_k^T$ [23]. However, our experience seems to indicates that ITRQ requires a tighter convergence tolerance (`tol` $= 10^{-7}$) for the preconditioned iterative solver to obtain sufficient reduction in the residual norm in approximate eigenpairs.

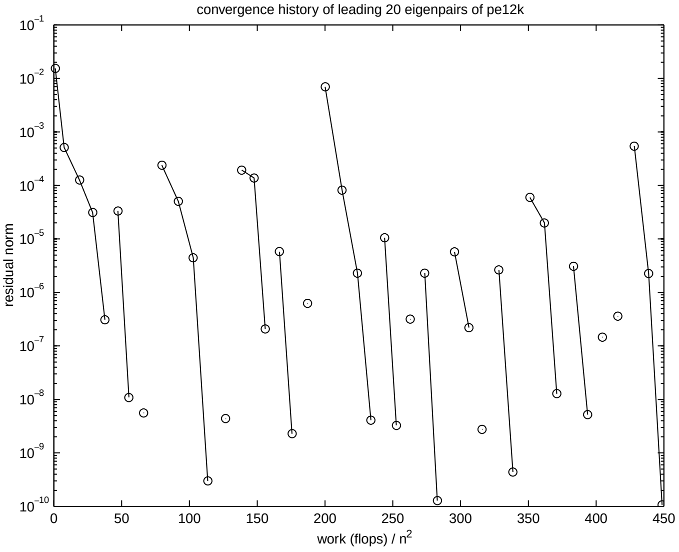


FIG. 6.2. *The residual history of the first* 20 *eigenpairs in an ITRQ run.*

Figure 6.2 also indicates that the convergence pattern of ITRQ is quite different from that of JD. Several approximate eigenpairs converge almost at the same time, whereas in JD an almost equal amount of time is spent on computing each eigenpair.

7. Conclusion. In this paper, we presented our computational experience in applying JD and ITRQ algorithms to obtain fundamental vibrational modes of large-scale molecular structures. The key to the efficiency of our computation is the choice of preconditioners used in the inner iteration of both JD and ITRQ. We found that both the banded preconditioner that contains mainly the bonded interaction and an incomplete Cholesky preconditioner are quite effective. We presented additional tuning techniques for improving the performance of JD. We believe both algorithms can be developed into general software tools for performing NCA on large-scale molecular structures.

Acknowledgments. We would like to thank Drs. Charles H. Romine and Eduardo D'Azevedo for carefully reading the manuscript and providing us with corrections and suggestions. We would also like to thank Prof. Ralph Byers for a helpful discussion on preconditioners during his visit to Oak Ridge National Laboratory. We are grateful for comments from the referees that helped improve the quality of this paper.

REFERENCES

[1] M. D. Barnes, C. Y. Kung, N. Lermer, K. Fukui, B. G. Sumpter, D. W. Noid, and J. U. Otaigbe, *Homogeneous polymer blend microparticles with a tunable refractive index*, Opt. Lett., 24 (1999), pp. 121–123.

[2] E. Chow, *private communication,* Lawrence Livermore National Laboratory, Livermore, CA, 2000.

[3] I. S. Duff and H. A. van der Vorst, *Developments and trends in parallel solutions of linear systems*, Parallel Comput., 25 (1999), pp. 1931–1970.

[4] D. R. Fokkema, G. L. G. Sleijpen, and H. A. Van der Vorst, *Jacobi–Davidson style QR and QZ algorithms for the reduction of matrix pencils*, SIAM J. Sci. Comput., 20 (1998), pp. 94–125.

[5] J. G. F. Francis, *The QR transformation a unitary analogue to the LR transformation—Part* 1, Comput. J., 4 (1961), pp. 265–271.

[6] J. G. F. Francis, *The QR transformation—Part* 2, Comput. J., 4 (1962), pp. 332–345.

[7] K. Fukui, B. G. Sumpter, D. Noid, C. Yang, and R. E. Tuzun, *Spectra analysis of macromolecular systems: Chain length effects in polymer particles*, J. Polymer Sci.: Part B, 38 (2000), pp. 1812–1823.

[8] C. Hayashi, R. Uyeda, and A. Tasaki, *Ultra-Fine Particles Technology*, Noyes, NJ, 1997.

[9] M. T. Heath, E. Ng, and B. W. Peyton, *Parallel algorithms for sparse linear systems*, SIAM Rev., 33 (1991), pp. 420–460.

[10] G. Karypis and V. Kumar, *METIS: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices*, Technical report, Department of Computer Science, University of Minnesota, Minneapolis, MN, 1998.

[11] C. Y. Kung, M. D. Barnes, N. Lermer, W. B. Whitten, and J. M. Ramsey, *Confinement and manipulation of individual molecules in attoliter volumes*, Anal. Chem., 70 (1998), pp. 658–661.

[12] C. Lanzcos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, J. Res. Nat. Bur. Standards, 45 (1950), pp. 255–282.

[13] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*, SIAM, Philadelphia, PA, 1998.

[14] T. A. Manteuffel, *An incomplete factorization technique for positive definite linear systems*, Math. Comp., 34 (1980), pp. 473–497.

[15] J. A. Meijerink and H. A. van der Vorst, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric m-matrix*, Math. Comp., 31 (1977), pp. 148–162.

[16] D. W. Noid, K. Fukui, B. G. Sumpter, C. Yang, and R. E. Tuzun, *Time-averaged normal coordinate analysis of polymer particles and crystals*, Chem. Phys. Lett, 316 (2000), pp. 285–296.

[17] P. C. Painter, M. M. Coleman, and J. L. Koeing, *The Theory of Vibrational Spectroscopy and Its Application to Polymer Materials*, John Wiley, New York, 1982.

[18] B. N. Parlett, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, NJ, 1980.

[19] T. Renger, *private communication,* California Institute of Technology, Pasadena, CA, 2000.

[20] G. L. G. Sleijpen and H. A. Van der Vorst, *A Jacobi–Davidson iteration method for linear eigenvalue problems*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 401–425.

[21] D. C. Sorensen, *Implicit application of polynomial filters in a k-step Arnoldi method*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 357–385.

[22] D. C. Sorensen and C. Yang, *Accelerating the Lanczos Algorithm via Polynomial Spectral Transformations*, Technical report TR 97-29, Department of Computational and Applied Mathematics, Rice University, Houston, TX, 1997.

[23] D. C. Sorensen and C. Yang, *A truncated RQ iteration for large scale eigenvalue calculations*, SIAM J. Matrix Anal. Appl., 19 (1998), pp. 1045–1073.

[24] R. E. Tuzun, D. W. Noid, and B. G. Sumpter, *Efficient computation of potential energy first and second derivatives for molecular dynamics, normal coordinate analysis, and molecular mechanics calculations*, Macromol. Theory Simul., 5 (1996), pp. 771–787.

[25] R. E. Tuzun, D. W. Noid, and B. G. Sumpter, *Efficient treatment of out-of-plane bend and improper torsion interactions in MM*2*, MM*3*, and MM*4 *molecular mechanics calculations*, J. Comput. Chem., 18 (1997), pp. 1804–1811.

[26] E. B. Wilson, J. C. Decius, and P. C. Cross, *Molecular Vibrations*, Dover, New York, 1955.

[27] C. Yang, *Convergence analysis of an inexact truncated RQ-iteration. Large scale eigenvalue problems*, Electron. Trans. Numer. Anal., 7 (1998), pp. 40–55.

[28] C. Yang, P. Raghavan, L. Arrowood, D. W. Noid, B. G. Sumpter, and R. E. Tuzun, *Large-Scale Normal Coordinate Analysis on Distributed Memory Parallel Systems*, manuscript.

SIAM J. Sci. Comput. © 2001 Society for Industrial and Applied Mathematics
Vol. 23, No. 2, pp. 583–590

GENERALIZING EIGENVALUE THEOREMS TO PSEUDOSPECTRA THEOREMS[*]

MARK EMBREE[†] AND LLOYD N. TREFETHEN[†]

Abstract. Analysis of nonsymmetric matrix iterations based on eigenvalues can be misleading. In this paper, we discuss sixteen theorems involving ε-pseudospectra that each generalize a familiar eigenvalue theorem and may provide more descriptive information in some cases. Our organizing principle is that each pseudospectral theorem reduces precisely to the corresponding eigenvalue theorem when $\varepsilon = 0$.

Key words. eigenvalues, pseudospectra, matrix iterations

AMS subject classifications. 15A18, 15A60, 65F10, 65F15

PII. S1064827500373012

1. Introduction.

Though we speak of linear algebra, matrix iterative methods belong to the realm of linear analysis. Convergence of errors or residuals to zero is the concern, and this process has meaning because the algebraic problem is embedded in a normed space. Except for questions concerning finite termination, the appropriate tools for analyzing convergence are not the tools of algebra, such as eigenvalues, which are basis-independent, but those of analysis, such as singular values, which are defined via norms and necessarily change with the basis.

In this paper we consider the particular tools of linear analysis known as pseudospectra, which were invented to give information about matrices that lack a well-conditioned basis of eigenvectors. For simplicity, our norm $\|\cdot\|$ will always be the vector 2-norm and the matrix 2-norm that it induces. With this choice of norm, the matrices of interest are those that are far from normal in the sense that their eigenvectors, if a complete set exists, are far from orthogonal. Many of our results can be extended to other norms and also to operators as well as matrices, but we will not discuss these generalizations.

Throughout the article, A is an $N \times N$ matrix, and $\Lambda(A)$ denotes its spectrum, i.e., its set of eigenvalues, a subset of the complex plane $\mathbb{C}$. The pseudospectra of A are nested subsets of $\mathbb{C}$ that expand to fill the plane as $\varepsilon \to \infty$.

Definition. *For each $\varepsilon \ge 0$, the ε-pseudospectrum $\Lambda_\varepsilon(A)$ of A is the set of numbers $z \in \mathbb{C}$ satisfying any of the following equivalent conditions:*

(i) $\|(z-A)^{-1}\| \ge \varepsilon^{-1}$;

(ii) $\sigma_{\min}(z-A) \le \varepsilon$;

(iii) $\|Au - zu\| \le \varepsilon$ *for some vector u with* $\|u\| = 1$;

(iv) *z is an eigenvalue of $A+E$ for some matrix E with* $\|E\| \le \varepsilon$.

Here $\sigma_{\min}$ denotes the smallest singular value, and we employ the convention that $\|(z-A)^{-1}\| = \infty$ for $z \in \Lambda(A)$.

Pseudospectra were introduced as early as 1975 [12] and became a popular tool during the 1990s. We will not give detailed references here, but we refer the reader to [23] and [24] for examples, to [25] for algorithms and a list of applications, and to [26]

[*] Received by the editors June 5, 2000; accepted for publication (in revised form) September 6, 2000; published electronically July 10, 2001. This work was supported by UK Engineering and Physical Sciences Research Council grant GR/M12414.

http://www.siam.org/journals/sisc/23-2/37301.html

[†] Oxford University Computing Laboratory, Wolfson Building, Parks Road, Oxford OX1 3QZ, UK (embree@comlab.ox.ac.uk, LNT@comlab.ox.ac.uk).

for history. For extensive online information about pseudospectra, including examples and a bibliography of papers by many authors, see the Pseudospectra Gateway [3].

This brief article is devoted to a simple idea:

Many theorems about eigenvalues are special cases $\varepsilon = 0$ of theorems about ε-pseudospectra.

Our whole content consists of the presentation of sixteen examples of theorems of this kind. These theorems are for the most part neither mathematically deep nor even new, though in some cases they have not been stated in the language of pseudospectra before. Nevertheless, for practical applications involving highly nonnormal matrices, they may sometimes be more useful than their eigenvalue special cases. This will tend to be so in situations where the eigenvalues of A are misleading, filling a region of $\mathbb{C}$ smaller than where A actually "lives." For an example illustrating the limitations of eigenvalue analysis for Krylov subspace methods for linear systems of algebraic equations, see [6]. Here is another extreme example. If A is nilpotent, with $A^K = 0$ for some $K \geq 1$, then $\Lambda(A) = \{0\}$. Some such matrices will have norms $\|A^k\|$ that diminish steadily toward 0 as $k \to K$, while for others, there may be no reduction until $k = K$ or great transient growth before the eventual decay. Eigenvalues alone cannot distinguish between these behaviors, but pseudospectra can.

Our presentation will adhere to a fixed pattern. In each case, we first list a theorem about eigenvalues, without proof, that is either elementary or well known. We follow this with a generalized theorem for pseudospectra together with an outline of a proof. Some pointers to the literature are included along the way, but we do not aim to be exhaustive, as it is often hard with this essentially elementary material to track down the first appearance of a result in print.

We hope that this article may provide a useful compendium for those concerned with nonnormal matrices and associated iterations, but we emphasize that this collection does not include all potentially useful theorems involving pseudospectra. By confining our attention to theorems that reduce for $\varepsilon = 0$ to valid statements about eigenvalues, we exclude some of the subtler estimates that may be obtained from pseudospectra, notably those based on contour integrals. One example is the Kreiss matrix theorem, which contains a constant eN that does not reduce cleanly to 1 as $\varepsilon \to 0$ [11, 19]. Another is the bound on a polynomial norm $\|p(A)\|$, of immediate relevance to iterations such as GMRES, that can be obtained by integrating $p(z)$ over the boundary contour(s) of $\Lambda_\varepsilon(A)$ [22]. For new results comparing such contour integral techniques to other approaches, see [5].

2. Sixteen theorems.

Our first theorem indicates the connection between the ill-conditioning of solving a linear system with A and the existence of a pseudoeigenvalue near the origin. This result has been attributed to Gastinel (see [21, pp. 120, 133], [28, p. 248]).

THEOREM 1. A is singular $\iff 0 \in \Lambda(A)$.

THEOREM 1ε. $\|A^{-1}\| \geq \varepsilon^{-1} \iff 0 \in \Lambda_\varepsilon(A)$.

Proof. The proof is immediate from the definitions. □

Pseudospectra possess the satisfying property that every connected component of the ε-pseudospectrum must contain at least one eigenvalue. This property forms the basis for the following result.

THEOREM 2. *A has N distinct eigenvalues $\Longrightarrow$ A is diagonalizable.*

THEOREM 2ε. *$\Lambda_\varepsilon(A)$ has N distinct components $\Longrightarrow$ A is diagonalizable.*

Proof. From definition (iv) of pseudospectra it is clear that for any $\delta > 0$, $\Lambda_\varepsilon(A)$ is contained in the interior of $\Lambda_{\varepsilon+\delta}(A)$. By the continuity of matrix eigenvalues with respect to perturbations, the same condition (iv) implies that if $\Lambda_\varepsilon(A)$ has N distinct components, so does $\Lambda_{\varepsilon+\delta}(A)$ for sufficiently small $\delta > 0$. Thus we see that $\|(z - A)^{-1}\|$ must achieve local maxima strictly in the interior of each of the N components of $\Lambda_{\varepsilon+\delta}(A)$. Now $\log \|(z - A)^{-1}\|$ is a subharmonic function of z throughout the complex plane except at the eigenvalues of A (see, e.g., [7, Thm. 3.13.1], [4]), and thus $\log \|(z-A)^{-1}\|$ and likewise $\|(z-A)^{-1}\|$ satisfy the maximum principle away from the eigenvalues of A. Putting these facts together, we see that each component of $\Lambda_\varepsilon(A)$ must contain an eigenvalue of A, which implies that A has N distinct eigenvalues and thus is diagonalizable. □

Gallestey has developed an algorithm for computing pseudospectra based on the maximum principle property used in the above proof [4]. A simpler exclusion algorithm, recently proposed by Koutis and Gallopoulos [10], is based upon the next result.

Theorem 3. $\|(z - A)^{-1}\| \ \ge\ \frac{1}{\operatorname{dist}(z,\Lambda(A))}$.

Theorem 3ε. $\|(z - A)^{-1}\| \ \ge\ \frac{1}{\operatorname{dist}(z,\Lambda_\varepsilon(A))+\varepsilon}$.

Proof. A perturbation of A of norm $\operatorname{dist}(z, \Lambda_\varepsilon(A)) + \varepsilon$ could make z an eigenvalue. □

The Koutis–Gallopoulos algorithm utilizes Theorem 3ε rewritten in the form

$$\operatorname{dist}(z, \Lambda_\varepsilon(A)) \ \ge\ \frac{1}{\|(z - A)^{-1}\|} - \varepsilon.$$

In our next theorem, S is an arbitrary nonsingular matrix and $\kappa(S)$ is its condition number, $\kappa(S) \equiv \|S\|\|S^{-1}\|$. Though the theorem is stated as an inclusion in one direction only, it applies in the other direction too, and in that sense Theorem 4 maintains our usual pattern of being the special case $\varepsilon = 0$ of Theorem 4ε. The result demonstrates that pseudospectra are invariant under unitary transformations, and also reflects the extent to which an ill-conditioned similarity transformation can alter pseudospectra. When B is diagonal, so that SBS^{-1} represents a diagonalization of A, Theorem 4ε is equivalent to the most familiar version of the Bauer–Fike theorem [1].

Theorem 4. $A = SBS^{-1} \implies \Lambda(A) = \Lambda(B)$.

Theorem 4ε. $A = SBS^{-1} \implies \Lambda_\varepsilon(A) \subseteq \Lambda_{\kappa(S)\varepsilon}(B)$.

Proof. Since $(z - A)^{-1} = S(z - B)^{-1}S^{-1}$, $\|(z - A)^{-1}\| \le \kappa(S)\|(z - B)^{-1}\|$. Therefore if $\|(z - A)^{-1}\| \ge \varepsilon^{-1}$, then $\|(z - B)^{-1}\| \ge (\kappa(S)\varepsilon)^{-1}$. □

The following theorem makes use of the idea of the "average pseudoeigenvalue" of a matrix, $\operatorname{mean}_{\lambda_\varepsilon\in\Lambda_\varepsilon(A)}\lambda_\varepsilon$. Of course, this quantity needs to be defined. We could be very specific and make use of, say, Haar measure (isotropy in $\mathbb{C}^N$) on the space of $N \times N$ matrices, but for the purposes of this theorem it is enough to say that $\operatorname{mean}_{\lambda_\varepsilon\in\Lambda_\varepsilon(A)}\lambda_\varepsilon$ is the mean of the eigenvalues of $A + E$ averaged over any fixed distribution on the matrices E with $\|E\| \le \varepsilon$ with the property that each matrix entry e_{ij} has mean 0.

Theorem 5. $\operatorname{tr}(A) \ =\ N \cdot \operatorname{mean}_{\lambda\in\Lambda(A)}\lambda$.

Theorem 5ε. $\operatorname{tr}(A) \ =\ N \cdot \operatorname{mean}_{\lambda_\varepsilon\in\Lambda_\varepsilon(A)}\lambda_\varepsilon$.

Proof. The theorem looks deep but is elementary. All we need to do is consider traces of perturbed matrices. Since each e_{jj} has mean 0 by assumption, so does their sum, and thus $\operatorname{tr}(A) = \operatorname{mean}_{\|E\|\le\varepsilon}\operatorname{tr}(A + E) = N \cdot \operatorname{mean}_{\lambda_\varepsilon\in\Lambda_\varepsilon(A)}\lambda_\varepsilon$. □

Our next pair of results requires a definition of the *condition number* $\kappa_A(\Sigma(A))$ of a compact set $\Sigma = \Sigma(A) \subset \mathbb{C}$ depending on A with respect to perturbations of A. If Σ_1 and Σ_2 are compact subsets of $\mathbb{C}$, let $d(\Sigma_1, \Sigma_2)$ denote the Hausdorff distance $d(\Sigma_1, \Sigma_2) = \max\{\max_{s\in\Sigma_1} d(s, \Sigma_2), \max_{s\in\Sigma_2} d(s, \Sigma_1)\}$, where $d(s, \Sigma)$ is the usual distance of a point s to a set Σ. Then

$$\kappa_A(\Sigma(A)) \equiv \limsup_{\delta\to 0}\left(\delta^{-1}\sup_{\|D\|\le\delta} d(\Sigma(A+D), \Sigma(A))\right).$$

THEOREM 6. $\Lambda(A)$ *depends continuously on* A*, with condition number* $\kappa_A(\Lambda(A)) = 1$ *if* A *is normal.*

THEOREM 6ε. $\Lambda_\varepsilon(A)$ *depends continuously on* A*, with condition number* $\kappa_A(\Lambda_\varepsilon(A)) = 1$ *if* A *is normal.*

Proof. The continuity of $\Lambda_\varepsilon(A)$ in the Hausdorff metric follows from the analogous continuity of $\Lambda(A)$. Suppose now that A is normal, so that its ε-pseudospectrum is the union of ε-disks centered at each eigenvalue. For any $\delta \ge 0$ and $D \in \mathbb{C}^{N\times N}$ with $\|D\| \le \delta$, we have $\max_{s\in\Lambda_\varepsilon(A+D)} d(s, \Lambda_\varepsilon(A)) \le \delta$ and similarly $\max_{s\in\Lambda_\varepsilon(A)} d(s, \Lambda_\varepsilon(A+D)) \le \delta$. Thus,

$$\sup_{\|D\|\le\delta} d(\Lambda_\varepsilon(A+D), \Lambda_\varepsilon(A)) \le \delta.$$

Since $\cup_{\|D\|\le\delta}\Lambda_\varepsilon(A+D) = \Lambda_{\varepsilon+\delta}(A)$, there always exists some D with $\|D\| \le \delta$ such that $\max_{s\in\Lambda_\varepsilon(A+D)} d(s, \Lambda_\varepsilon(A)) = \delta$, and for such a D we must have $\|D\| = \delta$, since the pseudospectra are strictly nested sets. It follows that

$$\kappa_A(\Lambda_\varepsilon(A)) \equiv \limsup_{\delta\to 0}\left(\delta^{-1}\sup_{\|D\|\le\delta} d(\Lambda_\varepsilon(A+D), \Lambda_\varepsilon(A))\right) = 1. \qquad \square$$

Eigenvalues can change dramatically with small perturbations, a warning that analysis based on them can be misleading. The following theorem hints that pseudospectra may be more robust.

THEOREM 7. $\Lambda(A+E) \subseteq \Lambda_{\|E\|}(A)$.

THEOREM 7ε. $\Lambda_\varepsilon(A+E) \subseteq \Lambda_{\varepsilon+\|E\|}(A)$.

Proof. If $z \in \Lambda_\varepsilon(A+E)$, then there exists a matrix F with $\|F\| \le \varepsilon$ such that $(A+E+F)u = zu$ for some $u \ne 0$. Since $\|E+F\| \le \varepsilon + \|E\|$, $z \in \Lambda_{\varepsilon+\|E\|}(A)$. □

We now turn to the problems of estimating the behavior of a matrix from its spectrum and pseudospectra.

THEOREM 8. $\lambda \in \Lambda(A) \implies \|A\| \ge |\lambda|$.

THEOREM 8ε. $\lambda_\varepsilon \in \Lambda_\varepsilon(A) \implies \|A\| \ge |\lambda_\varepsilon| - \varepsilon$.

Proof. If $\lambda_\varepsilon \in \Lambda_\varepsilon(A)$, then $Au = \lambda_\varepsilon u + \varepsilon v$ for some vectors $u, v \in \mathbb{C}$ with $\|u\| = \|v\| = 1$. It follows that $\|Au\| \ge |\lambda_\varepsilon| - \varepsilon$. □

The convergence analysis of stationary iterative methods is based on the behavior of powers of the iteration matrix. It has long been known that transient growth can occur even when the spectral radius of the iteration matrix is less than one (see, e.g., [27, p. 63]). The following two theorems use pseudospectra to describe this transient growth. The first is the "easy half of the Kreiss matrix theorem," that is, the half of that theorem that does not depend on N and whose proof is elementary [11].

THEOREM 9. $\max_{\lambda\in\Lambda(A)} |\lambda| > 1 \implies \sup_{k>0} \|A^k\| = \infty$.

THEOREM 9ε. $\max_{\lambda_\varepsilon\in\Lambda_\varepsilon(A)} |\lambda_\varepsilon| > 1 + C\varepsilon \implies \sup_{k\ge 0} \|A^k\| > C$.

Proof. Since $\|A^0\| = 1$, the result is trivial for $C < 1$, so assume $C \ge 1$. If $\max_{\lambda\in\Lambda(A)} |\lambda| > 1$, then the conclusion certainly holds, so assume $\max_{\lambda\in\Lambda(A)} |\lambda| \le 1$, in which case we have the convergent series representation

$$(z-A)^{-1} \;=\; z^{-1}(I + z^{-1}A + z^{-2}A^2 + \cdots),$$

which is valid for all z with $|z| > 1$. We now argue the contrapositive. If $\|A^k\| \le C$ for all $k \ge 0$, then

$$\|(z-A)^{-1}\| \;\le\; \frac{|z^{-1}|C}{1-|z^{-1}|} \;=\; \frac{C}{|z|-1}$$

for any z with $|z| > 1$. This implies that $\Lambda_\varepsilon(A)$ is contained in the disk about the origin of radius $1 + C\varepsilon$, i.e., $\max_{\lambda_\varepsilon \in \Lambda_\varepsilon(A)} |\lambda_\varepsilon| \le 1 + C\varepsilon$. □

THEOREM 10. $\lambda \in \Lambda(A) \implies \|A^k\| \ge |\lambda|^k$ *for all* k.

THEOREM 10ε. $\lambda_\varepsilon \in \Lambda_\varepsilon(A) \implies \|A^k\| \ge |\lambda_\varepsilon|^k - \frac{k\varepsilon\|A\|^{k-1}}{1-k\varepsilon/\|A\|}$ *for all* k *such that* $k\varepsilon < \|A\|$.

Proof. Pick E such that $\|E\| \le \varepsilon$ and $\lambda_\varepsilon \in \Lambda(A+E)$. Then $\|(A+E)^k\| \ge |\lambda_\varepsilon|^k$, which implies

$$\begin{aligned}\|A^k\| &\ge |\lambda_\varepsilon|^k - k\varepsilon\|A\|^{k-1} - \binom{k}{2}\varepsilon^2\|A\|^{k-2} - \cdots \\ &\ge |\lambda_\varepsilon|^k - k\varepsilon\|A\|^{k-1}\left(1 + k\varepsilon/\|A\| + (k\varepsilon)^2/\|A\|^2 + \cdots\right).\end{aligned}$$

Provided $k\varepsilon < \|A\|$, the series in this last equation converges, giving

$$\|A^k\| \ge |\lambda_\varepsilon|^k - \frac{k\varepsilon\|A\|^{k-1}}{1-k\varepsilon/\|A\|}. \qquad \square$$

Theorems 9 and 9ε have exact analogues for continuous time (see [14, 15]).

THEOREM 11. $\max_{\lambda\in\Lambda(A)} \mathrm{Re}\lambda > 0 \implies \sup_{t>0} \|e^{tA}\| = \infty$.

THEOREM 11ε. $\max_{\lambda_\varepsilon\in\Lambda_\varepsilon(A)} \mathrm{Re}\lambda_\varepsilon > C\varepsilon \implies \sup_{t>0} \|e^{tA}\| > C$.

Proof. As in the proof of Theorem 9ε, the conclusion is immediate if $C < 1$ or if $\max_{\lambda\in\Lambda(A)} \mathrm{Re}\lambda > 0$, so we assume that $C \ge 1$ and $\max_{\lambda\in\Lambda(A)} \mathrm{Re}\lambda \le 0$ and use the Laplace transform identity

$$(z-A)^{-1} \;=\; \int_0^\infty e^{-zt}e^{tA}dt,$$

which is valid for $\mathrm{Re}\, z > 0$. Again arguing the contrapositive, we note that if $\|e^{tA}\| \le C$ for all $t > 0$, then $\|(z-A)^{-1}\| \;\le\; C/\mathrm{Re}\, z$ for z with $\mathrm{Re}\, z > 0$, implying that $\Lambda_\varepsilon(A)$ is contained in the half-plane defined by $\mathrm{Re}\, z \le C\varepsilon$. □

Our next result is a pseudospectral generalization of Gerschgorin's theorem, which we believe to be new. It implies that if $\Lambda_\varepsilon(A)$ contains points distant from $\Lambda(A)$ for sufficiently small ε, then the bounds given by Gerschgorin's theorem will be more sharply descriptive of the pseudospectra than of the spectrum. Coupling this with Theorems 9ε and 10ε, one sees that Gerschgorin eigenvalue estimates may sometimes lead to more accurate predictions of transient behavior of iterative matrix processes than would be obtained from the exact eigenvalues! As has been pointed out in [13], this curious robustness phenomenon is of practical importance, for it sheds light on how it is that iterations such as GMRES may sometimes converge handily even when the associated Ritz values or harmonic Ritz values are far from accurate eigenvalue estimates. For these theorems, define $d_j = a_{jj}$ and $r_j = \sum_{k\ne j} |a_{jk}|$, and for any complex number z and real number $r \ge 0$, let $D(z, r)$ denote the closed disk about z of radius r.

THEOREM 12. $\Lambda(A) \;\subseteq\; \bigcup_j D(d_j, r_j)$.

THEOREM 12ε. $\Lambda_\varepsilon(A) \;\subseteq\; \bigcup_j D(d_j, r_j + \sqrt{N}\varepsilon)$.

Proof. Applying Gerschgorin's theorem to $A+E$ with $\|E\| \le \varepsilon$ yields inclusion disks centered at $d_j + e_{jj}$ with radius $\sum_{k\neq j} |a_{jk} + e_{jk}| \le r_j + \sum_{k\neq j} |e_{jk}|$. Each such disk is contained in the disk centered at d_j with radius $r_j + \sum_{k=1}^{N} |e_{jk}| = r_j + \|E_j\|_\infty$, where E_j denotes the matrix equal to E in the jth row and zero elsewhere. The term $\sqrt{N}\varepsilon$ comes from the inequality $\|E_j\|_\infty \le \sqrt{N}\|E_j\|_2 \le \sqrt{N}\|E\|_2$. □

The next result concerns the numerical range or field of values, which we denote by $W(A)$. In the context of iterative methods, the theorem indicates how analysis based on the field of values (see, e.g., [2]) relates to pseudospectral analysis. We write conv(S) for the convex hull in $\mathbb{C}$ of a set $S \subseteq \mathbb{C}$. The notation "$S \setminus \varepsilon$-border" also requires some explanation. By this we mean the set of points $z \in \mathbb{C}$ such that $D(z,\varepsilon) \subseteq S$. Perhaps Reddy, Schmid, and Henningson were the first to formulate this result in the language of pseudospectra [15, Thm. 2.2].

Theorem 13. *$W(A) \supseteq \operatorname{conv}(\Lambda(A))$.*

Theorem 13ε. *$W(A) \supseteq \operatorname{conv}(\Lambda_\varepsilon(A)) \setminus \varepsilon$-border.*

Proof. This result follows from a familiar result in functional analysis: that $W(A)$ is the intersection of all convex sets S that satisfy the condition

$$\|(z-A)^{-1}\| \le \frac{1}{\operatorname{dist}(z,S)}.$$

See, for example, Kato [9, p. 268]. □

The spectral mapping theorem (see, e.g., [9, p. 45]) is a jewel in the crown of eigenvalue theorems; it is theoretically appealing and practically relevant, forming the basis for rational transformation techniques for computing eigenvalues. The numerical range obeys a similar, though one-sided, mapping theorem [8]. Theorems 13 and 13ε suggest that a similar result might hold for pseudospectra. Our next theorem is a modest step in this direction, a precise mapping theorem for linear transformations [26, Thm. 2.4].

Theorem 14. *$\Lambda(\alpha + \beta A) = \alpha + \beta\Lambda(A)$ for $\alpha, \beta \in \mathbb{C}$.*

Theorem 14ε. *$\Lambda_{\varepsilon|\beta|}(\alpha + \beta A) = \alpha + \beta\Lambda_\varepsilon(A)$ for $\alpha, \beta \in \mathbb{C}$.*

Proof. The result is trivial when $\beta = 0$. Otherwise, note that

$$|\beta|\,\|(z-(\alpha+\beta A))^{-1}\| = \|(\beta^{-1}(z-\alpha) - A)^{-1}\|. \qquad □$$

For Theorems 15 and 16, let V denote an $N \times k$ rectangular matrix with orthonormal columns for some $k \le N$, as might be obtained by Arnoldi or subspace iteration, and let H denote a $k \times k$ square matrix. In the Arnoldi iteration, H would have Hessenberg form, but this is not necessary for these theorems. First, we assume that the columns of V exactly span an invariant subspace of A. The resulting theorem forms the basis for algorithms that compute pseudospectra by projecting A onto a carefully chosen invariant subspace [15, 25, 29].

Theorem 15. *$AV = VH \implies \Lambda(H) \subseteq \Lambda(A)$.*

Theorem 15ε. *$AV = VH \implies \Lambda_\varepsilon(H) \subseteq \Lambda_\varepsilon(A)$.*

Proof. If $\|Hu - zu\| \le \varepsilon$ for some $u \in \mathbb{C}^N$ with $\|u\| = 1$, then $\|VHu - Vzu\| \le \varepsilon$ too, and this implies $\|AVu - zVu\| \le \varepsilon$. □

Practical algorithms such as the implicitly restarted Arnoldi method [18] or subspace iteration (see, e.g., [16, section V.1]) may not easily yield an exact basis for the invariant subspace. Rather, the columns of V form an orthonormal basis for some *approximate* invariant subspace of A. Let H denote the generalized Rayleigh quotient this basis forms, $H \equiv V^*AV$. With this notation, eigenvalue Theorem 15 has an alternative, more practical pseudospectral generalization. This theorem is a fundamental result in the perturbation theory of invariant subspaces; see [20] and references therein.

THEOREM 16. $AV = VH \implies \Lambda(H) \subseteq \Lambda(A)$.

THEOREM 16ε. $AV = VH + R \implies \Lambda(H) \subseteq \Lambda_\varepsilon(A)$ *for* $\varepsilon = \|R\|$.

Proof. Consider the square matrix $E = -RV^*$. Then $(A+E)V = AV - R = VH$, so by Theorem 15, the eigenvalues of H are eigenvalues of $A+E$ and hence ε-pseudoeigenvalues of A for $\varepsilon = \|-RV^*\| = \|R\|$. □

For an Arnoldi factorization with k basis vectors, $V \in \mathbb{C}^{n\times k}$, Theorem 16$\varepsilon$ reduces to a well-known result: $\varepsilon = \|R\| = |h_{k+1,k}|$, where $h_{k+1,k}$ is the $(k+1, k)$ entry in the extended upper Hessenberg matrix (see, e.g., [17, Lem. 2.1]).

Acknowledgments. We thank Anne Greenbaum for carefully reading an earlier version of this manuscript and pointing out several corrections. We also appreciate helpful comments from Albrecht Böttcher, Nick Higham, and the referees. We especially thank Yvan Notay for noting that the constant in Theorem 12ε could be improved from N to $\sqrt{N}$.

REFERENCES

[1] F. L. BAUER AND C. T. FIKE, *Norms and exclusion theorems*, Numer. Math., 2 (1960), pp. 137–141.

[2] M. EIERMANN, *Fields of values and iterative methods*, Linear Algebra Appl., 180 (1993), pp. 167–197.

[3] M. EMBREE AND L. N. TREFETHEN, *Pseudospectra Gateway*, http://www.comlab.ox.ac.uk/pseudospectra, 2000.

[4] E. GALLESTEY, *Computing spectral value sets using the subharmonicity of the norm of rational matrices*, BIT, 38 (1998), pp. 22–33.

[5] A. GREENBAUM, *Using the Cauchy integral formula and the partial fractions decomposition of the resolvent to estimate* $\|f(A)\|$, SIAM J. Matrix Anal., submitted, 2000.

[6] A. GREENBAUM, V. PTÁK, AND Z. STRAKOŠ, *Any nonincreasing convergence curve is possible for GMRES*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 465–469.

[7] E. HILLE AND R. S. PHILLIPS, *Functional Analysis and Semi-Groups*, revised ed., AMS, Providence, RI, 1957.

[8] T. KATO, *Some mapping theorems for the numerical range*, Proc. Japan Acad., 41 (1965), pp. 652–655.

[9] T. KATO, *Perturbation Theory for Linear Operators*, 2nd ed., Springer-Verlag, New York, 1980.

[10] I. KOUTIS AND E. GALLOPOULOS, *Exclusion Regions and Fast Estimation of Pseudospectra*, Tech. report, Department of Computer Engineering and Informatics, HPCLAB, University of Patras, Patras, Greece, 2000.

[11] H.-O. KREISS, *Über die Stabilitätsdefinition für Differenzengleichungen die partielle Differentialgleichungen approximieren*, BIT, 2 (1962), pp. 153–181.

[12] H. J. LANDAU, *On Szegő's eigenvalue distribution theory and non-Hermitian kernels*, J. Analyse Math., 28 (1975), pp. 335–357.

[13] N. M. NACHTIGAL, L. REICHEL, AND L. N. TREFETHEN, *A hybrid GMRES algorithm for nonsymmetric linear systems*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 796–825.

[14] A. PAZY, *Semigroups of Linear Operators and Applications to Partial Differential Equations*, Springer-Verlag, New York, 1983.

[15] S. C. REDDY, P. J. SCHMID, AND D. S. HENNINGSON, *Pseudospectra of the Orr–Sommerfeld operator*, SIAM J. Appl. Math., 53 (1993), pp. 15–47.

[16] Y. SAAD, *Numerical Methods for Large Eigenvalue Problems*, Manchester University Press, Manchester, UK, 1992.

[17] V. SIMONCINI AND E. GALLOPOULOS, *Transfer functions and resolvent norm approximation of large matrices*, Electron. Trans. Numer. Anal., 7 (1998), pp. 190–201.

[18] D. C. SORENSEN, *Implicit application of polynomial filters in a k-step Arnoldi method*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 357–385.

[19] M. N. SPIJKER, *On a conjecture of LeVeque and Trefethen related to the Kreiss matrix theorem*, BIT, 31 (1991), pp. 551–555.

[20] G. W. STEWART, *Error and perturbation bounds for subspaces associated with certain eigenvalue problems*, SIAM Rev., 15 (1973), pp. 727–764.

[21] G. W. STEWART AND J. SUN, *Matrix Perturbation Theory*, Academic Press, Boston, 1990.

[22] L. N. TREFETHEN, *Approximation theory and numerical linear algebra*, in Algorithms for Approximation II, J. C. Mason and M. G. Cox, eds., Chapman and Hall, London, 1990, pp. 336–360.

[23] L. N. TREFETHEN, *Pseudospectra of matrices*, in Numerical Analysis 1991, D. F. Griffiths and G. A. Watson, eds., Longman Scientific and Technical, Harlow, UK, 1992, pp. 234–266.

[24] L. N. TREFETHEN, *Pseudospectra of linear operators*, SIAM Rev., 39 (1997), pp. 383–406.

[25] L. N. TREFETHEN, *Computation of pseudospectra*, in Acta Numer. 8, Cambridge University Press, Cambridge, UK, 1999, pp. 247–295.

[26] L. N. TREFETHEN, *Spectra and pseudospectra: The behavior of non-normal matrices and operators*, in The Graduate Student's Guide to Numerical Analysis, M. Ainsworth, J. Levesley, and M. Marletta, eds., Springer-Verlag, Berlin, 1999, pp. 217–250.

[27] R. S. VARGA, *Matrix Iterative Analysis*, Prentice–Hall, Englewood Cliffs, NJ, 1962. Second edition published by Springer-Verlag, Berlin, 2000.

[28] J. H. WILKINSON, *Sensitivity of eigenvalues* II, Util. Math., 30 (1986), pp. 243–286.

[29] T. G. WRIGHT AND L. N. TREFETHEN, *Large-Scale Computation of Pseudospectra Using ARPACK and Eigs*, Tech. report 00/11, Oxford University Computing Laboratory Numerical Analysis Group, Oxford, UK, 2000, SIAM J. Sci. Comput., 23 (2001), pp. 591–605.

SIAM J. Sci. Comput.
Vol. 23, No. 2, pp. 591–605

LARGE-SCALE COMPUTATION OF PSEUDOSPECTRA USING ARPACK AND EIGS[*]

THOMAS G. WRIGHT[†] AND LLOYD N. TREFETHEN[†]

Abstract. ARPACK and its Matlab counterpart, `eigs`, are software packages that calculate some eigenvalues of a large nonsymmetric matrix by Arnoldi iteration with implicit restarts. We show that at a small additional cost, which diminishes relatively as the matrix dimension increases, good estimates of pseudospectra in addition to eigenvalues can be obtained as a by-product. Thus in large-scale eigenvalue calculations it is feasible to obtain routinely not just eigenvalue approximations, but also information as to whether or not the eigenvalues are likely to be physically significant. Examples are presented for matrices with dimension up to 200,000.

Key words. Arnoldi, ARPACK, eigenvalues, implicit restarting, pseudospectra

AMS subject classifications. 65F15, 65F30, 65F50

PII. S106482750037322X

1. Introduction.

The matrices in many eigenvalue problems are too large to allow direct computation of their full spectra, and two of the iterative tools available for computing a part of the spectrum are ARPACK [10, 11] and its Matlab counterpart, `eigs`.[1] For nonsymmetric matrices, the mathematical basis of these packages is the Arnoldi iteration with implicit restarting [11, 23], which works by compressing the matrix to an "interesting" Hessenberg matrix, one which contains information about the eigenvalues and eigenvectors of interest. For general information on large-scale nonsymmetric matrix eigenvalue iterations, see [2, 21, 29, 31].

For some matrices, nonnormality (nonorthogonality of the eigenvectors) may be physically important [30]. In extreme cases, nonnormality combined with the practical limits of machine precision can lead to difficulties in accurately finding the eigenvalues. Perhaps the more common and more important situation is when the nonnormality is pronounced enough to limit the physical significance of eigenvalues for applications, without rendering them uncomputable. In applications, users need to know if they are in such a situation. The prevailing practice in large-scale eigenvalue calculations is that users get no information of this kind.

There is a familiar tool available for learning more about the cases in which nonnormality may be important: pseudospectra. Figure 1 shows some of the pseudospectra of the "Grcar matrix" of dimension 400 [6], the exact spectrum, and converged eigenvalue estimates (Ritz values) returned by a run of ARPACK (seeking the eigenvalues of largest modulus) for this matrix. In the original article [23] that described the algorithmic basis of ARPACK, Sorensen presented some similar plots of difficulties encountered with the Grcar matrix. This is an extreme example where the nonnormality is so pronounced that even with the convergence tolerance set to its lowest possible value, machine epsilon, the eigenvalue estimates are far from the true spectrum. From the Ritz values alone, one might not realize that anything was

[*] Received by the editors June 6, 2000; accepted for publication (in revised form) January 15, 2001; published electronically July 10, 2001.
http://www.siam.org/journals/sisc/23-2/37322.html

[†] Oxford University Computing Laboratory, Parks Road, Oxford OX1 3QD, UK (TGW@comlab.ox.ac.uk, LNT@comlab.ox.ac.uk).

[1] In Matlab version 5, `eigs` was an M-file adapted from the Fortran ARPACK codes. Starting with Matlab version 6, the `eigs` command calls the Fortran ARPACK routines themselves.

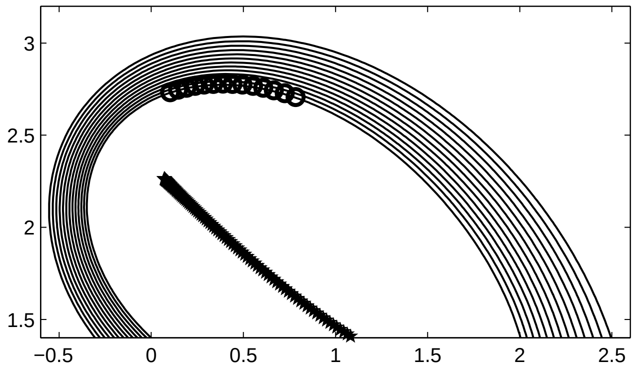


FIG. 1. *The ϵ-pseudospectra for $\epsilon = 10^{-5}, 10^{-6}, \ldots, 10^{-17}$ of the Grcar matrix (dimension 400), with the actual eigenvalues shown as solid stars and the converged eigenvalue estimates (for the eigenvalues of largest modulus) returned by ARPACK shown as open circles. The ARPACK estimates lie between the 10^{-16} and 10^{-17} pseudospectral contours.*

amiss. Once the pseudospectra are plotted too, it is obvious.

Computing the pseudospectra of a matrix of dimension N is traditionally an expensive task, requiring an $\mathcal{O}(N^3)$ singular value decomposition at each point in a grid. For a reasonably fine mesh, this leads to an $\mathcal{O}(N^3)$ algorithm with the constant of the order of thousands. Recent developments in algorithms for computing pseudospectra have improved the constant [28], and the asymptotic complexity for large sparse matrices [3, 12], but these are still fairly costly techniques. In this paper we show that for large matrices, we can cheaply compute an approximation to the pseudospectra in a region near the interesting eigenvalues. Our method uses the upper Hessenberg matrix constructed after successive iterations of the implicitly restarted Arnoldi algorithm, as implemented in ARPACK. Among other things, this means that after performing an eigenvalue computation with ARPACK or `eigs`, a user can quickly obtain a graphical check to indicate whether the Ritz values returned are likely to be physically meaningful. Our vision is that every ARPACK or `eigs` user ought to plot pseudospectra estimates routinely after their eigenvalue computations as a cheap "sanity check."

Some ideas related to ours have appeared in earlier papers by Nachtigal, Reichel, and Trefethen [17], Ruhe [19], Sorensen [23], Toh [24], and Toh and Trefethen [25]. For example, Sorensen plotted level curves of filter polynomials and observed that they sometimes approximated pseudospectra, and Ruhe showed that pseudospectra could be approximated by a rational Krylov method. What is new here is the explicit development of a method for approximating pseudospectra based on ARPACK. Of course, one could also consider the use of different low-dimensional compressions of a matrix problem such as those constructed by the Jacobi–Davidson algorithm [22]. Preliminary experiments, not reported here, show that this kind of Jacobi–Davidson approximation of pseudospectra can also be effective.

We start by giving an overview of pseudospectra calculations and the implicitly restarted Arnoldi iteration, followed by the practical details of our implementation along with a discussion of some of the problems we have had to deal with. After this we give some examples of the technique in practice. We also mention our MATLAB graphical user interface (GUI), which automates the computation of pseudospectra after the eigenvalues of a matrix have been computed by `eigs` in Matlab.

The computations presented in this paper were all performed using `eigs` in Matlab version 6 (which essentially is ARPACK) and our GUI rather than the Fortran ARPACK, although our initial experiments were done with the Fortran code.

2. Pseudospectra. There are several equivalent ways of defining $\Lambda_\epsilon(A)$, the ϵ-pseudospectrum of a matrix A. The two most important (see, e.g., [27]) are perhaps

$$\Lambda_\epsilon(A) = \{z \in \mathbb{C} : \|(zI - A)^{-1}\| \geq \epsilon^{-1}\} \tag{2.1}$$

and

$$\Lambda_\epsilon(A) = \{z \in \mathbb{C} : z \in \Lambda(A + E) \text{ for some } E \text{ with } \|E\| \leq \epsilon\}. \tag{2.2}$$

When the norms are taken to be the 2-norm, the definitions are equivalent to

$$\Lambda_\epsilon(A) = \{z \in \mathbb{C} : \sigma_{\min}(zI - A) \leq \epsilon\}, \tag{2.3}$$

where $\sigma_{\min}(\cdot)$ denotes minimum singular value. This provides the basis of many algorithms for computing pseudospectra. The most familiar technique is to use a grid over the region of the complex plane of interest and calculate the minimum singular value of $zI - A$ at each grid point z. These values can then be passed to a contour plotter to draw the level curves. For the rest of this paper, we consider the 2-norm; other norms are discussed in [8, 28].

The reason for the cost of computation of pseudospectra is now clear: the amount of work needed to compute the minimum singular value of a general matrix of dimension N is $\mathcal{O}(N^3)$ (see, e.g., [5]). However, several techniques have been developed to reduce this cost [28]. Here are two important ones.

(I) Project the matrix onto a lower dimensional invariant subspace via, e.g., a partial Schur factorization (Reddy, Schmid, and Henningson [18]). This works well when the interesting portion of the complex plane excludes a large fraction of the eigenvalues of the matrix. In this case, the effect of the omitted eigenvalues on the interesting portion of the pseudospectra is typically small, especially if the undesired eigenvalues are well conditioned. Projection can significantly reduce the size of the matrix whose pseudospectra we need to compute, making the singular value computation dramatically faster. In general, the additional cost of projecting the matrix is much less than the cost of repeatedly computing the smallest singular value for the shifted original matrix.

(II) Perform a single matrix reduction to Hessenberg or triangular form before doing any singular value decompositions (Lui [12]), allowing the singular value calculations to be done using a more efficient algorithm.

One way of combining these ideas is to do a complete Schur decomposition of the matrix, $A = UTU^*$, and then to reorder the diagonal entries of the triangular matrix to leave the "wanted" eigenvalues at the top. The reordered factorization can then be truncated leaving the required partial Schur factorization. We can now find the singular values of the matrices shifted for each grid point z using either the original matrix A or the triangular matrix T:

$$\sigma(zI - A) = \sigma(zUIU^* - UTU^*) = \sigma(U(zI - T)U^*) = \sigma(zI - T). \tag{2.4}$$

This allows us to work solely with the triangular matrix T once the $\mathcal{O}(N^3)$ factorization has been completed. The minimum singular value of $zI - T$ can be determined in $\mathcal{O}(N^2)$ operations[2] using the fact that $\sigma_{\min}(zI - T) = \sqrt{\lambda_{\min}((zI - T)^*(zI - T))}$. This can be calculated using either inverse iteration or inverse Lanczos iteration, which require solutions to systems of equations with the matrix $(zI - T)^*(zI - T)$. These can be solved in two stages, each using triangular system solves.

By combining these techniques with more subtle refinements we have an algorithm which is much more efficient than the straightforward method. It is suggested in [28] that the speedup obtained is typically a factor of about $N/4$, assuming the cost of the Schur decomposition is negligible compared with that of the rest of the algorithm. This will be the case on a fine grid for a relatively small matrix (N of the order of a thousand or less), but for larger matrices the Schur decomposition is relatively expensive, and it destroys any sparsity structure.

3. Arnoldi iteration.

The Arnoldi iteration for a matrix A of dimension N works by projecting A onto successive Krylov subspaces $\mathcal{K}_j(A, v_1) \subseteq \mathbb{C}^N$ of dimension j for some starting vector v_1 [1, 20]. It builds an orthonormal basis for the Krylov subspace by the Arnoldi factorization

$$AV_j = V_j H_j + f_j e_j^* = V_{j+1}\widetilde{H}_j, \tag{3.1}$$

where H_j is an upper Hessenberg matrix of dimension j, the columns of V_j form an orthonormal basis for the Krylov subspace, and f_j is orthogonal to the columns of V_j. The residual term $f_j e_j^*$ can be incorporated into the first term $V_j H_j$ by augmenting the matrix H_j with an extra row, all zeros except for the last entry, which is $\|f_j\|$, and including the next basis vector, $v_{j+1} = f_j/\|f_j\|$, in the matrix V_j. The matrix $\widetilde{H}_j$ is now rectangular, of size $(j+1) \times j$.

The matrix $\widetilde{H}_j$, being rectangular, does not have any eigenvalues, but we can define its pseudospectra in terms of singular values by (2.3). (With $\epsilon = 0$, we recover the definition occasionally used that λ is an eigenvalue of a rectangular matrix A if $A - \lambda I$ is rank-deficient, where I is the rectangular matrix of appropriate dimension with 1 on the diagonal and 0 elsewhere. In general, a rectangular matrix will have no eigenvalues, but it will have nonempty ϵ-pseudospectra for large enough ϵ.) It can then be shown [14, 25] that the pseudospectra of successive $\widetilde{H}_j$ are nested.

THEOREM 3.1. *Let A be an $N \times N$ matrix which is unitarily similar to a Hessenberg matrix H, and let $\widetilde{H}_j$ denote the upper left $(j+1) \times j$ section (in particular $\widetilde{H}_j$ could be created using a restarted Arnoldi iteration). Then for any $\epsilon \geq 0$,*

$$\Lambda_\epsilon(\widetilde{H}_1) \subseteq \Lambda_\epsilon(\widetilde{H}_2) \subseteq \cdots \subseteq \Lambda_\epsilon(A). \tag{3.2}$$

Thus as the iteration progresses, the pseudospectra of the rectangular Hessenberg matrices better approximate those of A, which gives some justification for the approximation $\Lambda_\epsilon(A) \approx \Lambda_\epsilon(\widetilde{H}_j)$. Unfortunately, this is only the case for the rectangular matrices $\widetilde{H}_j$. There do not appear to be any satisfactory theorems to justify a similar approximation for the square matrices H_j, and of course for ϵ sufficiently small the ϵ-pseudospectra of H_j must be disjoint from those of A, since they will be small sets surrounding the eigenvalues of H_j, which are in general distinct from those of A. This is not the case for the rectangular matrix as there will not be points in the complex plane with infinite resolvent norm unless a Ritz value exactly matches an

[2] An $\mathcal{O}(N^2)$ algorithm can also be used for Hessenberg matrices [12], but for those we do not have the advantage of projection.

eigenvalue of the original matrix. That is, $\Lambda_\epsilon(\widetilde{H}_j)$ is typically empty for sufficiently small ϵ.

Although the property (3.2) is encouraging, theorems guaranteeing rapid convergence in all cases cannot be expected. The quality of the approximate pseudospectra depends on the information in the Krylov subspace, which in turn depends on the starting vector v_1. Any guarantee of rapid convergence could at best be probabilistic.

3.1. Implicitly restarted Arnoldi. In its basic form, the Arnoldi process may require close to N iterations before the subspace contains good information about the eigenvalues of interest. However, the information contained within the Hessenberg matrix is very dependent on the starting vector v_1: if v_1 contains relatively small components of the eigenvectors corresponding to the eigenvalues which are *not* required, convergence may be quicker and the subspace size need not grow large. To avoid the size of the subspace growing too large, practical implementations of the Arnoldi iteration restart when the subspace size j reaches a certain threshold [20]. A new starting vector $\hat{v}_1$ is chosen which has smaller components in the directions of eigenvectors corresponding to unwanted eigenvalues, and the process is begun again.

Implicit restarting [11, 23] is based upon the same idea, except that subspace is only implicitly compressed to a single starting vector $\hat{v}_1$. What is explicitly formed is an Arnoldi factorization of size k based on this new starting vector, where k is the number of desired eigenvalues, and this Arnoldi factorization is obtained by carrying out $p - k$ implicitly shifted steps of the QR algorithm, with shifts possibly corresponding to unwanted eigenvalue estimates. The computation now proceeds in an accordion-like manner, expanding the subspace to its maximum size p, then compressing to a smaller subspace.[3] This is computationally more efficient than simple restarting because the subspace is already of size k when the iteration restarts, and in addition, the process is numerically stable due to the use of orthogonal transformations in performing the restarting. This technique has made the Arnoldi iteration competitive for finding exterior eigenvalues of a wide range of nonsymmetric matrices.

3.2. Arnoldi for pseudospectra. In a 1996 paper, Toh and Trefethen [25] demonstrated that the Hessenberg matrix created during the Arnoldi process can sometimes provide a good approximation to the pseudospectra of the original matrix. They provided results for both the square matrix H_j and the rectangular matrix $\widetilde{H}_j$. We choose to build our method around the rectangular Hessenberg matrices $\widetilde{H}_j$, even though this makes the pseudospectral computation harder than if we worked with the square matrix. The advantage of this is that we retain the properties of Theorem 3.1, and the following in particular:

For every $\epsilon \geq 0$, the approximate ϵ-pseudospectrum generated by our ARPACK algorithm is a subset of the ϵ-pseudospectrum of the original matrix,

$$\Lambda_\epsilon(\widetilde{H}_p) \subseteq \Lambda_\epsilon(A). \tag{3.3}$$

This is completely different from the familiar situation with Ritz values, which are, after all, the points in the 0-pseudospectrum of a square Hessenberg matrix. Ritz values need not be contained in the true spectrum. Simply by adding one more row to consider a rectangular matrix, we have obtained a guaranteed inclusion for every ϵ.

The results presented by Toh and Trefethen focus on trying to approximate the full pseudospectra of the matrix (i.e., around the entire spectrum) and they do not use

[3] Our use of the variable p follows `eigs`. In ARPACK and in Sorensen's original paper [23], this would be $p + k$.

any kind of restarting in their implementation of the Arnoldi iteration. While this is a useful theoretical idea, we think it is of limited practical value for computing highly accurate pseudospectra since good approximations are often obtained generally only for large subspace dimensions.

Our work is more local; we want a good approximation to the pseudospectra in the region around the eigenvalues requested from ARPACK or `eigs`. By taking advantage of ARPACK's implicit restarting, we keep the size of the subspace (and hence $\widetilde{H}_p$) reasonably small, allowing us to compute (local) approximations to the pseudospectra more rapidly, extending the idea of [25] to a fully practical technique (for a more restricted problem).

4. Implementation. In deciding to use the rectangular Hessenberg matrix $\widetilde{H}_j$, we have made the post-ARPACK phase of our algorithm more difficult. While the simple algorithm of computing the minimum singular value of $zI-A$ at each point has approximately the same cost for a rectangular matrix as a square one, the speedup techniques described in section 2 are difficult to translate into the rectangular case.

The first idea, projection to a lower dimensional invariant subspace, does not make sense for rectangular matrices because there is no such thing as an invariant subspace. The second idea, preliminary triangularization using a Schur decomposition, also does not extend to rectangular matrices, for although it is possible to triangularize the rectangular matrix while keeping the same singular values (by performing a QR factorization, for example), doing so destroys the vital property of shift-invariance (see (2.4)).

However, our particular problem has a feature we have not yet considered: the matrix is Hessenberg. One way to exploit this property is to perform a QR factorization of the matrix obtained after shifting for each grid point. The upper triangular matrix R has the same singular values as the shifted matrix, and they are also unchanged on removing the last row of zeros, which makes the matrix square. We can now use the inverse Lanczos iteration as in section 2 to find its smallest singular value. The QR factorization can be done with an $\mathcal{O}(N^2)$ algorithm (see, e.g., [5, p. 228]), which makes the overall cost $\mathcal{O}(N^2)$. Unfortunately, the additional cost of the QR factorization at each stage makes this algorithm slightly slower for the small matrices (dimensions 50–150) output from ARPACK than for square matrices of the same size, but this appears to be the price to be paid for using matrices which have the property of (3.3).

4.1. Refinements. In some cases we have found that inverse iteration to find the minimum eigenvalue of $(zI-R)^*(zI-R)$ is more efficient than inverse Lanczos iteration but only when used with continuation (Lui [12]). Continuation works by using the vector corresponding to the smallest singular value from the previous grid point as the starting guess for the next grid point.

This sounds like a good idea; if the two shifted matrices differ by only a small shift, their singular values (and singular vectors) will be similar. When it works, it generally means that only a single iteration is needed to satisfy the convergence criterion. However, as Lui indicates, there is a problem with this approach if the smallest and second smallest singular values "change places" between two values of z: the iteration may converge to the second smallest singular value instead of the smallest, since the starting vector had such a strong component in the direction of the corresponding singular vector. This leads to the convergence criterion being satisfied for the wrong singular value (even after several iterations).

```
Choose max subspace size $p$—larger $p$ for better pseudospectra.
Choose number of eigenvalues $k$—larger $k$ for better pseudospectra.
Run ARPACK($A, p, k$) to obtain $\widetilde{H}_p$.
Define a grid over a region of $\mathbb{C}$ enclosing converged Ritz values.
For each grid point $z$:
    Perform reduced QR factorization of shifted matrix: $z\tilde{I} - \widetilde{H}_p = \widetilde{Q}_p R$.
    Get $\lambda_{\max}(z)$ from Lanczos iteration on $(R^*R)^{-1}$, random starting vector.
    $\sigma_{\min}(z) := 1/\sqrt{\lambda_{\max}(z)}$.
end.
Start GUI and create contour plot of the $\sigma_{\min}$ values.
Allow adjustment of parameters (e.g., grid size, contour levels) in GUI.
```

FIG. 2. *Pseudocode for our algorithm.*

In the course of our research, we have found several test matrices which suffer from this problem, including the so-called Tolosa matrix [4]. Accordingly, because of our desire to create a robust algorithm, we do not use inverse iteration. In theory it is also possible to use continuation with inverse Lanczos iteration, but our experiments indicate that the benefit is small and it again brings a risk of misconvergence.

Our algorithm (the main loop of which is similar to that in [28]) is summarized in Figure 2.

5. Practical examples.

While one aim of our method is to approximate the pseudospectra of the original matrix accurately, this is perhaps no more important than the more basic mission of exhibiting the degree of nonnormality the matrix has, so that the ARPACK or `eigs` user gets some idea of whether the Ritz values returned are likely to be physically meaningful. Even in cases where the approximations of the sets $\Lambda_\epsilon(A)$ are inaccurate, a great deal may still be learned from their qualitative properties.

In the following examples, ARPACK was asked to look for the eigenvalues of largest real part except where otherwise indicated. However, the choice of region of the complex plane to focus on is unimportant for our results and is determined by which eigenvalues are of interest for the particular problem at hand. The number of requested eigenvalues k was chosen rather arbitrarily to be large enough so that the approximate pseudospectra clearly indicate the true behavior in the region of the complex plane shown, and the maximum subspace size p was chosen to ensure convergence of ARPACK for the particular choice of k. Experiments show that the computed pseudospectra are not very sensitive to the choices of k and p, provided they are large enough, but we have not attempted to optimize these choices.

5.1. Two extremes.

Our first example (Figure 3), from Matrix Market [15], shows a case where the approximation is extremely good. The matrix is the Jacobi matrix of dimension 800 for the reaction-diffusion Brusselator model from chemical engineering [7], and one seeks the rightmost eigenvalues. The matrix is not highly nonnormal, and the pseudospectra given by the approximation almost exactly match the

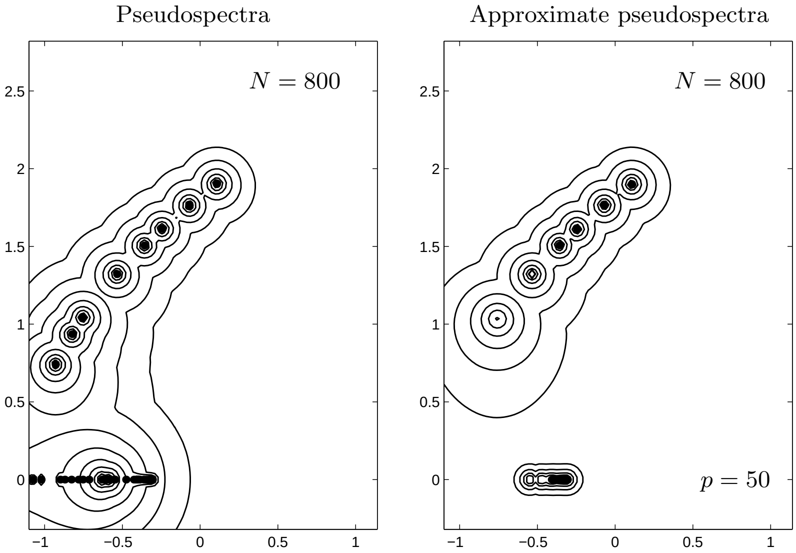


FIG. 3. *Pseudospectra for the matrix* `rdb800l` *(left) computed using the standard method, and pseudospectra of the upper Hessenberg matrix of dimension* $p = 50$ *computed using ARPACK (right) in about* 9% *of the computer time (on the fine grid used here). Levels are shown for* $\epsilon = 10^{-1}, 10^{-1.2}, \ldots, 10^{-2.4}$. *The number of matrix-vector products needed by ARPACK* (n_v) *is* 1,493.

true pseudospectra around the converged Ritz values. This is a case where the pseudospectra computed after running ARPACK indicate that the eigenvalues returned are both accurate and physically meaningful, and that no further investigation is necessary. In this computation we used a maximum subspace dimension of $p = 50$ and requested $k = 30$ eigenvalues.

The second case we consider is one where the matrix has a high degree of nonnormality—the Grcar matrix. As seen in Figure 1, ARPACK can converge to Ritz values which are eigenvalues of a perturbation of order machine precision of the original matrix, and the nonnormality of this particular matrix (here of dimension 400) means that the Ritz values found can lie a long way from the spectrum of the matrix. Figure 4 shows that the pseudospectra of the Hessenberg matrix (computed using $p = 50$, $k = 45$, and asking for eigenvalues of largest modulus) in this case are not good approximations to the pseudospectra of the original one.

This is typical for highly nonnormal matrices—the Hessenberg matrix cannot capture the full extent of the nonnormality, particularly when more than p eigenvalues of the original matrix lie within the region of the complex plane in which the pseudospectra are computed. In other words, the approximation is typically not so good in areas away from the Ritz values computed, and then only accurately approximates the pseudospectra of the original matrix when the Ritz values are good approximations to the eigenvalues. Despite this, a plot like that of Figure 4 will instantly indicate to the ARPACK user that the matrix at hand is strongly nonnormal and needs further investigation.

5.2. A moderately nonnormal example. While the above examples show two extreme cases, many important applications are more middle-of-the-range, where

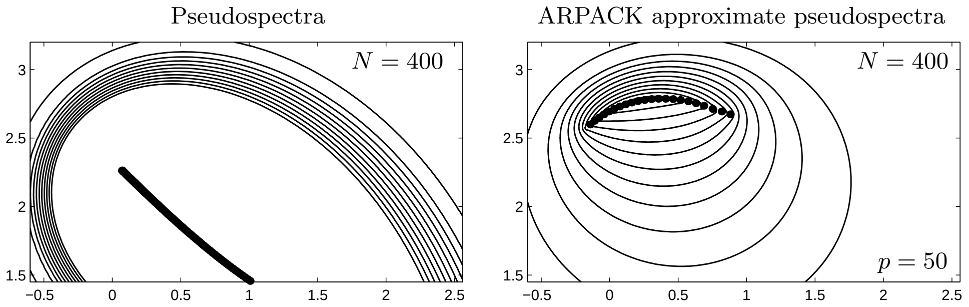


Fig. 4. *The pseudospectra of the Grcar matrix of dimension* 400 *(left) computed using the standard method, and the pseudospectra of the upper Hessenberg matrix of dimension* 50 *computed using ARPACK (right) in about* 8% *of the computer time (on this fine grid). Contours are shown for* $\epsilon = 10^{-1}, 10^{-2}, \ldots, 10^{-11}$*, and* $n_v =$ 8,284.

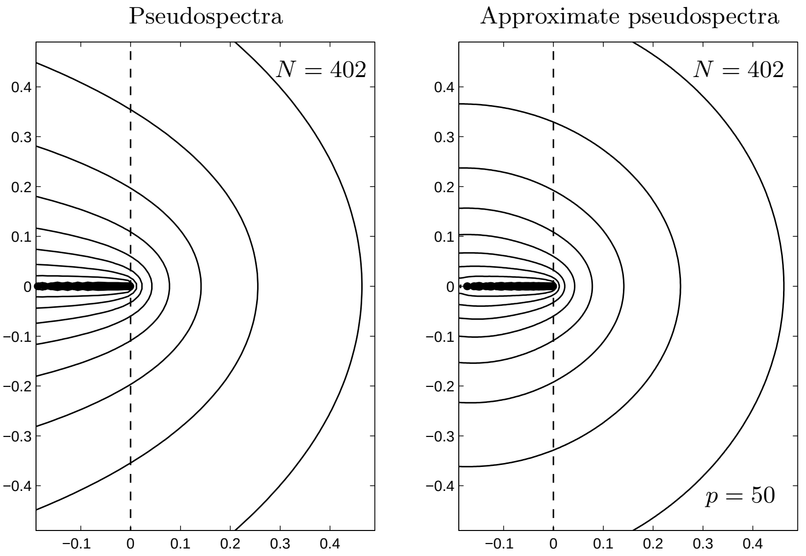


Fig. 5. *Pseudospectra for* $\epsilon = 10^{-1}, 10^{-1.5}, \ldots, 10^{-5}$ *for linearized fluid flow through a circular pipe at Reynolds number* 10,000 *(streamwise-independent disturbances with azimuthal wave number* 1*),* $n_v =$ 708,678.

machine precision is sufficient to accurately converge the eigenvalues, but pronounced nonnormality may nevertheless diminish the physical significance of some of them. A good example of a case in which this is important is the matrix created by linearization about the laminar solution of the Navier–Stokes equations for fluid flow in an infinite circular pipe [26]. (Our matrix is obtained by a Petrov–Galerkin spectral discretization of the Navier–Stokes problem due to Meseguer and Trefethen [16]. The axial and azimuthal wave numbers are 0 and 1, respectively, and the matrix dimension is 402.) The pseudospectra are shown in Figure 5, and although the eigenvalues all have negative real part, implying stability of the flow, the pseudospectra protrude far into the right half-plane. This implies pronounced transient growth of some perturbations of the velocity field in the pipe, which in the presence of nonlinearities in practice may lead to transition to turbulence [30]. The approximate pseudospectra also highlight

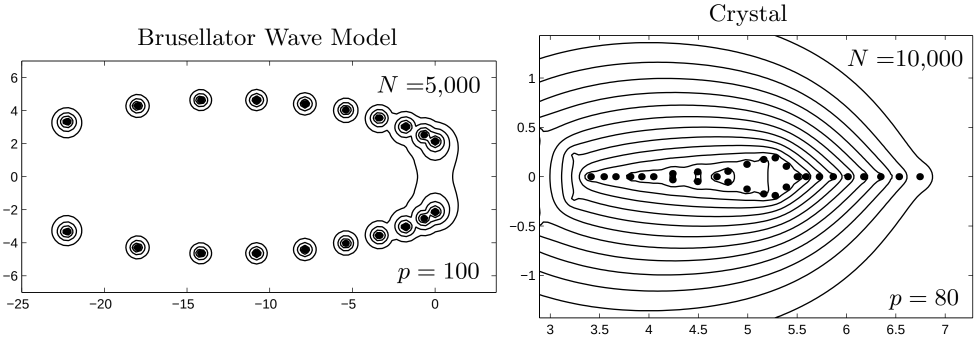


FIG. 6. *Left: The* $\epsilon = 10^{-0.2}, 10^{-0.4}, \ldots, 10^{-1.6}$ *pseudospectra for the Brusselator wave model,* $n_v = 16{,}906$*. Right: Pseudospectral contours for* $\epsilon = 10^{-1}, 10^{-2}, \ldots, 10^{-13}$ *for a matrix of dimension* 10,000 *from the Crystal set at Matrix Market,* $n_v = 22{,}968$*.*

this behavior. The parameters used here were $p = 50$ and $k = 25$.

5.3. Larger examples. We now consider four larger examples. The first is the Brusselator wave model from Matrix Market (not to be confused with the very first example), which models the concentration waves for reaction and transport interaction of chemical solutions in a tubular reactor [9]. Stable periodic solutions exist for a parameter when the rightmost eigenvalues of the Jacobian are purely imaginary. For a matrix of dimension 5,000, using a subspace of dimension 100 and asking ARPACK for 20 eigenvalues, we obtained the eigenvalue estimates and approximate pseudospectra shown in Figure 6 (left). The departure from normality is evidently mild, and the conclusion from this computation is that the Ritz values returned by ARPACK are likely to be accurate and the corresponding eigenvalues physically meaningful.

Figure 6 (right) shows approximate pseudospectra for a matrix of dimension 10,000, taken from the Crystal set at Matrix Market, which arises in a stability analysis of a crystal growth problem [32]. The eigenvalues of interest are the ones with largest real part. The fact that we can see the 10^{-13} pseudospectrum (when the axis scale is $\mathcal{O}(1)$) indicates that this matrix is significantly nonnormal, and although the matrix is too large for us to be able to compute its exact pseudospectra for comparison, this is certainly a case where the nonnormality could be important, making all but the rightmost few eigenvalues of dubious physical significance in an application. The ARPACK parameters we used in this case were $p = 80$ and $k = 30$, and the computation took about one hour on our Sun Ultra 5 workstation. Although we do not have the true pseudospectra in this case, we would expect that the rightmost portion should be fairly accurate where there is a good deal of Ritz data and relatively little nonnormality. We expect that the leftmost portion is less accurate where the effect of the remaining eigenvalues of the matrix unknown to the approximation begins to become important.

The third example, Figure 7 (left), shows the Airfoil matrix created by performing transient stability analysis of a Navier–Stokes solver [13], also from Matrix Market. In this case the matrix appears fairly close to normal, and the picture gives every reason to believe that the eigenvalues have physical meaning. Using $p = 80$ and $k = 20$, ARPACK took about 9 hours to converge to the eigenvalues, while we were able to plot the pseudospectra in about 3 minutes (even on the fine grid used here).

Our final example is a matrix which is bidiagonal plus random sparse entries

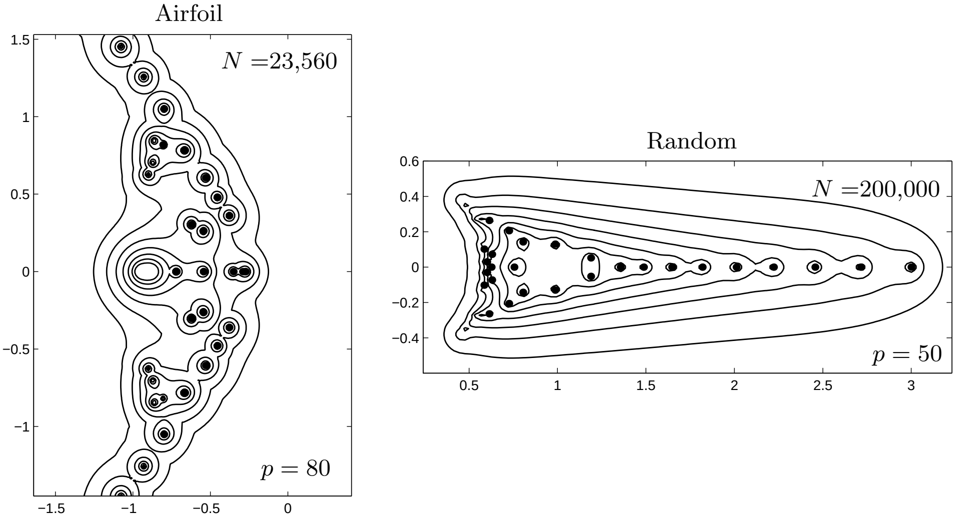


FIG. 7. *Left: The* $\epsilon = 10^{-1}, 10^{-1.25}, \ldots, 10^{-2.5}$ *pseudospectra for the Airfoil matrix from Matrix Market of dimension* 23,560*,* $n_v = 72{,}853$*. Right: The* $\epsilon = 10^{-1}, 10^{-1.5}, \ldots, 10^{-4.5}$ *pseudospectra for our random matrix, with* $n_v = 61{,}347$*.*

elsewhere, created in MATLAB by

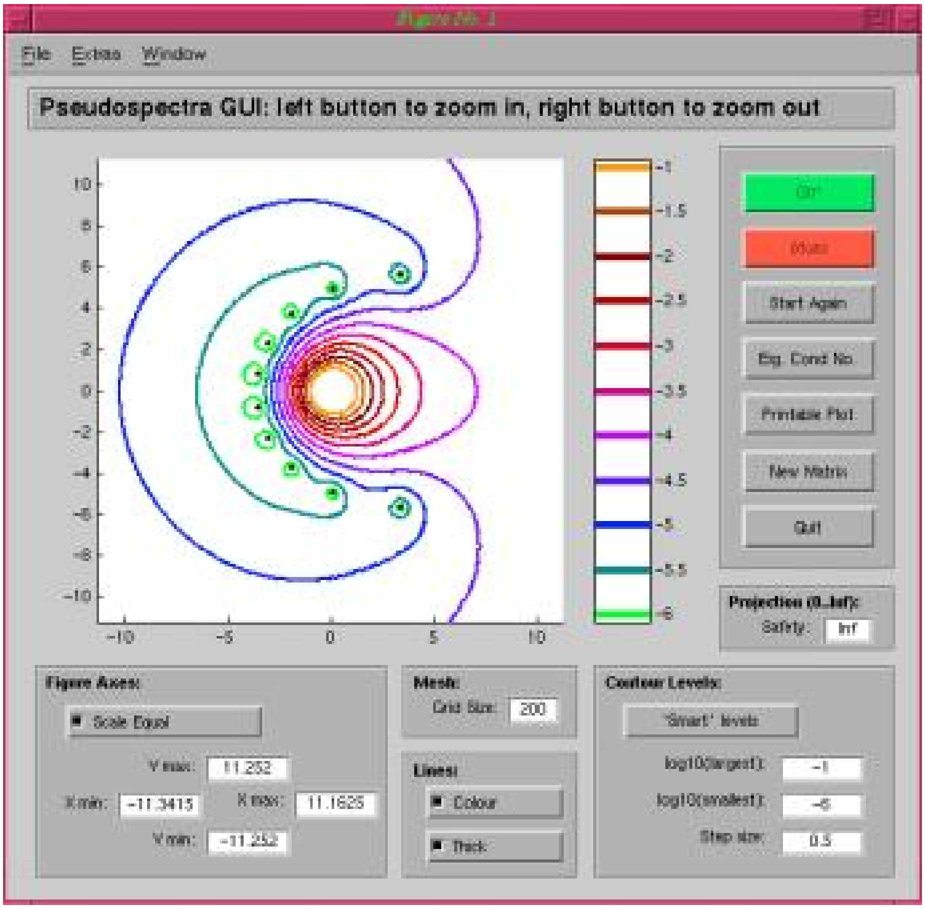
```
A = spdiags([3*exp(-(0:N-1)'/10) .5*ones(N,1)], 0:1, N, N)
      + .1*sprandn(N,N,10/N);
```

The approximation to the pseudospectra of the matrix of dimension 200,000 is shown in Figure 7 (right), from which we can conclude that the eigenvalue estimates returned are probably accurate, but that the eigenvalues toward the left of the plot would likely be of limited physical significance in a physical application, if there were one, governed by this matrix. We used a subspace size of 50 and requested 30 eigenvalues from this example, and the whole computation took about 26 hours.

6. MATLAB GUI. We have created a MATLAB GUI to automate the process of computing pseudospectra, and Figure 8 shows a snapshot after a run of ARPACK. Initially the pseudospectra are computed on a coarse grid to give a fast indication of the nonnormality of the matrix, but the GUI allows control over the number of grid points if a higher quality picture is desired. Other features include the abilities to change the contour levels shown without recomputing the underlying values, and to select parts of the complex plane to zoom in for greater detail. The GUI can also be used as a graphical front end to our other pseudospectra codes for computing pseudospectra of smaller general matrices. The codes are available on the World Wide Web from http://web.comlab.ox.ac.uk/oucl/work/nick.trefethen/.

7. Discussion. The examples of section 5 give an indication of the sort of pictures obtained from our technique. For matrices fairly close to normal, the approximation is typically a very close match to the exact pseudospectra, but for more highly nonnormal examples the agreement is not so close. This is mainly due to the effect of eigenvalues which the Arnoldi iteration has not found: their effect on the pseudospectra is typically more pronounced for nonnormal matrices.

The other point to note is that if we use the Arnoldi iteration to look (for ex-



Pseudospectra GUI: left button to zoom in, right button to zoom out

Projection (0..Inf):
Safety: Inf

Figure Axes:
■ Scale Equal
Y max: 11.252
X min: -11.3415 X max: 11.1625
Y min: -11.252

Mesh:
Grid Size: 200

Lines:
■ Colour
■ Thick

Contour Levels:
log10(largest): −1
log10(smallest): −6
Step size: 0.5

FIG. 8. *A snapshot of the* MATLAB *GUI after computing the pseudospectra of a matrix.*

ample) for eigenvalues of largest real part, the rightmost part of the approximate pseudospectra will be a reasonably good approximation. This can clearly be seen in Figure 4 where we are looking for eigenvalues of largest modulus: the top parts of the pseudospectra are fairly good and only deteriorate lower down where the effect of the "unfound" eigenvalues becomes important.

However, as mentioned in the introduction, creating accurate approximations of pseudospectra was only part of the motivation for this work. Equally important has been the goal of providing information which can help the user of ARPACK or `eigs` decide whether the computed eigenvalues are physically meaningful. For this purpose, estimating the degree of nonnormality of the matrix is more important than getting an accurate plot of the exact pseudospectra.

One of the biggest advantages of our technique is that while the time spent on the ARPACK computation grows as the dimension of the matrix increases, the time spent on the pseudospectra computation remains roughly constant. This is because the pseudospectra computation is based just on the final Hessenberg matrix, of dimension typically in the low hundreds at most. Figure 9 shows the proportion of time spent on the pseudospectra part of the computation for the examples we have presented here. These timings are based on the time to compute the initial output from our

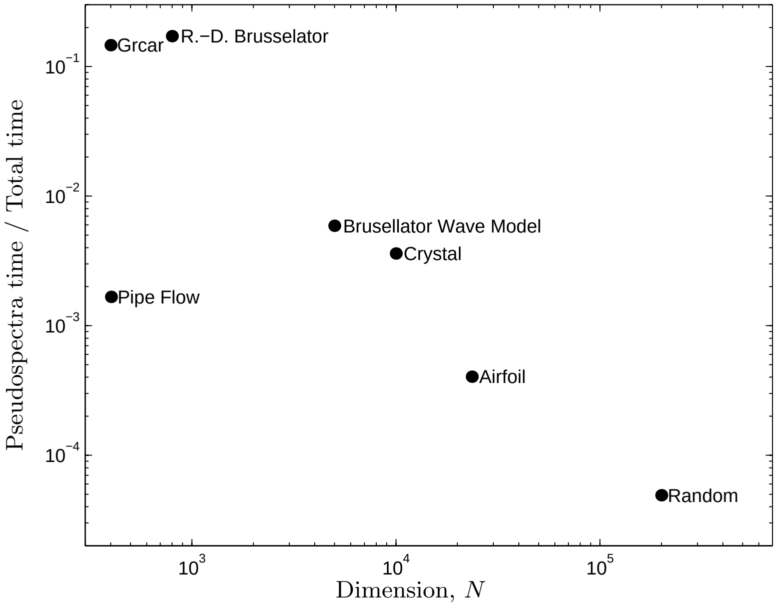


FIG. 9. *The proportion of the total computation time spent on computing the pseudospectra for the examples presented in this paper. For large N, the pseudospectra are obtained at very little additional cost.*

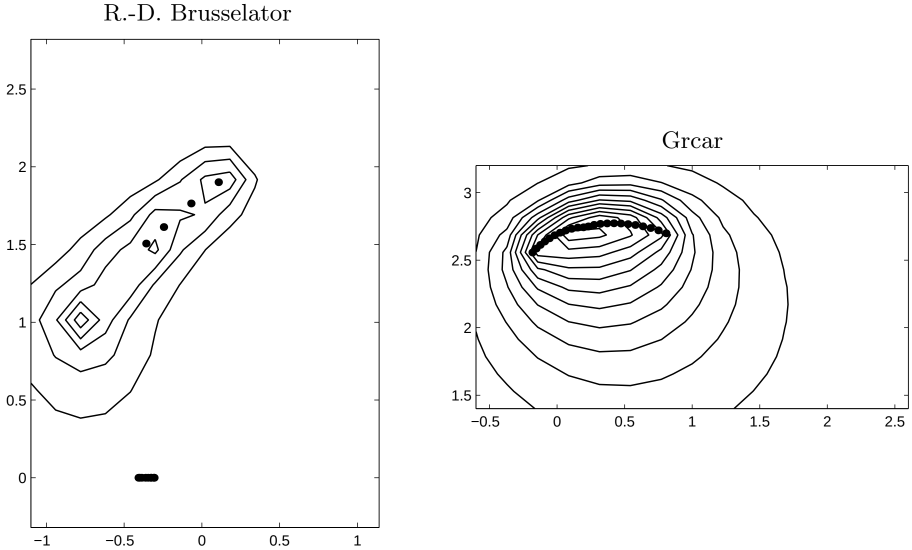


FIG. 10. *Lower-quality plots of pseudospectra produced by our GUI on its default coarse* 15×15 *grid. Such plots take just a few seconds. The matrices shown are* `rdb800l` *with* $\epsilon = 10^{-1}, 10^{-1.2}, \ldots, 10^{-1.8}$ *(cf. Figure* 3*) and the Grcar matrix with* $\epsilon = 10^{-1}, 10^{-2}, \ldots, 10^{-10}$ *(cf. Figure* 4*).*

GUI using a coarse grid such as those illustrated in Figure 10, which is our standard resolution for day-to-day work. For the "publication quality" pseudospectra of the resolution of the other plots in this paper, the cost is about thirty times higher, but this is still much less than the cost of ARPACK for dimensions N in the thousands.

Acknowledgments. We would like to thank Rich Lehoucq for his advice on ARPACK beginning during his visit to Oxford in October–November 1999, Penny Anderson of The MathWorks, Inc., for her help with the beta version of the new `eigs` command, and Mark Embree for his comments on drafts of the paper.

REFERENCES

[1] W. E. Arnoldi, *The principle of minimized iteration in the solution of the matrix eigenvalue problem*, Quart. Appl. Math., 9 (1951), pp. 17–29.

[2] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, eds., *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*, SIAM, Philadelphia, 2000.

[3] T. Braconnier and N. J. Higham, *Computing the field of values and pseudospectra using the Lanczos method with continuation*, BIT, 36 (1996), pp. 422–440.

[4] F. Chatelin and S. Godet-Thobie, *Stability analysis in aeronautical industries*, in Proceedings of the 2nd Symposium on High-Performance Computing, Montpellier, France, M. Durand and F. E. Dabaghi, eds., Elsevier–North-Holland, Amsterdam, 1991.

[5] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed., The Johns Hopkins University Press, Baltimore, MD, London, UK, 1996.

[6] J. F. Grcar, *Operator Coefficient Methods for Linear Equations*, Tech. report SAND89-8691, Sandia National Labs, Livermore, CA, 1989.

[7] B. Hassard, N. Kazarinoff, and Y. Wan, *Theory and Applications of Hopf Bifurcation*, Cambridge University Press, Cambridge, UK, 1981.

[8] N. J. Higham and F. Tisseur, *A block algorithm for matrix* 1*-norm estimation, with an application to* 1*-norm pseudospectra*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1185–1201.

[9] P. J. Holmes, ed., *Waves in Distributed Chemical Systems: Experiments and Computations*, Proceedings of the Asilomar Conference Ground, Pacific Grove, CA, 1979, SIAM, Philadelphia, 1980.

[10] R. B. Lehoucq, K. Maschhoff, D. Sorensen, and C. Yang, *ARPACK Software Package*, http://www.caam.rice.edu/software/ARPACK/, 1996.

[11] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK Users' Guide*, SIAM, Philadelphia, 1998.

[12] S. H. Lui, *Computation of pseudospectra by continuation*, SIAM J. Sci. Comput., 18 (1997), pp. 565–573.

[13] A. Mahajan, E. H. Dowell, and D. Blis, *Eigenvalue calculation procedure for an Euler-Navier-Stokes solver with applications to flows over airfoils*, J. Comput. Phys., 97 (1991), pp. 398–413.

[14] T. A. Manteuffel and G. Starke, *On hybrid iterative methods for nonsymmetric systems of linear equations*, Numer. Math., 73 (1996), pp. 489–506.

[15] *Matrix Market*, http://math.nist.gov/MatrixMarket/, 2000.

[16] A. Meseguer and L. N. Trefethen, *A Spectral Petrov-Galerkin Formulation for Pipe Flow* I*: Linear Stability and Transient Growth*, Tech. report NA-00/18, Oxford University Computing Laboratory, Oxford, UK, 2000.

[17] N. M. Nachtigal, L. Reichel, and L. N. Trefethen, *A hybrid GMRES algorithm for nonsymmetric linear systems*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 796–825.

[18] S. C. Reddy, P. J. Schmid, and D. S. Henningson, *Pseudospectra of the Orr–Sommerfeld operator*, SIAM J. Appl. Math., 53 (1993), pp. 15–47.

[19] A. Ruhe, *Rational Krylov algorithms for nonsymmetric eigenvalue problems.* II*. Matrix pairs*, Linear Algebra Appl., 197/198 (1994), pp. 283–295.

[20] Y. Saad, *Variations on Arnoldi's method for computing eigenelements of large unsymmetric matrices*, Linear Algebra Appl., 34 (1980), pp. 269–295.

[21] Y. Saad, *Numerical Methods for Large Eigenvalue Problems*, Manchester University Press, Manchester, UK, 1992.

[22] G. L. G. Sleijpen and H. A. van der Vorst, *A Jacobi–Davidson iteration method for linear eigenvalue problems*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 401–425.

[23] D. C. Sorensen, *Implicit application of polynomial filters in a k-step Arnoldi method*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 357–385.

[24] K.-C. Toh, *Matrix Approximation Problems and Nonsymmetric Iterative Methods*, Ph.D. thesis, Cornell University, Ithaca, NY, 1996.

[25] K.-C. Toh and L. N. Trefethen, *Calculation of pseudospectra by the Arnoldi iteration*, SIAM J. Sci. Comput., 17 (1996), pp. 1–15.

[26] A. E. Trefethen, L. N. Trefethen, and P. J. Schmid, *Spectra and pseudospectra for pipe Poiseuille flow*, Comput. Methods Appl. Mech. Engrg., 175 (1999), pp. 413–420.

[27] L. N. Trefethen, *Pseudospectra of matrices*, in Numerical Analysis 1991 (Dundee, 1991), Longman Sci. Tech., Harlow, UK, 1992, pp. 234–266.

[28] L. N. Trefethen, *Computation of Pseudospectra*, Acta Numer. 8, Cambridge University Press, Cambridge, UK, 1999.

[29] L. N. Trefethen and D. Bau, III, *Numerical Linear Algebra*, SIAM, Philadelphia, 1997.

[30] L. N. Trefethen, A. E. Trefethen, S. C. Reddy, and T. A. Driscoll, *Hydrodynamic stability without eigenvalues*, Science, 261 (1993), pp. 578–584.

[31] H. A. van der Vorst, *Computational Methods for Large Eigenvalue Problems*, North-Holland, Amsterdam, to appear.

[32] C. Yang, D. C. Sorensen, D. I. Meiron, and B. Wedeman, *Numerical Computation of the Linear Stability of the Diffusion Model for Crystal Growth Simulation*, Tech. report TR96-04, Department of Computational and Applied Mathematics, Rice University, Houston, TX, 1996.

SIAM J. SCI. COMPUT.
Vol. 23, No. 2, pp. 606–628

A JACOBI–DAVIDSON TYPE SVD METHOD*

MICHIEL E. HOCHSTENBACH†

Abstract. We discuss a new method for the iterative computation of a portion of the singular values and vectors of a large sparse matrix. Similar to the Jacobi–Davidson method for the eigenvalue problem, we compute in each step a correction by (approximately) solving a correction equation. We give a few variants of this Jacobi–Davidson SVD (JDSVD) method with their theoretical properties. It is shown that the JDSVD can be seen as an accelerated (inexact) Newton scheme. We experimentally compare the method with some other iterative SVD methods.

Key words. Jacobi–Davidson, singular value decomposition (SVD), singular values, singular vectors, norm, augmented matrix, correction equation, (inexact) accelerated Newton, improving singular values

AMS subject classifications. 65F15 (65F35)

PII. S1064827500372973

1. Introduction.

Suppose that we want to compute one or more singular values, and the corresponding singular vectors, of the real $m \times n$ matrix A. This subject has already been studied from a number of different viewpoints [5, 6, 1, 19, 20, 13], for example, to determine a few of the largest or smallest singular triples. This partial SVD can be computed in two different ways using equivalent eigenvalue decompositions.

The first is to compute some eigenvalues and eigenvectors of the $n \times n$ matrix A^TA or the $m \times m$ matrix AA^T. For large (sparse) matrices, direct methods like the QR method are unattractive, but there exist several iterative methods. In [13], for example, (block) Lanczos [10] and Davidson [2] are applied to A^TA. Another candidate is Jacobi–Davidson [15]. Note that it is in general not advisable (or necessary) to explicitly form the product A^TA. The nonzero eigenvalues of A^TA and AA^T are the squares of the nonzero singular values of A. This works positively for the separation of large singular values, but it forces a clustering of small ones. Moreover, it can be hard to find very small singular values (relative to the largest singular value) accurately. Apart from this, the approaches via A^TA or AA^T are asymmetric: in the process we approximate only one of the two singular vectors. The second vector can be obtained from the first by a multiplication by A or A^T, but this may introduce extra loss of accuracy. Besides, when we have approximations to both the left and right singular vector, we can use only one of them as a starting vector for an iterative method.

A second approach is to compute some eigenvalues and eigenvectors of the *augmented matrix*

$$\begin{pmatrix} 0 & A \\ A^T & 0 \end{pmatrix}. \tag{1.1}$$

This approach has its own advantages and disadvantages. The eigenvalues of the augmented matrix are plus and minus the singular values of A, and we can extract the left and right singular vectors from the eigenvectors by just taking the first and second part (see section 2). This makes an extra multiplication by A or A^T unnecessary. We

*Received by the editors June 5, 2000; accepted for publication (in revised form) January 11, 2001; published electronically July 10, 2001.

http://www.siam.org/journals/sisc/23-2/37297.html

†Mathematical Institute, Utrecht University, P.O. Box 80.010, NL-3508 TA Utrecht, The Netherlands (hochstenbach@math.uu.nl, www.math.uu.nl/people/hochstenbach).

do not have the drawback of squaring small singular values. On the negative side, the augmented matrix is larger in size, and the smallest singular values are in the interior of the spectrum.

The Lanczos method for the augmented matrix has been studied by a number of authors [5, 6, 1]. The Lanczos process does not exploit the special (block or "two-cyclic") structure of the matrix, unless the starting vector is of the form $(u, 0)$ or $(0, v)$. This is essentially Lanczos bidiagonalization of A; see [7, p. 495].

We can also consider the Jacobi–Davidson method [15] for the augmented matrix. This is an efficient method for the computation of a few eigenpairs, and it is of a different nature in comparison to Lanczos. The essence of Jacobi–Davidson is its correction equation, where the shifted operator is restricted to the subspace orthogonal to the current approximation to an eigenvector. When we solve this equation exactly, we can show that the updated vector is the same as the one we would get by one step of Rayleigh quotient iteration (RQI). But in practice one solves the Jacobi–Davidson correction equation only approximately, and one accelerates the convergence by projecting the matrix onto the subspace spanned by all iterates. Therefore, Jacobi–Davidson can also be viewed as an inexact accelerated RQI.

"Standard" Jacobi–Davidson does not make use of the structure of the augmented matrix. In this paper we propose a Jacobi–Davidson variant that *does* take advantage of the special structure of the matrix. Instead of searching the eigenvector in one subspace, we search the left and right singular vectors in separate subspaces. We still solve a correction equation for the augmented matrix, but we use different projections, and we split the approximate solution of this equation for the expansion of the two search spaces. More similarities and differences are discussed in section 7.3.

After some preparations in section 2, we introduce the new approach, which we call the Jacobi–Davidson SVD (JDSVD), in section 3. In section 4, a few variants of the algorithm with their properties are presented. In section 5, we show that the JDSVD process can be viewed as an (inexact) accelerated Newton scheme, and in section 6 we focus on convergence. Various aspects of the method are discussed in section 7, and after numerical examples in section 8, we finish with conclusions in section 9.

2. Preparations. Let A be a real $m \times n$ matrix with SVD $A = U_* \Sigma V_*^T$ and singular values

$$0 \le \sigma_{\min} = \sigma_p \le \sigma_{p-1} \le \cdots \le \sigma_2 \le \sigma_1 = \sigma_{\max},$$

where $p := \min\{m, n\}$. Denote the corresponding left and right singular vectors by $u_{*,j}$ $(1 \le j \le m)$ and $v_{*,j}$ $(1 \le j \le n)$.

Throughout the paper, $\|\cdot\|$ stands for $\|\cdot\|_2$, and we write $\sigma_j(B)$ for the jth largest singular value of a real matrix B and simply σ_j for the jth largest singular value of A. Furthermore, $\Lambda(B)$ is the spectrum of B, and $\Sigma(B)$ is the set of singular values of B. If B is a real symmetric matrix, then $\lambda_j(B)$ denotes the jth largest eigenvalue of B.

If $a \in \mathbb{R}^m$ and $b \in \mathbb{R}^n$, then, for convenience, we write $\binom{a}{b} \in \mathbb{R}^{m+n}$ also as (a, b). If X is a matrix, then we denote the subspace spanned by the columns of X by $\mathcal{X}$. We use the notation $\mathcal{K}_l(B, x)$ for the Krylov subspace of dimension l generated by B and x.

DEFINITION 2.1. *Let $u \in \mathbb{R}^m$, $v \in \mathbb{R}^n$, $\mathcal{X} \subset \mathbb{R}^m$, and $\mathcal{Y} \subset \mathbb{R}^n$. We say that $\binom{u}{v} \in \mathbb{R}^{m+n}$ is double-orthogonal to the pair of subspaces $(\mathcal{X}, \mathcal{Y})$ if both $u \perp \mathcal{X}$*

and $v \perp \mathcal{Y}$, *which is denoted by* $\binom{u}{v} \perp\perp \binom{\mathcal{X}}{\mathcal{Y}}$. *By* $(u,v)^{\perp\perp}$ *we denote the subspace* $\{(a,b) \in \mathbb{R}^m \times \mathbb{R}^n : u^T a = v^T b = 0\}$.

The following lemma gives a relation between the singular triples of A and the eigenpairs of the augmented matrix.

LEMMA 2.2 (Jordan–Wielandt; see Theorem I.4.2 of [18]). *The augmented matrix* (1.1) *has eigenvalues*

$$-\sigma_1, \ldots, -\sigma_p, \underbrace{0, \ldots, 0}_{|m-n|}, \sigma_p, \ldots, \sigma_1$$

and eigenvectors

$$\begin{pmatrix} u_{*,j} \\ \pm v_{*,j} \end{pmatrix} \quad (1 \le j \le p)$$

corresponding to the $\pm\sigma_j$ *and, if* $m \neq n$, *additionally,*

$$\textit{either} \begin{pmatrix} u_{*,j} \\ 0 \end{pmatrix} \ (n+1 \le j \le m) \quad \textit{or} \quad \begin{pmatrix} 0 \\ v_{*,j} \end{pmatrix} \ (m+1 \le j \le n),$$

depending on whether $m > n$ *or* $n > m$.

The next definition is the natural analogue of the definition of a simple eigenvalue (see, e.g., [18, p. 15]).

DEFINITION 2.3. *We call* σ_i *a simple singular value of* A *if* $\sigma_i \neq \sigma_j$ *for all* $j \neq i$.

The following lemma gives a link between a simple singular value of A and a simple eigenvalue of A^TA and AA^T.

LEMMA 2.4. *Let* $\sigma > 0$. *Then* σ *is a simple singular value of* A *if and only if* σ^2 *is a simple eigenvalue of* A^TA *and* AA^T.

Proof. The nonzero eigenvalues of A^TA and AA^T are just the squares of the nonzero singular values of A (see, for example, [18, p. 31]). □

Note that the condition $\sigma > 0$ in the previous lemma is necessary. For example, 0 is a simple singular value of the 1×2 matrix $A = (0\ 0)$, but it is not a simple eigenvalue of A^TA.

For future use, we mention the following well-known results.

LEMMA 2.5 (Weyl; see pp. 101–102 of [21], Corollary IV.4.9 of [18], and Theorem 10.3.1 of [12]). *Let* B *and* E *be real symmetric* $n \times n$ *matrices. Then for all* $1 \le j \le n$

$$\lambda_j(B) + \lambda_n(E) \le \lambda_j(B+E) \le \lambda_j(B) + \lambda_1(E).$$

LEMMA 2.6 (see (3.3.17) of [9]). *If* B *and* E *are* $m \times n$ *matrices, then for* $1 \le i,\ j \le p$, *and* $i + j \ \le p+1$,

$$\sigma_{i+j-1}(B+E) \le \sigma_i(B) + \sigma_j(E).$$

In particular, for $j = 1$ *this yields* $\sigma_i(B+E) \le \sigma_i(B) + \sigma_1(E)$ *for* $i = 1, \ldots, p$.

LEMMA 2.7 (see (7.3.8) of [8]). *Let* B *and* E *be real* $m \times n$ *matrices. Then*

$$\sum_{j=1}^{p} (\sigma_j(B+E) - \sigma_j(B))^2 \le \|E\|_F^2.$$

LEMMA 2.8. *If* U *and* V *are orthogonal* $m \times m$ *and* $n \times n$ *matrices, respectively, then for all* $1 \le j \le p$ *we have* $\sigma_j(U^TAV) = \sigma_j(A)$. *In particular,* $\|U^TAV\| = \|A\|$.

Proof. The SVD of U^TAV is just $(U^TU_*)\Sigma(V^TV_*)^T$. The final statement follows from the characterization of the matrix two-norm as the largest singular value. □

LEMMA 2.9 (see (3.1.3) of [9]). *Let B be an $m \times n$ matrix, and let B_l denote a submatrix of B obtained by deleting a total of l rows and/or columns from B. Then*

$$\sigma_j(B) \geq \sigma_j(B_l) \geq \sigma_{j+l}(B)$$

for $1 \leq j \leq p$, where for a $q \times r$ matrix X we set $\sigma_j(X) = 0$ if $j > \min\{q, r\}$.

3. The JDSVD correction equation. Suppose that we have k-dimensional *search spaces* $\mathcal{U} \subset \mathbb{R}^m$ and $\mathcal{V} \subset \mathbb{R}^n$ and *test spaces* $\mathcal{X} \subset \mathbb{R}^m$ and $\mathcal{Y} \subset \mathbb{R}^n$. To determine approximations θ, η to a singular value, and $u \in \mathcal{U}, v \in \mathcal{V}$ (of unit norm) to the corresponding left and right singular vectors, we impose the *double Galerkin condition* with respect to $\mathcal{X}$ and $\mathcal{Y}$ on the *residual* r:

$$r = r(\theta, \eta) := \begin{pmatrix} Av - \theta u \\ A^Tu - \eta v \end{pmatrix} \perp\perp \begin{pmatrix} \mathcal{X} \\ \mathcal{Y} \end{pmatrix}. \tag{3.1}$$

Because $u \in \mathcal{U}$ and $v \in \mathcal{V}$, we can write $u = Uc$ and $v = Vd$, where the columns of the $m \times k$ matrix U and the columns of the $n \times k$ matrix V form bases for $\mathcal{U}$ and $\mathcal{V}$, respectively, and $c, d \in \mathbb{R}^k$. Then we want to find θ, η, c, and d that are solutions of

$$\left\{ \begin{array}{lcl} X^TAVd & = & \theta\, X^TUc, \\ Y^TA^TUc & = & \eta\, Y^TVd, \end{array} \right. \tag{3.2}$$

where X and Y are matrices with columns that form bases for $\mathcal{X}$ and $\mathcal{Y}$. For *test vectors* $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, we have, in particular, that $r \perp\perp (x, y)$; so if $x^Tu \neq 0$ and $y^Tv \neq 0$,

$$\theta = \frac{x^TAv}{x^Tu}, \qquad \eta = \frac{y^TA^Tu}{y^Tv}. \tag{3.3}$$

This shows that the approximations θ and η may differ. We discuss possible choices for $\mathcal{X}$ and $\mathcal{Y}$ and the resulting relations for u and v in the following section. For now, suppose that we have approximations (u, v, θ, η). We would like to have a double-orthogonal correction $(s, t) \perp\perp (u, v)$ to (u, v) such that

$$\left\{ \begin{array}{lcl} A(v + t) & = & \sigma(u + s), \\ A^T(u + s) & = & \tau(v + t), \end{array} \right. \tag{3.4}$$

where $\sigma > 0$ and $\tau > 0$ need not be equal because the vectors are not normalized. However, since $A^TA(v+t) = \sigma\tau(v+t)$, we have $\sigma\tau = \sigma_i^2$ for some $1 \leq i \leq p$. Equations (3.4) can be rearranged to obtain

$$\begin{pmatrix} -\sigma I_m & A \\ A^T & -\tau I_n \end{pmatrix} \begin{pmatrix} s \\ t \end{pmatrix} = -\begin{pmatrix} Av - \theta u \\ A^Tu - \eta v \end{pmatrix} + \begin{pmatrix} (\sigma - \theta)u \\ (\tau - \eta)v \end{pmatrix} = -r + \begin{pmatrix} (\sigma - \theta)u \\ (\tau - \eta)v \end{pmatrix}.$$

Because σ and τ are unknown, we do not know the differences $(\sigma - \theta)u$ and $(\tau - \eta)v$ either. Therefore, we can consider the projection of the last equation onto $(x, y)^{\perp\perp}$ along (u, v). This projection is given by

$$\begin{pmatrix} I_m - \frac{ux^T}{x^Tu} & 0 \\ 0 & I_n - \frac{vy^T}{y^Tv} \end{pmatrix},$$

and it fixes r. Projecting the previous equation, we get

$$\begin{pmatrix} I_m - \frac{ux^T}{x^Tu} & 0 \\ 0 & I_n - \frac{vy^T}{y^Tv} \end{pmatrix} \begin{pmatrix} -\sigma I_m & A \\ A^T & -\tau I_n \end{pmatrix} \begin{pmatrix} s \\ t \end{pmatrix} = -r, \qquad \text{where } (s,t) \perp\perp (u,v). \tag{3.5}$$

Since σ and τ are unknown, an obvious choice is to replace them by θ and η. This can be considered as "throwing away second order terms" ($\sigma - \theta$, $\tau - \eta$, s, and t will all be asymptotically small) and suggests that the JDSVD is in fact a Newton method, which is true indeed (see section 5). Furthermore, since for every $\widetilde{x} \in \mathbb{R}^m$ and $\widetilde{y} \in \mathbb{R}^n$ such that $u^T\widetilde{x} \neq 0$ and $v^T\widetilde{y} \neq 0$

$$\begin{pmatrix} I_m - \frac{\widetilde{x}u^T}{u^T\widetilde{x}} & 0 \\ 0 & I_n - \frac{\widetilde{y}v^T}{v^T\widetilde{y}} \end{pmatrix} \begin{pmatrix} s \\ t \end{pmatrix} = \begin{pmatrix} s \\ t \end{pmatrix},$$

(3.5) leads to the *JDSVD correction equation*

$$\begin{pmatrix} I_m - \frac{ux^T}{x^Tu} & 0 \\ 0 & I_n - \frac{vy^T}{y^Tv} \end{pmatrix} \begin{pmatrix} -\theta I_m & A \\ A^T & -\eta I_n \end{pmatrix} \begin{pmatrix} I_m - \frac{\widetilde{x}u^T}{u^T\widetilde{x}} & 0 \\ 0 & I_n - \frac{\widetilde{y}v^T}{v^T\widetilde{y}} \end{pmatrix} \begin{pmatrix} s \\ t \end{pmatrix} = -r, \tag{3.6}$$

where $(s,t) \perp\perp (u,v)$. We see that the operator in (3.6) is symmetric if and only if $\widetilde{x}$ and $\widetilde{y}$ are a nonzero multiple of x and y. It maps $(u,v)^{\perp\perp}$ to $(x,y)^{\perp\perp}$. In sections 5 and 6 we explain why this process may lead to fast convergence, and we meet a generalized version of the correction equation. In the next section we examine several choices for the Galerkin conditions (3.1).

4. Choices for the Galerkin conditions.

Consider the eigenvalue problem for a symmetric matrix B, where we have one subspace $\mathcal{W}$ that is used both as search space and test space. If the columns of W form an orthonormal basis for $\mathcal{W}$, then the projected matrix W^TBW has some nice properties; see [12, section 11.4]. We will see that searching in two spaces, as in the JDSVD, spreads those properties over a few Galerkin choices. In this section we examine some obvious choices.

4.1. The standard choice $\mathcal{X} = \mathcal{U}$ and $\mathcal{Y} = \mathcal{V}$.

Consider the situation where the search spaces $\mathcal{U}$ and $\mathcal{V}$ are of equal dimension k. Let us first take the test spaces $\mathcal{X}$ and $\mathcal{Y}$ equal to the search spaces $\mathcal{U}$ and $\mathcal{V}$.

If the columns of U and V form orthonormal bases for $\mathcal{U}$ and $\mathcal{V}$, then with the notation $H := U^TAV$, (3.2) reduces to

$$Hd = \theta c \quad \text{and} \quad H^Tc = \eta d. \tag{4.1}$$

This gives approximations $u = Uc$ and $v = Vd$, where c and d are, respectively, left and right singular vectors of H. With the requirement $\|c\| = \|d\| = 1$ and test vectors $x = u$ and $y = v$, we get

$$\theta = \eta = u^TAv. \tag{4.2}$$

For reasons of symmetry, we choose $\widetilde{x} = x$ $(= u)$ and $\widetilde{y} = y$ $(= v)$. The resulting algorithm for the computation of $\sigma_{\max}$ is given in Algorithm 4.1.

In step 2 of the algorithm, RMGS stands for repeated modified Gram–Schmidt (see, for example, [7, pp. 231–232]), used to make s and t orthogonal to U_{k-1} and

```
Input: a device to compute $Av$ and $A^Tu$ for arbitrary $u$ and $v$, starting vectors $u_1$ and $v_1$,
    and a tolerance $\varepsilon$.
Output: the approximate singular triple $(\theta, u, v)$ for the largest singular value
    $\sigma_{\max}$ and its corresponding singular vectors satisfying $\|\binom{Av-\theta u}{A^Tu-\theta v}\| \le \varepsilon$.

1.    $s = u_1,\ t = v_1,\ U_0 = [\ ],\ V_0 = [\ ]$
      for $k = 1, \ldots$
2.        $U_k = \mathrm{RMGS}(U_{k-1}, s)$
          $V_k = \mathrm{RMGS}(V_{k-1}, t)$
3.        Compute $k$th column of $W_k = AV_k$
          Compute $k$th row and column of $H_k = U_k^T AV_k = U_k^T W_k$
4.        Compute largest singular triple $(\theta, c, d)$ of $H_k$, $(\|c\| = \|d\| = 1)$
          $u = U_k c,\ v = V_k d$
5.        $r = \binom{Av-\theta u}{A^Tu-\theta v} = \binom{W_k d-\theta u}{A^Tu-\theta v}$
6.        Stop if $\|r\| \le \varepsilon$
7.        Solve (approximately) an $(s,t) \perp\perp (u,v)$ from
      $\begin{pmatrix} I_m - uu^T & 0 \\ 0 & I_n - vv^T \end{pmatrix} \begin{pmatrix} -\theta I_m & A \\ A^T & -\theta I_n \end{pmatrix} \begin{pmatrix} I_m - uu^T & 0 \\ 0 & I_n - vv^T \end{pmatrix} \begin{pmatrix} s \\ t \end{pmatrix} = -r$
```

ALG. 4.1. *The JDSVD algorithm for the computation of* $\sigma_{\max}(A)$ *with Galerkin conditions* $\mathcal{X} = \mathcal{U}$, $\mathcal{Y} = \mathcal{V}$. *The approximations* (θ, η, u, v) *satisfy* $\theta = \eta = u^TAv$.

V_{k-1}, and to expand the search spaces with the normalized vectors. Furthermore, [ ] stands for the empty matrix, and we omit the index k of all variables that are overwritten in every step. If we are interested in another singular value, for example, the smallest, or the one closest to a specific target, we should adjust our choice in step 4 of the algorithm accordingly. The variant of Algorithm 4.1 is the only variant of the JDSVD for which the operator in (3.6) is symmetric and maps $(u,v)^{\perp\perp}$ in itself. Other choices imply that the operator is not symmetric or maps $(u,v)^{\perp\perp}$ to a different space. See also section 7.2.

4.2. Optimality of this choice. The following two results indicate that the method resulting from this standard Galerkin choice is optimal in some sense. Suppose we have an $m \times k$ matrix U and an $n \times k$ matrix V. Then for any $k \times k$ matrices K and L there are associated an $m \times k$ residual matrix $R(K)$ and an $n \times k$ residual matrix $\widetilde{R}(L)$:

$$R(K) := AV - UK \quad \text{and} \quad \widetilde{R}(L) := A^TU - VL.$$

If there exist K and L such that these residual matrices are zero, then we have found left and right singular subspaces, i.e., invariant subspaces of A^TA and AA^T. The following theorem states that if both U and V have orthonormal columns, then $H := U^TAV$ and $H^T = V^TA^TU$ minimize the norm of these residual matrices, which is a desirable property. It is a generalization of a result in the theory for eigenproblems (see [12, Theorem 11.4.2] and [18, Theorem IV.1.15]), which deals with residuals of the form $AV - VK$.

THEOREM 4.1 (cf. Theorem 11.4.2 of [12]). *For given* $m \times k$ *matrix* U *and* $n \times k$ *matrix* V, *let* $H = U^TAV$.

(a) *If the columns of* U *are orthonormal, then for all* $k \times k$ *matrices* K *we have* $\|R(H)\| \le \|R(K)\|$. *Moreover,* H *is unique with respect to the Frobenius norm* $\|R(H)\|_F \le \|R(K)\|_F$ *with equality only when* $K = H$.

(b) *If the columns of V are orthonormal, then $H^T = V^T A^T U$ minimizes the norm of $\widetilde{R}(L)$, and H^T is unique with respect to the Frobenius norm.*

Proof. Suppose that the columns of U are orthonormal; then $U^T U = I$, so

$$\begin{aligned} R(K)^T R(K) &= V^T A^T A V + K^T K - K^T H - H^T K \\ &= V^T A^T A V - H^T H + (K-H)^T (K-H) \\ &= R(H)^T R(H) + (K-H)^T (K-H). \end{aligned}$$

Since $(K-H)^T(K-H)$ is positive semidefinite, it follows that

$$\|R(K)\|^2 = \lambda_{\max}(R(K)^T R(K)) \geq \lambda_{\max}(R(H)^T R(H)) = \|R(H)\|^2,$$

where we used Lemma 2.5 in the inequality. For uniqueness, we realize that $\|B\|_F^2 = \text{Trace}(B^T B)$ for every real matrix B. Part (b) can be proved using the same methods. □

Proposition 4.2. *Let u and v be approximations of unit norm. Then*

$$(\theta, \eta) = (u^T A v, u^T A v) \text{ minimizes } \|r(\theta, \eta)\|.$$

Proof. This can be shown by differentiating $\|r(\theta,\eta)\|^2$ with respect to θ and η. □

Because of these two results, it is a natural idea to take the k singular values $\theta_j^{(k)}$ of H_k as approximations to the singular values of A. When U_k and V_k have orthonormal columns, we see by Lemma 2.8 that these approximations converge in a finite number of steps to the singular values of A. In the following theorem we show that the approximations converge monotonically increasing.

Theorem 4.3. *Let $\theta_k^{(k)} \leq \cdots \leq \theta_1^{(k)}$ be the singular values of $H_k := U_k^T A V_k$, where U_k and V_k have orthonormal columns. Then for all fixed j and increasing k, the $\theta_j^{(k)}$ converge monotonically increasing to the σ_j.*

Proof. H_k is a submatrix of H_{k+1}, so according to Lemma 2.9 $\theta_j^{(k+1)} \geq \theta_j^{(k)}$ for $1 \leq j \leq k$. Because of the orthogonality of U_k and V_k, the $\theta_j^{(k)}$ converge to the σ_j. □

Remark 4.4. *In practice, one often observes that the $\theta_j^{(k)}$ converge strictly monotonically to the σ_j. With the aid of* [21, pp. 94–98]*, conditions could be formulated under which the convergence is strict.*

Note that the theorem does *not* say that the smallest approximations $\theta_k^{(k)}$ converge monotonically (decreasing) to σ_p, because Lemma 2.9 only gives us $\theta_{k+1}^{(k+1)} \leq \theta_{k-1}^{(k)}$. For example, if $u_k \approx u_{*,p}$ and $v_k \approx v_{*,p-1}$, then $\theta_k^{(k)} \approx 0$, so we see that the smallest approximation can in fact be (much) smaller than σ_p. Experiments show that the convergence of the $\theta_k^{(k)}$ can be irregular and slow (see section 8). This is a serious difficulty of working with the augmented matrix, because the smallest singular values are in the interior of its spectrum. We discuss this matter further in sections 4.3, 4.4, and 7.5. The following theorem gives some relations between the singular values of H_k and those of A. For clarity, we leave out the index k as much as possible.

Theorem 4.5 (cf. Theorems 11.5.1 and 11.5.2 of [12] and Corollary IV.4.15 of [18]). *For $j = 1, \ldots, k$, there exist singular values $\sigma_{j'}$ of A which can be put in one-one correspondence with the singular values θ_j of H in such a way that*

$$|\sigma_{j'} - \theta_j| \leq \max\{\|R(H)\|, \|\widetilde{R}(H^T)\|\} \qquad (1 \leq j \leq k).$$

Moreover,

$$\sum_{j=1}^{k} (\sigma_{j'} - \theta_j)^2 \le \|R(H)\|_F^2 + \|\widetilde{R}(H^T)\|_F^2.$$

Proof. Let the columns of $\widetilde{U}$ and $\widetilde{V}$ be orthonormal bases for the orthogonal complements of U and V, respectively. Then both $[U\ \widetilde{U}]$ and $[V\ \widetilde{V}]$ are orthogonal and

$$[U\ \widetilde{U}]^T A [V\ \widetilde{V}] = \begin{pmatrix} H & 0 \\ 0 & \widetilde{U}^T A \widetilde{V} \end{pmatrix} + \begin{pmatrix} 0 & U^T A \widetilde{V} \\ \widetilde{U}^T A V & 0 \end{pmatrix}. \tag{4.3}$$

Using Lemmas 2.8 and 2.6, respectively, we obtain for $1 \le j \le p = \min\{m, n\}$

$$\sigma_j(A) = \sigma_j\left([U\ \widetilde{U}]^T A [V\ \widetilde{V}]\right) \le \sigma_j \begin{pmatrix} H & 0 \\ 0 & \widetilde{U}^T A \widetilde{V} \end{pmatrix} + \sigma_{\max} \begin{pmatrix} 0 & U^T A \widetilde{V} \\ \widetilde{U}^T A V & 0 \end{pmatrix}.$$

Now

$$[U\ \widetilde{U}]^T R(H) = \begin{pmatrix} 0 \\ \widetilde{U}^T A V \end{pmatrix} \qquad \text{and} \qquad [V\ \widetilde{V}]^T \widetilde{R}(H^T) = \begin{pmatrix} 0 \\ \widetilde{V}^T A^T U \end{pmatrix},$$

so, because of the orthogonal invariance of the norm (see Lemma 2.8), $\|R(H)\| = \|\widetilde{U}^T A V\|$ and $\|\widetilde{R}(H^T)\| = \|\widetilde{V}^T A^T U\| = \|U^T A \widetilde{V}\|$. Because

$$\Sigma \begin{pmatrix} H & 0 \\ 0 & \widetilde{U}^T A \widetilde{V} \end{pmatrix} = \Sigma(H) \ \cup \ \Sigma(\widetilde{U}^T A \widetilde{V}),$$

there exist indices j' such that

$$\sigma_{j'} \begin{pmatrix} H & 0 \\ 0 & \widetilde{U}^T A \widetilde{V} \end{pmatrix} = \theta_j.$$

So the theorem's first inequality is obtained by

$$\sigma_{\max} \begin{pmatrix} 0 & U^T A \widetilde{V} \\ \widetilde{U}^T A V & 0 \end{pmatrix} = \max\ \{\|\widetilde{U}^T A V\|, \|U^T A \widetilde{V}\|\} = \max\ \{\|R(H)\|, \|\widetilde{R}(H^T)\|\}.$$

For the second inequality, apply Lemma 2.7 to the splitting of (4.3). □

For the following proposition, we need the minimax theorem for singular values [9, Theorem 3.1.2]

$$\sigma_j = \max_{\mathcal{X}^j \subset \mathbb{R}^n} \ \min_{0 \ne x \in \mathcal{X}} \frac{\|Ax\|}{\|x\|}, \tag{4.4}$$

where $\mathcal{X}^j$ ranges over all subspaces of $\mathbb{R}^n$ of dimension j.

The following proposition states that the singular values of $U_k^T A V_k$ are also *not* optimal in another sense.

Proposition 4.6. *Let U_k and V_k have orthonormal columns. For $1 \le j \le k$,*

$$\sigma_j(U_k^T A V_k) \le \sigma_j(A V_k) \qquad \textit{and} \qquad \sigma_j(U_k^T A V_k) \le \sigma_j(A^T U_k).$$

Proof. This follows from (4.4) and the inequalities $\|U_k^T A V_k y\| \le \|A V_k y\|$ and $\|V_k^T A^T U_k x\| \le \|A^T U_k x\|$. □

We have seen that the $\sigma_j(U_k^T A V_k)$ increase monotonically and that they are bounded above by both $\sigma_j(AV_k) = \lambda_j^{1/2}(V_k^T A^T A V_k)$ and $\sigma_j(A^T U_k) = \lambda_j^{1/2}(U_k^T A A^T U_k)$. This forms one motivation to study other Galerkin choices. A second is the possibly irregular convergence of the smallest singular value of $U_k^T A V_k$.

4.3. Other choices. Suppose that the columns of V form an orthonormal basis for $\mathcal{V}$. By the Galerkin choice $\mathcal{X} = A\mathcal{V}$, $\mathcal{Y} = \mathcal{V}$, with test vectors $x = Av$, $y = v$, and $u = Uc$, $v = Vd$, and $\|v\| = 1$, (3.2) reduces to

$$\left\{ \begin{array}{lcl} V^T A^T A V d & = & \theta\, V^T A^T U c, \\ V^T A^T U c & = & \eta\, d. \end{array} \right. \tag{4.5}$$

One can check that to satisfy the Galerkin conditions, $(\theta\eta, d)$ should be an eigenpair of $V^T A^T A V$. Now first suppose that $V^T A^T U$ is nonsingular. Note that in this case $\eta \neq 0$; otherwise, $V^T A^T U$ would be singular. It follows that $c = \eta (V^T A^T U)^{-1} d$, $\eta = v^T A^T u$, and $\theta = v^T A^T A v / v^T A u$. When $V^T A^T U$ is singular, then this construction is impossible, but in this case we can simply restart the process or add extra vectors to the search spaces (see section 7.6).

With this Galerkin choice, θ and η do not converge monotonically in general, but we can apply well-known results from eigenvalue theory to ensure that their product does converge monotonically to the squares of the singular values and also to the smallest. In section 7.2 we discuss the resulting correction equation.

Likewise, if the columns of U form an orthonormal basis for $\mathcal{U}$, the Galerkin choice $\mathcal{X} = \mathcal{U}$, $\mathcal{Y} = A^T\mathcal{U}$ leads to the determination of $(\theta\eta, c)$, an eigenpair of $U^T A A^T U$. These two approaches are natural with respect to minimax considerations, as we will see now.

LEMMA 4.7. *Let $\xi \in [0,1]$. Then we have the following minimax property for singular values:*

$$\sigma_j = \max_{\substack{\mathcal{S}^j \subset \mathbb{R}^m \\ \mathcal{T}^j \subset \mathbb{R}^n}} \min_{\substack{0 \neq s \in \mathcal{S}^j \\ 0 \neq t \in \mathcal{T}^j}} \xi \frac{\|At\|}{\|t\|} + (1-\xi) \frac{\|A^T s\|}{\|s\|} \qquad (1 \le j \le p). \tag{4.6}$$

Proof. This follows from (4.4) and the observation that A and A^T have the same singular values. □

When we have search spaces $\mathcal{U}$ and $\mathcal{V}$, it is a natural idea to substitute $\mathcal{U}$ for $\mathbb{R}^m$ and $\mathcal{V}$ for $\mathbb{R}^n$ in (4.6), as a generalization of a similar idea in the theory of eigenproblems; see [12, p. 236]. This gives the following approximations for the singular values:

$$\tau_j = \max_{\substack{\mathcal{S}^j \subset \mathcal{U} \\ \mathcal{T}^j \subset \mathcal{V}}} \min_{\substack{0 \neq s \in \mathcal{S}^j \\ 0 \neq t \in \mathcal{T}^j}} \xi \frac{\|At\|}{\|t\|} + (1-\xi) \frac{\|A^T s\|}{\|s\|}. \tag{4.7}$$

The following theorem relates these approximations to the Ritz values of $A^T A$ and AA^T.

THEOREM 4.8. $\tau_j = \xi(\lambda_j^{1/2}(V^T A^T A V)) + (1-\xi)(\lambda_j^{1/2}(U^T A A^T U))$.

Proof. We have that $\mathcal{T}^j \subset \mathcal{V}$ if and only if $\mathcal{T}^j = V\widetilde{\mathcal{T}}^j := \{Vt : t \in \widetilde{\mathcal{T}}^j\}$ and $\widetilde{\mathcal{T}}^j \subset \mathbb{R}^k$. So for the first term of the expression for the τ_j we have that

$$\max_{\mathcal{T}^j \subset \mathcal{V}} \min_{0 \neq t \in \mathcal{T}^j} \frac{\|At\|^2}{\|t\|^2} = \max_{\widetilde{\mathcal{T}}^j \subset \mathbb{R}^k} \min_{0 \neq t \in \widetilde{\mathcal{T}}^j} \frac{t^T V^T A^T A V t}{\|t\|^2} = \lambda_j(V^T A^T A V).$$

For the second term we have a similar expression. □

When we take $\xi = 0$ and $\xi = 1$ in Theorem 4.8, we recognize the Galerkin approaches described in (4.5) and the discussion after that. They can essentially be viewed as a two-sided approach to A^TA or AA^T, in the sense that we have approximations to both the left and the right singular vector during the process.

As a generalization, we can consider the test spaces $\mathcal{X}$ and $\mathcal{Y}$ with bases $\alpha u_i+\beta Av_i$ and $\gamma v_i + \delta A^Tu_i$, respectively, where $\alpha^2 + \beta^2 = \gamma^2 + \delta^2 = 1$. Every choice other than $\alpha = \gamma = 1$ (the standard Galerkin choice discussed in section 4.1) involves the computation of additional projected matrices and more work per iteration.

Another possibility is to take search spaces of unequal dimension, that is, $\mathcal{U}_k$ and $\mathcal{V}_l$, where $k \neq l$. However, in view of the symmetric role of $\mathcal{S}_j$ and $\mathcal{T}_j$ in (4.6), this is probably not very useful.

4.4. Harmonic singular triples. As observed in section 4.2, the standard Galerkin choice leads to monotone convergence for the largest singular value, but it can imply irregular behavior for the smallest singular value. A related problem is the selection of the best approximate vectors. Suppose that $u = \sum_{j=1}^m \gamma_j u_{*,j}$ and $v = \sum_{j=1}^n \delta_j v_{*,j}$ are approximate vectors; then $\theta = u^TAv = \sum_{j=1}^p \gamma_j\delta_j\sigma_j$. (We may assume θ is nonnegative; otherwise, take $-u$ instead of u.) Now suppose that $\theta \approx \sigma_1$, in the sense that $\sigma_2 < \theta < \sigma_1$ and that $\sigma_1 - \theta$ is (much) smaller than $\theta - \sigma_2$. Then we conclude that $\gamma_1 \approx 1$ and $\delta_1 \approx 1$, so u and v are good approximations to $u_{*,1}$ and $v_{*,1}$. But when $\theta \approx \sigma_p$, u and v are not necessarily good approximations to $u_{*,p}$ and $v_{*,p}$. For example, u could have a large component of $u_{*,p-1}$ and a small component of $u_{*,1}$, and v could have a large component of $v_{*,p-2}$ and a small component of $v_{*,1}$. In conclusion, when we search for the largest singular value, it is asymptotically safe to select the largest singular triple of H, but for the smallest singular value the situation is more subtle.

Suppose for the moment that A is square and invertible. If the minimal singular value is the one of interest, the above discussion suggests to study the singular values of A^{-1}. Based on the SVD of A^{-1}

$$A^{-1} = V_*\Sigma^{-1}U_*^T,$$

we try to find the largest singular values of A^{-1} with respect to certain search spaces $\widehat{\mathcal{U}}, \widehat{\mathcal{V}}$ and test spaces $\widehat{\mathcal{X}}, \widehat{\mathcal{Y}}$. The new Galerkin conditions become (cf. (3.1))

$$\begin{pmatrix} A^{-1}\widehat{u} - \widehat{\theta}\widehat{v} \\ A^{-T}\widehat{v} - \widehat{\eta}\widehat{u} \end{pmatrix} \perp\perp \begin{pmatrix} \widehat{\mathcal{Y}} \\ \widehat{\mathcal{X}} \end{pmatrix},$$

where $\widehat{u} \in \widehat{\mathcal{U}}$ and $\widehat{v} \in \widehat{\mathcal{V}}$, say, $\widehat{u} = \widehat{U}\widehat{c}$ and $\widehat{v} = \widehat{V}\widehat{d}$. To avoid having to work with A^{-1}, we take for the search spaces $\widehat{\mathcal{U}} = A\mathcal{V}$ and $\widehat{\mathcal{V}} = A^T\mathcal{U}$ (cf. [15]). This gives the system

$$\left\{ \begin{array}{rcl} \widehat{Y}^TV\widehat{c} & = & \widehat{\theta}\,\widehat{Y}^TA^TU\widehat{d}, \\ \widehat{X}^TU\widehat{d} & = & \widehat{\eta}\,\widehat{X}^TAV\widehat{c}. \end{array} \right.$$

Taking $\widehat{\mathcal{X}} = \mathcal{U}$ and $\widehat{\mathcal{Y}} = \mathcal{V}$ results in equivalent conditions as in the standard choice (4.1); only now $(\widehat{\eta}, \widehat{\theta})$ and $(\widehat{c}, \widehat{d})$ play the role of (θ^{-1}, η^{-1}) and (d, c). The choices $(\widehat{\mathcal{X}}, \widehat{\mathcal{Y}}) = (A\mathcal{V}, \mathcal{V})$, $(\mathcal{U}, A^T\mathcal{U})$, and $(A\mathcal{V}, A^T\mathcal{U})$ lead to different approximations: to (4.5) and other systems described in section 4.3, only the roles of the variables differ. We can call these approximations *harmonic singular triples*, in analogy to the harmonic

Ritz pairs in the eigenproblem [11]. So these harmonic approximations have two advantages: the monotone behavior of the approximations to the smallest singular value, and the selection of a good approximate "smallest" vector.

The conclusion is that the nondefault Galerkin choices, as presented in section 4.3, can also be seen as a "harmonic" approach to the problem. Finally, when A is singular or even nonsquare, we can consider A^+ with respect to the test and search spaces $A\mathcal{V}$ and $A^T\mathcal{U}$, exploiting the fact that $AA^+A = A$.

5. The JDSVD as an (inexact) accelerated Newton scheme. In [16], it is shown that the Jacobi–Davidson method can be interpreted as an inexact accelerated Newton scheme [4] for the eigenvalue problem. Here we show that the same is true for the JDSVD applied to the singular value problem. Define $F : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^m \times \mathbb{R}^n$ as

$$F(u, v) := \begin{pmatrix} Av - \theta u \\ A^T u - \eta v \end{pmatrix},$$

where $\theta = \theta(u, v)$ and $\eta = \eta(u, v)$ are as in (3.3). Thus the function F is nonlinear. Consider the singular value problem where we require the singular vectors u_*, v_* to be scaled such that $u_*^T a = 1$ and $v_*^T b = 1$ for certain vectors $a \in \mathbb{R}^m$ and $b \in \mathbb{R}^n$. So we look for solutions u_*, v_* of the equation $F(u, v) = 0$ in the "hyperplane"

$$\left\{(u, v) \in \mathbb{R}^m \times \mathbb{R}^n : u^T a = 1,\ v^T b = 1\right\}.$$

We introduce these a and b to derive a more general form of the correction equation (3.6). If (u_k, v_k) are approximations to the singular vectors, then the next Newton approximations (u_{k+1}, v_{k+1}) are given by $(u_{k+1}, v_{k+1}) = (u_k, v_k) + (s_k, t_k)$, where $(s_k, t_k) \perp\perp (a, b)$ satisfies

$$DF(u_k, v_k)(s_k, t_k) = -F(u_k, v_k) = -r_k.$$

Omitting the index k, one may check (remembering that $\theta = \theta(u, v)$ and $\eta = \eta(u, v)$ are as in (3.3)) that the Jacobian $DF(u, v)$ of F is given by

$$DF(u, v) = \begin{pmatrix} I_m - \frac{ux^T}{x^T u} & 0 \\ 0 & I_n - \frac{vy^T}{y^T v} \end{pmatrix} \begin{pmatrix} -\theta I_m & A \\ A^T & -\eta I_n \end{pmatrix}.$$

Hence the correction equation of the Newton step is given by

$$\begin{pmatrix} I_m - \frac{ux^T}{x^T u} & 0 \\ 0 & I_n - \frac{vy^T}{y^T v} \end{pmatrix} \begin{pmatrix} -\theta I_m & A \\ A^T & -\eta I_n \end{pmatrix} \begin{pmatrix} s \\ t \end{pmatrix} = -r, \qquad \text{where } (s, t) \perp\perp (a, b).$$

For every $\widetilde{x}, \widetilde{y}$ so that $a^T \widetilde{x} \neq 0$ and $b^T \widetilde{y} \neq 0$, this is equivalent to the slightly more general form of the JDSVD correction equation (in comparison with (3.6)),

$$\begin{pmatrix} I_m - \frac{ux^T}{x^T u} & 0 \\ 0 & I_n - \frac{vy^T}{y^T v} \end{pmatrix} \begin{pmatrix} -\theta I_m & A \\ A^T & -\eta I_n \end{pmatrix} \begin{pmatrix} I_m - \frac{\widetilde{x} a^T}{a^T \widetilde{x}} & 0 \\ 0 & I_n - \frac{\widetilde{y} b^T}{b^T \widetilde{y}} \end{pmatrix} \begin{pmatrix} s \\ t \end{pmatrix} = -r, \tag{5.1}$$

where $(s, t) \perp\perp (a, b)$. Note that the substitution $a = u$ and $b = v$ gives (3.6).

If we keep $a, b, x, \widetilde{x}, y$, and $\widetilde{y}$ fixed during the process, and if $x^T u_*, y^T v_*, a^T \widetilde{x}$, and $b^T \widetilde{y}$ are nonzero, then Newton iteration produces a series (u_k, v_k) that converges asymptotically quadratically towards (u_*, v_*) if the starting vector (u_1, v_1) is sufficiently close to (u_*, v_*).

But if we take $a, b, x, \widetilde{x}, y$, and $\widetilde{y}$ variable but converging to certain vectors, such that the denominators in (5.1) do not vanish, we get asymptotically quadratic convergence as well. The choice $a = x = \widetilde{x} = u_k$ and $b = y = \widetilde{y} = v_k$ leads to Algorithm 4.1. With other Galerkin choices described in section 4, the test vectors (x, y) are, in general, not equal to the approximations (u, v), and in this situation the vectors a and b can be useful; see sections 6 and 7.2.

We see that the JDSVD is a Newton scheme, accelerated by the usage of all previous iterates and the projection of A on the subspace that they span. This *subspace acceleration* accelerates the "prequadratic" phase of the method and ensures that we find a singular triple in a finite number of steps. It may be expensive to solve the correction equation exactly. Instead we may solve (5.1) approximately (see section 7.1); the resulting method is an inexact accelerated Newton scheme.

In [14], it is proved that if the correction equation is solved exactly, then Jacobi–Davidson applied to a symmetric matrix has asymptotically cubic convergence. Because the augmented matrix (1.1) is symmetric, we expect that the JDSVD can also reach cubic convergence. The next section shows that this expectation is correct indeed.

6. Convergence. In the previous section we have already seen that the correction equation represents a Jacobian system in a Newton step. Now we focus on the convergence (see [14] for similar observations for Jacobi–Davidson applied to the eigenvalue problem).

In a mathematical sense, it is not meaningful to speak of asymptotic convergence, because we know that the process ends in a finite number of steps. However, in many practical situations a singular triple will be approximated well, long before the dimension of the search spaces reaches p. At that stage, these approximations display a behavior that looks like a converging infinite sequence close to its limit. When speaking of asymptotic convergence, we think of this situation. In other words, by the word "asymptotically" we mean the situation where we have a (very) good approximation to the singular triple, rather than the situation where $k \to \infty$.

In the correction equation (5.1), u and v are the current approximations and x and y are test vectors, but we have not said much about choosing $\widetilde{x}$, $\widetilde{y}$, a, and b. They can vary per step. The next lemma and proposition show that the exact JDSVD (that is, the JDSVD where we solve the correction equation exactly) has asymptotically cubic convergence for specific choices of the test vectors x and y and the vectors a and b. To be precise, with ε small enough, if

$$|\angle(u_k, u_*)| = \mathcal{O}(\varepsilon) \text{ and } |\angle(v_k, v_*)| = \mathcal{O}(\varepsilon) \tag{6.1}$$

and if

$$a = x \text{ and } b = y \qquad \text{and} \qquad |\angle(x, u_*)| = \mathcal{O}(\varepsilon) \text{ and } |\angle(y, v_*)| = \mathcal{O}(\varepsilon), \tag{6.2}$$

then $|\angle(u_{k+1}, u_*)| = \mathcal{O}(\varepsilon^3)$ and $|\angle(v_{k+1}, v_*)| = \mathcal{O}(\varepsilon^3)$. Then the approximate singular values (see (3.3)) converge cubically as well.

LEMMA 6.1 (cf. Lemma 3.1 of [14]). *Assume that $Av_* = \sigma u_*$ and $A^T u_* = \tau v_*$, where $\sigma, \tau > 0$, and that $\sqrt{\sigma\tau}$ is a simple singular value of A. Let a, b, x, $\widetilde{x}$, y, and*

$\widetilde{y}$ be such that x^Tu_*, y^Tv_*, $a^T\widetilde{x}$, $b^T\widetilde{y}$, a^Tu_*, and b^Tv_* are all nonzero. Then the map

$$G := \begin{pmatrix} I_m - \frac{u_*x^T}{x^Tu_*} & 0 \\ 0 & I_n - \frac{v_*y^T}{y^Tv_*} \end{pmatrix} \begin{pmatrix} -\sigma I_m & A \\ A^T & -\tau I_n \end{pmatrix} \begin{pmatrix} I_m - \frac{\widetilde{x}a^T}{a^T\widetilde{x}} & 0 \\ 0 & I_n - \frac{\widetilde{y}b^T}{b^T\widetilde{y}} \end{pmatrix}$$

is a bijection from $(a,b)^{\perp\perp}$ onto $(x,y)^{\perp\perp}$.

Proof. Suppose $(z_1,z_2) \perp\perp (a,b)$ and $G(z_1,z_2)=0$. We show that $z_1=z_2=0$. We have

$$\begin{pmatrix} -\sigma I_m & A \\ A^T & -\tau I_n \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} \mu u_* \\ \nu v_* \end{pmatrix}$$

for certain μ,ν. Then

$$\begin{cases} Az_2 &= \sigma z_1 + \mu u_*, \\ A^Tz_1 &= \tau z_2 + \nu v_*. \end{cases}$$

Multiplying the first equation by A^T and the second by A, we find

$$\begin{cases} (A^TA - \sigma\tau)z_2 &= (\sigma\nu + \tau\mu)v_*, \\ (AA^T - \sigma\tau)z_1 &= (\sigma\nu + \tau\mu)u_*. \end{cases}$$

So both z_1 and u_* belong to the kernel of $(AA^T-\sigma\tau)^2$, and both z_2 and v_* belong to the kernel of $(A^TA-\sigma\tau)^2$. From the simplicity of $\sigma\tau$ using Lemma 2.4, we have that z_1 and z_2 are multiples of u_* and v_*, respectively. Because $z_1 \perp a$, $z_2 \perp b$, and $a^Tu_* \neq 0$, $b^Tv_* \neq 0$, we conclude $z_1=z_2=0$. The bijectivity follows from comparing dimensions. □

Proposition 6.2 (cf. Theorem 3.2 of [14]). *With the assumptions of Lemma 6.1, if the initial vectors are close enough to the singular vectors corresponding to a simple nonzero singular value (i.e., if (6.1) holds), and if the correction equation is solved exactly, then for fixed vectors x, y, a, and b, the JDSVD process has quadratic convergence. Moreover, if (6.2) holds, then the JDSVD has even cubic convergence.*

Proof. For convenience write

$$P = \begin{pmatrix} I_m - \frac{ux^T}{x^Tu} & 0 \\ 0 & I_n - \frac{vy^T}{y^Tv} \end{pmatrix}, \ B = \begin{pmatrix} -\theta I_m & A \\ A^T & -\eta I_n \end{pmatrix}, \ Q = \begin{pmatrix} I_m - \frac{\widetilde{x}a^T}{a^T\widetilde{x}} & 0 \\ 0 & I_n - \frac{\widetilde{y}b^T}{b^T\widetilde{y}} \end{pmatrix}.$$

Then the correction equation (5.1) reads, for $(s,t) \perp\perp (a,b)$,

$$PBQ(s,t) = PB(s,t) = -r = -B(u,v).$$

Suppose that $\widetilde{u}$ and $\widetilde{v}$ are scalar multiples of the singular vectors u_* and v_* and that $(\widetilde{u},\widetilde{v}) = (u,v) + (e,f)$, where $(e,f) \perp\perp (a,b)$, and $\|e\| = \mathcal{O}(\varepsilon)$, $\|f\| = \mathcal{O}(\varepsilon)$. Our first goal is to show that $\|(e-s,f-t)\| = \mathcal{O}(\varepsilon^2)$. We know that there are $\sigma,\tau > 0$ such that

$$0 = \begin{pmatrix} -\sigma I_m & A \\ A^T & -\tau I_n \end{pmatrix} \begin{pmatrix} \widetilde{u} \\ \widetilde{v} \end{pmatrix} = \begin{pmatrix} -\theta I_m & A \\ A^T & -\eta I_n \end{pmatrix} \begin{pmatrix} \widetilde{u} \\ \widetilde{v} \end{pmatrix} - \begin{pmatrix} (\sigma-\theta)\widetilde{u} \\ (\tau-\eta)\widetilde{v} \end{pmatrix}.$$

Therefore, we have

$$\begin{pmatrix} -\theta I_m & A \\ A^T & -\eta I_n \end{pmatrix} \begin{pmatrix} e \\ f \end{pmatrix} = -\begin{pmatrix} -\theta I_m & A \\ A^T & -\eta I_n \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} + \begin{pmatrix} (\sigma-\theta)\widetilde{u} \\ (\tau-\eta)\widetilde{v} \end{pmatrix}. \tag{6.3}$$

We multiply this on the left side by P and use the fact that $PB(u, v) = B(u, v)$:

$$PB(e, f) = -B(u, v) + P((\sigma - \theta)\widetilde{u}, (\tau - \eta)\widetilde{v}). \tag{6.4}$$

Subtracting $PB(s, t) = -B(u, v)$ from (6.4), and noting that $P(u, v) = 0$, we get

$$PB(e - s, f - t) = P((\sigma - \theta)e, (\tau - \eta)f). \tag{6.5}$$

Multiplying (6.3) on the left by $\left(\begin{smallmatrix} x & 0 \\ 0 & y \end{smallmatrix}\right)^T$ leads to

$$\begin{pmatrix} \sigma - \theta \\ \tau - \eta \end{pmatrix} = \begin{pmatrix} (x^T\widetilde{u})^{-1} & 0 \\ 0 & (y^T\widetilde{v})^{-1} \end{pmatrix} \begin{pmatrix} -\theta x^T & x^T A \\ y^T A^T & -\eta y^T \end{pmatrix} \begin{pmatrix} e \\ f \end{pmatrix}. \tag{6.6}$$

So for fixed x, y, a, and b we have $\|PB(e - s, f - t)\| = \mathcal{O}(\varepsilon^2)$. Using Lemma 6.1 and the assumption that the initial vectors are close enough to the singular vectors, we see that PB in (6.5) is invertible, so $\|(e - s, f - t)\| = \mathcal{O}(\varepsilon^2)$, which implies quadratic convergence. But, if additionally, (6.2) holds, then

$$\left\| \begin{pmatrix} -\theta x^T & x^T A \\ y^T A^T & -\eta y^T \end{pmatrix} \begin{pmatrix} e \\ f \end{pmatrix} \right\| = \left\| \begin{pmatrix} a^T A f \\ b^T A^T e \end{pmatrix} \right\| = \sigma \left\| \begin{pmatrix} b^T f \\ a^T e \end{pmatrix} \right\| + \mathcal{O}(\varepsilon^2) = \mathcal{O}(\varepsilon^2),$$

so from (6.6) we see that $\|(\sigma - \theta, \tau - \eta)\| = \mathcal{O}(\varepsilon^2)$. We conclude that in this case the convergence is even cubic. □

One may check that the hypotheses on x and y in the theorem are true if we choose $x_k = u_k$ or $x_k = Av_k$, and $y_k = v_k$ or $y_k = A^T u_k$ in the process. The cubic convergence can be observed in practice; see section 8.

7. Various aspects of the method.

7.1. Solving the correction equation.

We now translate a number of observations for Jacobi–Davidson in [15, 14] to the JDSVD context. Consider the situation after k steps of the JDSVD algorithm. For easy reading, we again leave out the index k. In this section we take for simplicity the Galerkin spaces used in section 4.1, but most arguments carry over to other choices. First we rewrite the correction equation. Because of $(s, t) \perp\perp (u, v)$, we can eliminate the projections and write (3.6) as

$$\begin{pmatrix} -\theta I_m & A \\ A^T & -\theta I_n \end{pmatrix} \begin{pmatrix} s \\ t \end{pmatrix} = -r + \begin{pmatrix} \alpha u \\ \beta v \end{pmatrix}, \tag{7.1}$$

where α and β are determined by the requirement that $(s, t) \perp\perp (u, v)$. If we have a nonsingular preconditioner $M \approx \left(\begin{smallmatrix} -\theta I_m & A \\ A^T & -\theta I_n \end{smallmatrix}\right)$, then we can take an approximation

$$(\widetilde{s}, \widetilde{t}) = -M^{-1} r + M^{-1}(\alpha u, \beta v). \tag{7.2}$$

1 (cf. [15, p. 406, point 1]). If we approximate (s, t) simply by $\pm r$ (by taking $M = \mp I$ and $\alpha = \beta = 0$), then, because of the orthogonalization at step 2 of Algorithm 4.1, this is equivalent to taking $(\widetilde{s}, \widetilde{t}) = (Av, A^T u)$. By induction one can prove that for the special case where we take this simple approximation in every step, we have

$$\mathcal{U}_{2k} = \mathcal{K}_k(AA^T, u_1) \oplus \mathcal{K}_k(AA^T, Av_1), \qquad \mathcal{V}_{2k} = \mathcal{K}_k(A^T A, v_1) \oplus \mathcal{K}_k(A^T A, A^T u_1),$$

as long as the Krylov subspaces have a trivial intersection. Compare this with bidiagonalization, where

$$\mathcal{U}_k = \mathcal{K}_k(AA^T, Av_1), \qquad \mathcal{V}_k = \mathcal{K}_k(A^T A, v_1).$$

2 (cf. [15, p. 408, point 3]). If θ is not equal to a singular value, then $M = \left(\begin{smallmatrix} -\theta I_m & A \\ A^T & -\theta I_n \end{smallmatrix}\right)$ is nonsingular and $M^{-1}r = (u, v)$. So for the updated vectors $\widetilde{u}, \widetilde{v}$ we have

$$\begin{pmatrix} \widetilde{u} \\ \widetilde{v} \end{pmatrix} = \begin{pmatrix} u+s \\ v+t \end{pmatrix} = \begin{pmatrix} -\theta I_m & A \\ A^T & -\theta I_n \end{pmatrix}^{-1} \begin{pmatrix} \alpha u \\ \beta v \end{pmatrix}. \tag{7.3}$$

We conclude that exact the JDSVD can be seen as an accelerated scaled RQI.

3 (cf. [15, p. 409, point 4]). If we take $M \neq \left(\begin{smallmatrix} -\theta I_m & A \\ A^T & -\theta I_n \end{smallmatrix}\right)$, M nonsingular, then with $(\widetilde{s}, \widetilde{t}) = M^{-1}(\alpha u, \beta v)$ we obtain an inexact shift and invert method. This may be an attractive alternative if (7.3) is expensive.

4. When we are interested in a singular value close to a specific *target* τ, we can replace this in the left-hand side of the correction equation (3.6):

$$\begin{pmatrix} I_m - uu^T & 0 \\ 0 & I_n - vv^T \end{pmatrix} \begin{pmatrix} -\tau I_m & A \\ A^T & -\tau I_n \end{pmatrix} \begin{pmatrix} I_m - uu^T & 0 \\ 0 & I_n - vv^T \end{pmatrix} \begin{pmatrix} s \\ t \end{pmatrix} = -r.$$

The advantage of this approach is that we avoid misconvergence to some unwanted singular value "on the way." For example, if we want to compute the largest singular value, we can use a known approximation of $\sigma_{\max}$ as a target. In practice, $\tau \approx \|A\|_\infty$ may be a good guess (see section 8). For the minimal singular value, we can take $\tau = 0$ or a small positive number as target. As soon as we notice that the process starts to converge, we may replace the target in the correction equation by the current approximation to the singular value again.

5. In practice we often solve (5.1) approximately by an iterative method: for example, a few steps of GMRES or MINRES if the operator is symmetric (in case of the standard Galerkin choice). We may use a (projected) preconditioner; see section 7.8.

7.2. The correction equation with nonstandard Galerkin choices. In the case of nonstandard Galerkin choices (see section 4.3), we may have the situation that $(x, y) \neq (u, v)$. Now we exploit the flexibility of (a, b) in (5.1): by the choice

$$(a, b) = (x, y) \text{ and } (\widetilde{x}, \widetilde{y}) = (u, v), \tag{7.4}$$

we ensure that the operator in (5.1) maps $(x, y)^{\perp\perp}$ onto itself, and that the asymptotic convergence is cubic according to Theorem 6.2 (if the correction equation is solved exactly). Another option is

$$(a, b) = (u, v) \text{ and } (\widetilde{x}, \widetilde{y}) = (x, y) \tag{7.5}$$

to make the operator in (5.1) symmetric. In this case the operator maps $(u, v)^{\perp\perp}$ to $(x, y)^{\perp\perp}$. Therefore, we should use a left "preconditioner" that maps the image space $(x, y)^{\perp\perp}$ bijectively onto the domain space $(u, v)^{\perp\perp}$ (see also section 8 and [14, 17]).

7.3. Comparison with Jacobi–Davidson on the augmented matrix. It is interesting to compare the JDSVD with Jacobi–Davidson on the augmented matrix, starting with the "same" starting vector $w_1 = (u_1, v_1)/\sqrt{2}$.

There are some analogies between Jacobi–Davidson and the JDSVD. When their correction equations are solved exactly, both converge asymptotically cubically to a simple eigenvalue of the augmented matrix. Moreover, the costs per iteration are almost the same; the only difference is that in each step the JDSVD needs a small SVD, while Jacobi–Davidson needs a small eigenvalue decomposition. The storage requirements are also comparable.

The main difference is the fact that the JDSVD, by construction, searches in two (smaller) subspaces, while Jacobi–Davidson has one search space. If Jacobi–Davidson solves its correction equation exactly, then in fact it solves (7.3) with $\alpha = \beta$ [15]. This suggests that the JDSVD can cope better with "unbalanced" vectors, that is, vectors (u, v), where $\|u\| \neq \|v\|$. An extreme example of this can be seen by taking a starting vector of the form $(u_*, \delta v_*)$ for $0 < \delta < 1$. In contrast to Jacobi–Davidson, the JDSVD terminates after computing a zero residual.

Another (mostly theoretical) difference is the fact that the JDSVD terminates for every starting vector after at most $\max\{m, n\}$ iterations, and Jacobi–Davidson terminates on the augmented matrix after at most $m + n$ iterations. In section 8, we compare the methods experimentally.

7.4. Refinement procedure. Suppose that we have found an approximate minimal right singular vector $v = (1 - \varepsilon^2)^{1/2} v_{\min} + \varepsilon v_{\max}$ by an iterative method applied to $A^T A$, so that $\sin \angle(v, v_{\min}) = \varepsilon$. Then, in the absence of other information, $u = Av = (1 - \varepsilon^2)^{1/2} \sigma_{\min} u_{\min} + \varepsilon \sigma_{\max} u_{\max}$ is the best approximation to the left singular vector we have to our disposal. But $\tan \angle(u, u_{\min}) \approx \varepsilon \frac{\sigma_{\max}}{\sigma_{\min}} = \kappa(A)\varepsilon$, and this can be large. Moreover, $\|u\|^2 = (1 - \varepsilon^2)\sigma_{\min}^2 + \varepsilon^2 \sigma_{\max}^2$ can be an inaccurate approximation to $\sigma_{\min}^2$, and so may $\|A^T u\|^2 / \|u\|^2$.

Hence the approximations to small singular values, resulting from working with $A^T A$, may be inaccurate. In this situation, we may try to improve the approximate singular triple by a two-sided approach like the JDSVD. The following lemma gives a link with [3], where a system with a matrix of the form

$$\begin{pmatrix} -\theta I_m & A & -u & 0 \\ A^T & -\theta I_n & 0 & -v \\ 2u^T & 0 & 0 & 0 \\ 0 & 2v^T & 0 & 0 \end{pmatrix} \tag{7.6}$$

is used for improving an approximate singular triple.

Lemma 7.1 (cf. Theorem 3.5 of [14]). *The JDSVD correction equation* (5.1) *is equivalent to*

$$\begin{pmatrix} -\theta I_m & A & -u & 0 \\ A^T & -\eta I_n & 0 & -v \\ a^T & 0 & 0 & 0 \\ 0 & b^T & 0 & 0 \end{pmatrix} \begin{pmatrix} s \\ t \\ \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \theta u - Av \\ \eta v - A^T u \\ 0 \\ 0 \end{pmatrix}; \tag{7.7}$$

that is, if (s, t, α, β) *is a solution of* (7.7)*, then* (s, t) *is a solution of the correction equation* (5.1)*, and if* (s, t) *is a solution of* (5.1)*, then there exist unique* α, β *such that* (s, t, α, β) *is a solution of* (7.7)*.*

Proof. We use the same notation as in the proof of Theorem 6.2. The system (7.7) is equivalent to

$$B(s, t) - (\alpha u, \beta v) = -r \qquad \text{and} \qquad (s, t) \perp\!\!\perp (a, b).$$

By splitting the first equation in $(x,y)^{\perp\perp}$ and its complement, we obtain

$$\left\{\begin{array}{ccc} PB(s,t) & = & -r, \\ \begin{pmatrix} \alpha \\ \beta \end{pmatrix} & = & \begin{pmatrix} (x^Tu)^{-1} & 0 \\ 0 & (y^Tv)^{-1} \end{pmatrix} \begin{pmatrix} x^T & 0 \\ 0 & y^T \end{pmatrix} B \begin{pmatrix} s \\ t \end{pmatrix}, \\ (s,t) & \perp\perp & (a,b). \end{array}\right.$$

Note that we have used $Pr = r$, $P(\alpha u, \beta v) = 0$, and $r \perp\perp (x,y)$. The first and third equation together are equivalent to the correction equation (5.1), and the second equation determines α, β uniquely. □

REMARK 7.2. *Of course, this equivalence is valid only when both* (7.7) *and* (5.1) *are solved exactly, not when we solve them approximately.*

In particular, when we substitute $\eta = \theta$ and $(a,b) = 2(u,v)$, the matrix in (7.7) becomes (7.6).

7.5. Smallest singular value. As mentioned in section 4.1, the standard variant of the JDSVD may have difficulties with finding the smallest singular value of a matrix. This is not surprising, because the small singular values of A correspond to the interior eigenvalues of the augmented matrix. But in many applications, e.g., the computation of pseudospectra, the smallest singular value is just what we want to compute.

We can use the JDSVD with the nonstandard Galerkin (harmonic) variants, mentioned in sections 4.3 and 4.4, starting with zero, or a small positive number as a target, and solve the correction equation rather accurately, possibly with the aid of a preconditioner; see section 8. In this way the method is close to a shift and invert iteration but less expensive. Of course it is hereby advantageous to have a good initial triple (e.g., coming from an iterative method on A^TA); the JDSVD (with nonstandard Galerkin) can then be used as refinement procedure.

7.6. Restart. A nice property of Jacobi–Davidson is its flexibility in restarting. The JDSVD, too, has this advantage: we can restart at every moment in the process with any number of vectors, only keeping those parts of the search spaces that look promising, or possibly adding some extra vectors. All we have to do is compute the new resulting $H = U^TAV$ and continue. This is practical when the search spaces become large or to avoid a breakdown in case of the nonstandard Galerkin choices. Of course, the JDSVD can also be *started* with search spaces of dimension larger than one.

7.7. Deflation. We can compute multiple singular triples of A by using a deflation technique. If we have found a singular triple of A, and we want to find another, we can deflate the augmented matrix to avoid finding the same triple again. For the JDSVD, this can be done as follows. Suppose that U_f and V_f contain the already found singular vectors. Then it can be checked that, if we found the exact vectors,

$$\begin{pmatrix} I_m - U_fU_f^T & 0 \\ 0 & I_n - V_fV_f^T \end{pmatrix} \begin{pmatrix} 0 & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} I_m - U_fU_f^T & 0 \\ 0 & I_n - V_fV_f^T \end{pmatrix}$$

has the same eigenvalues as the original augmented matrix, except that the found eigenvalues are transformed to zeros. The method can then be restarted with another approximate triple.

7.8. Preconditioning the correction equation. The correction equation of the JDSVD can be preconditioned in a manner similar to Jacobi–Davidson (see, for example, [17]). We use the same notation as in the proof of Theorem 6.2 for the important case $Q = P$. Suppose that we have a preconditioner M for B. For left preconditioning we are given $(s, t) \perp\perp (x, y)$, and we have to solve for $(z_1, z_2) \perp\perp (x, y)$ from

$$PMP(z_1, z_2) = PBP(s, t).$$

Note that we project the preconditioner as well. Hence, for some α, β,

$$(z_1, z_2) = M^{-1}B(s, t) - M^{-1}(\alpha u, \beta v),$$

and by using the test vectors we obtain

$$\begin{pmatrix} x & 0 \\ 0 & y \end{pmatrix}^T M^{-1} \begin{pmatrix} u & 0 \\ 0 & v \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} x & 0 \\ 0 & y \end{pmatrix}^T M^{-1}B \begin{pmatrix} s \\ t \end{pmatrix}.$$

A recipe for computing (z_1, z_2) is given by the following four steps.

(1) Compute $(\widetilde{u}_1, \widetilde{u}_2) = M^{-1}(u, 0)$ and $(\widetilde{v}_1, \widetilde{v}_2) = M^{-1}(0, v)$.
(2) Compute $(\widetilde{s}, \widetilde{t}) = M^{-1}B(s, t)$.
(3) Compute (α, β) from $\left(\begin{smallmatrix} x^T\widetilde{u}_1 & x^T\widetilde{v}_1 \\ y^T\widetilde{u}_2 & y^T\widetilde{v}_2 \end{smallmatrix}\right)\left(\begin{smallmatrix} \alpha \\ \beta \end{smallmatrix}\right) = \left(\begin{smallmatrix} x^T\widetilde{s} \\ y^T\widetilde{t} \end{smallmatrix}\right)$.
(4) Compute $(z_1, z_2) = (\widetilde{s}, \widetilde{t}) - \alpha(\widetilde{u}_1, \widetilde{u}_2) - \beta(\widetilde{v}_1, \widetilde{v}_2)$.

An important observation is that step (1) and the computation of the 2×2 matrix in step (3) have to be performed only once at the start of the iterative solution process of the correction equation. The right-hand side of the correction equation, minus the residual, is handled similarly.

8. Numerical experiments. Our experiments are coded in MATLAB and are executed on a SUN workstation. The following lemma implies that up to rounding errors, it is not a loss of generality to consider (rectangular) diagonal matrices A.

LEMMA 8.1. *If there are no rounding errors, and the JDSVD's correction equation* (5.1) *in step k is solved by l_k steps of GMRES, then the JDSVD applied to*

(a) $A = U_*\Sigma V_*^T$, *with starting vectors u_1 and v_1,*
(b) Σ, *with starting vectors $\widetilde{u}_1 := U_*^T u_1$ and $\widetilde{v}_1 := V_*^T v_1$,*

gives "the same" results; that is,

$$\widetilde{\theta}_k = \theta_k \qquad \textit{and} \qquad \|\widetilde{r}_k\| = \|r_k\|.$$

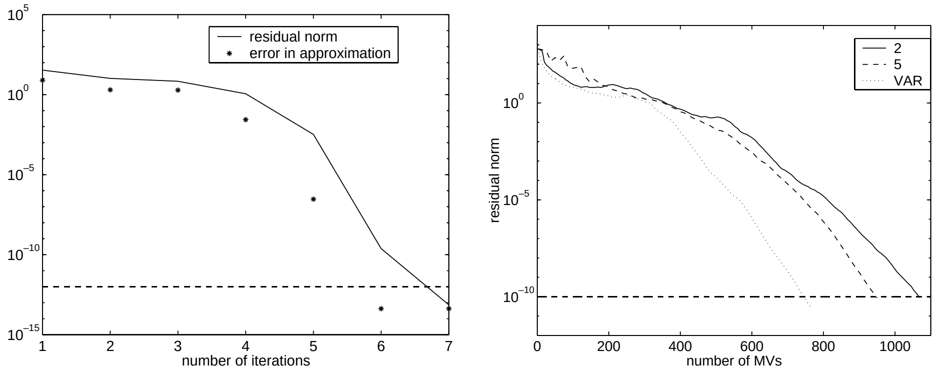
Proof. Define

$$Q = \begin{pmatrix} U_*^T & 0 \\ 0 & V_*^T \end{pmatrix};$$

then Q is orthogonal, and one may verify that $(\widetilde{u}_1, \widetilde{v}_1) = Q(u_1, v_1)$, $\widetilde{\theta}_1 := \widetilde{u}_1^T\Sigma\widetilde{v}_1 = u_1^TAv_1 =: \theta_1$, and $\widetilde{r}_1 = Qr_1$. A well-known property of Krylov subspaces ensures that (see [12, p. 264])

$$Q^T\mathcal{K}_l\left(\begin{pmatrix} 0 & \Sigma \\ \Sigma^T & 0 \end{pmatrix}, \widetilde{r}\right) = \mathcal{K}_l\left(Q^T\begin{pmatrix} 0 & \Sigma \\ \Sigma^T & 0 \end{pmatrix}Q, Q^T\widetilde{r}\right) = \mathcal{K}_l\left(\begin{pmatrix} 0 & A \\ A^T & 0 \end{pmatrix}, r\right).$$

With little extra work one can check that the same relation holds for the shifted and projected matrices that are present in the correction equation (5.1), where one



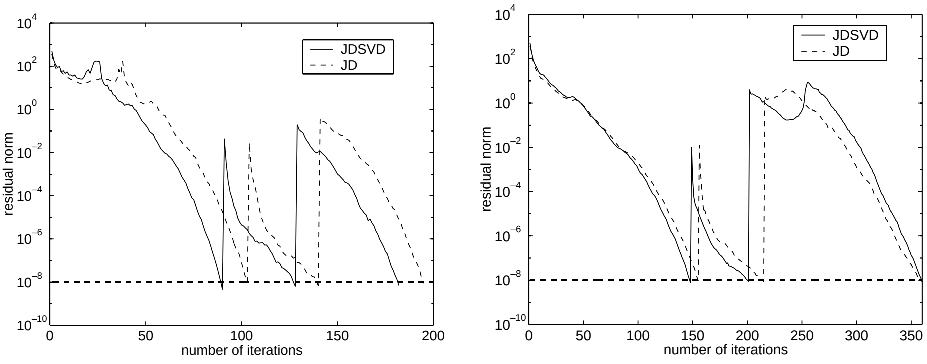
(a) (b)

FIG. 8.1. (a) *The convergence history of the exact JDSVD algorithm for* diag(1..100) *as in Algorithm* 4.1: *residual norm (solid line) and error in the approximations to* $\sigma_{\max}$ *(dots). The horizontal dotted line indicates the stopping tolerance.* (b) *Convergence for* diag(1..1000) *using, respectively,* 5, 2, *and a variable number of GMRES steps to solve the correction equation.*

should bear in mind that all other vectors involved in the projectors (a, b, x, y, $\widetilde{x}$, and $\widetilde{y}$) must also be altered for the Σ-system in the obvious way. So the approximate solutions from the correction equations satisfy $(\widetilde{s}_1, \widetilde{t}_1) = Q(s_1, t_1)$. By induction we can prove that $\widetilde{U}_k = U_*^T U_k$ and $\widetilde{V}_k = V_*^T V_k$, so the projected matrices are the same in both cases: $\widetilde{H}_k := \widetilde{U}_k^T \Sigma \widetilde{V}_k = U_k^T A V_k = H_k$. In particular, the approximations to the singular values are the same, and the approximations (u_k, v_k) and $(\widetilde{u}_k, \widetilde{v}_k)$ are orthogonal transformations of each other: $(\widetilde{u}_k, \widetilde{v}_k) = Q(u_k, v_k)$ and $\widetilde{r}_k = Qr_k$, so $\|\widetilde{r}_k\| = \|r_k\|$. □

For this reason, we first study some phenomena on $A = \text{diag}([1 : 100])$ and $A = \text{diag}([1 : 1000])$. In Figure 8.1(a), the solid line is the convergence history of (the standard variant of Algorithm 4.1 of) the JDSVD for the computation of the largest singular triple of $A = \text{diag}([1 : 100])$. The starting vectors are the normalized $v_1 = v_{\max} + 0.1r$, where r is a vector with random entries, chosen from a uniform distribution on the unit interval, and $u_1 = Av_1/\|Av_1\|$. The dots represent the error in the approximation $\sigma_{\max} - \theta_k^{(k)}$. In all figures, a horizontal dotted line indicates the stopping tolerance. We solve the correction equation by 200 steps of GMRES. Because the (augmented) matrices in the correction equation (step 7 of Algorithm 4.1) are of size 200×200, this means (theoretically) exactly, so according to Theorem 6.2 we expect cubic convergence. In Figure 8.1(a) we see, for instance, that the error in the approximation in iteration number 5 decreases from $\approx 10^{-2}$ to $\approx 10^{-7}$.

In Figure 8.1(b), we take $A = \text{diag}([1 : 1000])$, and u_1 and v_1 random vectors (as described above) with unit norm. We experiment with the number of GMRES steps. For the solid line, we solve the correction equation approximately by five steps of GMRES, which we denote by GMRES$_5$, for the dashed line by GMRES$_2$, and for the dotted line by a variable number equal to $\max\{2 \cdot (\lceil -\log \|r\| \rceil + 1), 0\}$. Measured in terms of matrix-vector products (MVs), the variable choice is best, followed by GMRES$_5$. An explanation of this is that when the initial approximations are not good (as in this case), it is of no use to try hard to solve the correction equation in the beginning. When we are almost converging, it may make sense to solve it more



(a) (b)

FIG. 8.2. (a) *The JDSVD (solid) and Jacobi–Davidson (dashed) for the three largest* σ*s of* diag(1..1000)*.* (b) *The same as Figure* 8.2(a)*, only with GMRES*$_2$ *to solve the correction equation.*

accurately to get fast convergence. See also [17].

In Figure 8.2(a) we compare, for $A = \mathrm{diag}([1 : 1000])$, the standard JDSVD for the three largest singular triples (solid), with Jacobi–Davidson on the augmented matrix for the computation of the three largest eigenpairs (dashed), each with GMRES$_5$. For the JDSVD, we take v_1 as a random vector, and $u_1 = Av_1/\|Av_1\|$. For Jacobi–Davidson we take the "same" starting vector $(u_1, v_1)/\sqrt{2}$. We see that the JDSVD is faster for the first triple; for the second and third we restart with a good approximation, and then the histories are similar.

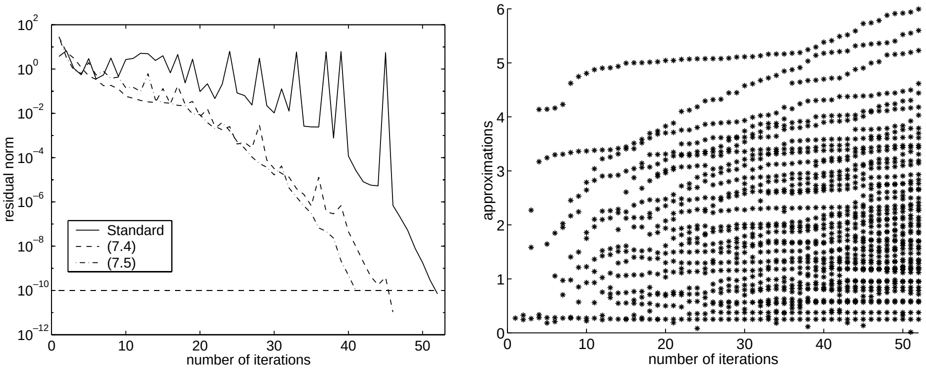
In Figure 8.2(b), we do the same, but now using GMRES$_2$. For the first two triples, the JDSVD is somewhat faster than Jacobi–Davidson, for the third JDSVD in the first instance (mis)converges to the fourth largest singular value 997. Other experiments also suggest that the JDSVD is generally (somewhat) faster than Jacobi–Davidson on the augmented matrix.

Next, we take some examples from the Matrix Market (these matrices can be downloaded from `http://math.nist.gov/MatrixMarket`). For Figure 8.3(a), we apply different JDSVD variants to find the smallest singular triple of PDE225, using two random starting vectors and GMRES$_{10}$ (no preconditioning). In all variants, we take initially target 0, but when $\|r\| < 10^{-3}$, we replace the target by the best approximations again (see section 7.1, point 4). The solid line is the standard choice; we see an irregular convergence history, as could be expected (see section 4). The dashed line represents the Galerkin choice (4.5), where in the correction equation (5.1) we substitute (7.4). Finally, the dash-dotted line is (4.5) with (7.5) substituted in (5.1). In the last case, as seen in section 7.2, the operator in (5.1) maps $(u, v)^{\perp\perp}$ to $(x, y)^{\perp\perp}$. Since in this case $v = y$ but $u \neq x$, we use a left "preconditioner" to handle the correction equation correctly. The preconditioned identity

$$\begin{pmatrix} I_m - \frac{xu^T}{u^Tx} & 0 \\ 0 & I_n \end{pmatrix} I_{m+n} \begin{pmatrix} I_m - \frac{ux^T}{x^Tu} & 0 \\ 0 & I_n \end{pmatrix}$$

maps $(x, y)^{\perp\perp}$ back to $(u, v)^{\perp\perp}$.

In Figure 8.3(b), the standard JDSVD's approximations to the singular values during this process are plotted. These are "regular," nonharmonic estimates. Note



(a) (b)

FIG. 8.3. (a) *Three different JDSVD variants for the computation of* $\sigma_{\min}$ *of* PDE225: *standard,* (4.5) + (5.1) + (7.4), *and* (4.5) + (5.1) + (7.5). (b) *(Nonharmonic) approximations to the singular values by the standard variant.*

the monotone convergence of the approximations to the largest singular values but the irregular behavior of the approximations to the smallest singular value.

Next, we compare the JDSVD with Lanczos applied to A^TA for the computation of $\sigma_{\max}$. These methods are of a different nature. The Lanczos method can be viewed as an accelerated power method, while the JDSVD can be seen as an accelerated inexact RQI. An advantage of the JDSVD is that we may use preconditioning for the correction equation. Therefore, we expect that if we have a reasonable preconditioner, and if preconditioning is relatively cheap in comparison to a multiplication by A or A^T, then the JDSVD can be cheaper than Lanczos. On the other hand, if $m \gg n$, or if there is no good or cheap preconditioner available, then we expect that Lanczos will be better. Table 8.1 shows some test results. For the JDSVD, we take a target $\tau \approx \|A\|_\infty$, in the hope that $\tau \approx \sigma_{\max}$. We make an incomplete LU-decomposition (using a drop tolerance displayed in the table) of the augmented matrix (1.1) minus τ times the identity, and we use $M = LU$ as a preconditioner. The starting vector v_1 is the vector with all coordinates equal to one, and is then normalized, and u_1 is a random vector. We solve the correction equation by only preconditioning the residual ("0 steps of GMRES"). The process is continued until $\|r\| < 10^{-8}$. Lanczos's method uses v_1 as starting vector and continues until $\|(A^TA - \theta^2)v\| < 10^{-8}$. The matrix A_1 stands for $\mathrm{diag}(1:100) + 0.1 \cdot \mathrm{rand}(n,n)$, where $\mathrm{rand}(n,n)$ denotes an $n \times n$-matrix with random entries, chosen from a uniform distribution on the unit interval. See [13] for more information on the origin and singular values of the other matrices. For the JDSVD, a pair is given, consisting of the number of MVs and the number of systems with L or U. For Lanczos we show the number of MVs.

For HOR131, the target τ is relatively far from $\sigma_{\max} \approx 0.66$. We see that although the JDSVD uses fewer MVs than Lanczos, Lanczos is cheaper when we take the preconditioning into account. Although for PORES3 ($\sigma_{\max} \approx 1.5 \cdot 10^5$) Lanczos finds a good approximate vector, its residual does not reach the required 10^{-8} due to the high condition number of the matrix. The JDSVD does converge, so this is an example of a situation where the JDSVD could be used as refinement. For SHERMAN1, the target is a reasonable approximation to $\sigma_{\max} \approx 5.05$. When we

Table 8.1
Some experiments with the JDSVD to compute $\sigma_{\max}$, using incomplete LU-factorizations of the shifted augmented matrix. The number of MVs, and the number of systems with L or U is displayed in the 5th column. The shift (or target) τ (6th column) for the preconditioning is roughly taken to be $\|A\|_\infty$. The last three columns give information on the incomplete LU-factorization: the drop tolerance of ILU, and the resulting number of nonzeros of L and U. We compare the JDSVD's results with the MVs of Lanczos applied to A^TA (4th column).

Matrix	Size	nnz(A)	Lan(A^TA)	JDSVD	τ	tol	nnz(L)	nnz(U)
HOR131	434 × 434	4182	30	(28, 65)	0.90	10^{-2}	1792	1792
PORES3	532 × 532	3474	–	(72, 175)	$2 \cdot 10^5$	10^{-1}	1301	1300
SHERMAN1	1000 × 1000	3750	74	(24, 66)	5	10^{-2}	4805	4803
A_1	100 × 100	10000	102	(38, 108)	106	10^{-2}	299	299

take the preconditioning into account, Lanczos is cheaper than the JDSVD. The last row of the table is an example where preconditioning is relatively cheap. The reason for this is that we now take the diagonal of A, instead A itself, to form an augmented matrix of the form (1.1) and to make an ILU-decomposition. Using far more MVs, Lanczos is (also counting the preconditioning) more expensive.

Finally, in Table 8.2, we compare the JDSVD for the computation of $\sigma_{\min}$ with Lanczos applied to $(A^TA)^{-1}$. We use the Galerkin choice (4.5) for the JDSVD. Note that the comparison with Lanczos is mainly meant to get an idea of how well the JDSVD performs. In practice, for large (sparse) A, it is too expensive to work with A^{-1} and A^{-T} or $(A^TA)^{-1}$. For the JDSVD, we take a small target $\tau = 10^{-5}$, drop tolerance 10^{-3}, and again we make an incomplete LU-decomposition based on this target. The starting vectors are the same as for Table 8.1. We solve the correction equation by preconditioning only the residual ("0 steps of GMRES"). Both processes are continued until $\|r\| < 10^{-7}$.

Table 8.2
Some experiments with the JDSVD to compute $\sigma_{\min}$. The numbers of MVs and systems with L or U (3rd column), and the number of nonzeros of L and U are displayed. We compare the JDSVD's results with the number of MVs of Lanczos applied to $(A^TA)^{-1}$.

Matrix	Lan(A^TA)$^{-1}$	JDSVD	nnz(L)	nnz(U)
HOR131	–	(26, 72)	20593	21167
PORES3	12	(36, 108)	3683	5491
SHERMAN1	20	(20, 54)	11575	11738
A_1	14	(28, 78)	200	200

We see that although the JDSVD may in general use more MVs, it may be much cheaper than Lanczos, due to the sparsity of A, L, and U. For HOR131, Lanczos does not converge to the required 10^{-7}. Again A_1 serves as an example for the situation where preconditioning is relatively cheap, which makes the JDSVD attractive. We also tried Lanczos applied to A^TA for the computation of $\sigma_{\min}$, but the results were bad (262 MVs for A_1, and more than 500 MVs for the other matrices).

9. Conclusions. We have discussed an alternative approach for the computation of a few singular values and vectors of a matrix. The JDSVD method searches in two separate subspaces, and it can be interpreted as an inexact Newton method for the singular value problem. The JDSVD can also be seen as an inexact accelerated scaled RQI method. Therefore, the best results may be expected when we have a good initial starting triple (refinement), but we can start with any approximations. While the asymptotic convergence is cubic if the correction equation is solved exactly, in practice we solve it approximately, and then the convergence typically looks (super)linear. Although we mainly discussed the application of the JDSVD for the largest and smallest singular value, the method is in principle suitable for all singular values. We may use preconditioning for the solution of the correction equation. This can be a decisive factor for fast convergence. Experiments indicate that the JDSVD is a good competitor to other iterative SVD methods, in particular when A is (almost) square and we have a reasonable, relatively cheap preconditioner for the correction equation.

Acknowledgments. The author thanks Henk van der Vorst and Gerard Sleijpen for interesting discussions, and the referees for many helpful comments.

REFERENCES

[1] J. Cullum, R. A. Willoughby, and M. Lake, *A Lánczos algorithm for computing singular values and vectors of large matrices*, SIAM J. Sci. Statist. Comput., 4 (1983), pp. 197–215.

[2] E. R. Davidson, *The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices*, J. Comput. Phys., 17 (1975), pp. 87–94.

[3] J. J. Dongarra, *Improving the accuracy of computed singular values*, SIAM J. Sci. Statist. Comput., 4 (1983), pp. 712–719.

[4] D. R. Fokkema, G. L. G. Sleijpen, and H. A. van der Vorst, *Accelerated inexact Newton schemes for large systems of nonlinear equations*, SIAM J. Sci. Comput., 19 (1998), pp. 657–674.

[5] G. Golub and W. Kahan, *Calculating the singular values and pseudo-inverse of a matrix*, J. Soc. Indust. Appl. Math. Ser. B Numer. Anal., 2 (1965), pp. 205–224.

[6] G. H. Golub, F. T. Luk, and M. L. Overton, *A block Lanczos method for computing the singular values and corresponding singular vectors of a matrix*, ACM Trans. Math. Software, 7 (1981), pp. 149–169.

[7] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed., The John Hopkins University Press, Baltimore, London, 1996.

[8] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, Cambridge, UK, 1985.

[9] R. A. Horn and C. R. Johnson, *Topics in Matrix Analysis*, Cambridge University Press, Cambridge, UK, 1991.

[10] C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, Journal of Research of the National Bureau of Standards, 45 (1950), pp. 255–282.

[11] C. C. Paige, B. N. Parlett, and H. A. van der Vorst, *Approximate solutions and eigenvalue bounds from Krylov subspaces*, Numer. Linear Algebra Appl., 2 (1995), pp. 115–133.

[12] B. N. Parlett, *The Symmetric Eigenvalue Problem*, SIAM, Philadelphia, 1997 (corrected reprint of the 1980 original).

[13] B. Philippe and M. Sadkane, *Computation of the fundamental singular subspace of a large matrix*, Linear Algebra Appl., 257 (1997), pp. 77–104.

[14] G. L. G. Sleijpen, A. G. L. Booten, D. R. Fokkema, and H. A. van der Vorst, *Jacobi-Davidson type methods for generalized eigenproblems and polynomial eigenproblems*, BIT, 36 (1996), pp. 595–633.

[15] G. L. G. Sleijpen and H. A. van der Vorst, *A Jacobi–Davidson iteration method for linear eigenvalue problems*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 401–425.

[16] G. L. G. Sleijpen and H. A. van der Vorst, *The Jacobi-Davidson method for eigenvalue problems and its relation with accelerated inexact Newton schemes*, in Iterative Methods in Linear Algebra II, IMACS Series in Computational and Applied Mathematics 3, S. D. Margenov and P. S. Vassilevski, eds., IMACS, New Brunswick, NJ, 1996, pp. 377–389.

[17] G. L. G. Sleijpen, H. A. van der Vorst, and E. Meijerink, *Efficient expansion of subspaces in the Jacobi-Davidson method for standard and generalized eigenproblems*, Electron. Trans. Numer. Anal., 7 (1998), pp. 75–89.

[18] G. W. Stewart and J.-g Sun, *Matrix Perturbation Theory*, Academic Press, San Diego, 1990.

[19] S. Van Huffel, *Iterative algorithms for computing the singular subspace of a matrix associated with its smallest singular values*, Linear Algebra Appl., 154/156 (1991), pp. 675–709.

[20] S. Varadhan, M. W. Berry, and G. H. Golub, *Approximating dominant singular triplets of large sparse matrices via modified moments*, Numer. Algorithms, 13 (1996), pp. 123–152.

[21] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, UK, 1965.

SIAM J. SCI. COMPUT.
Vol. 23, No. 2, pp. 629–650

ELEMENT-FREE AMGe: GENERAL ALGORITHMS FOR COMPUTING INTERPOLATION WEIGHTS IN AMG[*]

VAN EMDEN HENSON[†] AND PANAYOT S. VASSILEVSKI[†]

Abstract. We propose a new general algorithm for constructing interpolation weights in algebraic multigrid (AMG). It exploits a proper extension mapping outside a neighborhood about a fine degree of freedom (dof) to be interpolated. The extension mapping provides boundary values (based on the coarse dofs used to perform the interpolation) at the boundary of the neighborhood. The interpolation value is then obtained by matrix dependent harmonic extension of the boundary values into the interior of the neighborhood.

We describe the method, present examples of useful extension operators, provide a two-grid analysis of model problems, and, by way of numerical experiments, demonstrate the successful application of the method to discretized elliptic problems.

Key words. algebraic multigrid, interpolation weights, sparse matrices, finite elements, unstructured meshes

AMS subject classifications. 65F10, 65N20, 65N30

PII. S1064827500372997

1. Introduction.

The classical algebraic multigrid (AMG) algorithm [2, 3, 9] was developed for operators represented by symmetric, positive definite M-matrices. While the algorithm works well for many real-world problems [10, 6, 11], there are situations in which it does not perform particularly well. One reason for this is that in some instances the classical definition of interpolation does not adequately interpolate the smooth modes of the error. More specifically, standard AMG interpolation makes certain assumptions about the nature of the smooth error which may not be valid for operators that are not M-matrices. A more sophisticated characterization of smooth error is required to develop an adequate interpolation formula.

To provide a better characterization of smooth error, a method known as AMGe, for element-based algebraic multigrid, was developed recently [4] for finite element discretizations. AMGe provides an accurate interpolation formula by using the individual element stiffness matrices to construct a neighborhood matrix for each fine degree of freedom (dof). The neighborhood matrix—the sum of the individual stiffness matrices for all the elements containing the point at which the dof is defined—acts as a local “Neumann”-type version of the original operator. According to AMGe theory, once the local matrix is developed and coarse-grid points are chosen, solving a simple minimization problem yields the optimal interpolation operator for each dof. It is shown in [4] that the method indeed produces superior interpolation and leads to improved convergence rates on several types of problems, including both scalar problems and systems of PDEs, such as elasticity problems.

An obvious drawback to AMGe, naturally, is that it requires that the element stiffness matrices be available. While this is often the case, their storage can be expensive. Further, AMGe requires that coarse level elements be constructed and

[*]Received by the editors June 6, 2000; accepted for publication (in revised form) January 10, 2001; published electronically July 10, 2001. This work was performed under the auspices of the U. S. Department of Energy by the University of California Lawrence Livermore National Laboratory under contract W-7405-Eng-48.

http://www.siam.org/journals/sisc/23-2/37299.html

[†]Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, 7000 East Avenue, Mail stop L-560, Livermore, CA 94550 (vhenson@llnl.gov, panayot@llnl.gov).

their individual stiffness matrices be available. Determining the coarse elements is a difficult and laborious task.

In this paper we examine the construction of the interpolation operator in both classical AMG and AMGe and present them within a common framework. Our purpose is to extend and generalize the classical interpolation, which was originally motivated for *M*-matrices, to develop a rule applicable in more general settings. Accordingly, we propose a new method for determining the interpolation weights that attempts to capture the benefits of AMGe interpolation without requiring access to the individual element stiffness matrices. This method is applicable to finite difference, finite element, or finite volume discretizations, and we concentrate on the symmetric positive definite case. Essentially, it seeks to determine, for each fine dof, a neighborhood matrix that can be utilized in the same manner that the local assembled stiffness matrix is used in AMGe. We do this by defining a neighborhood for the fine dof and examining the rows of the original matrix that correspond to the points in that neighborhood. A set of exterior dofs is defined and a mapping developed that extends functions on the neighborhood to the exterior dofs. This essentially imposes a set of boundary conditions on the neighborhood. Here we propose a unified way of building these boundary conditions. One may view them as an extension (extrapolation) of a vector defined on the neighborhood to its immediate exterior. This extension can be performed using constant vectors or any other vectors that may be of interest (such as the rigid body motions in elasticity problems). The extension can be built for each dof in the exterior based on the matrix sparsity pattern.

By incorporating the action of the extension operator into the local connections of the neighborhood, a modified local matrix is created. This matrix is then used in a manner similar to that employed in AMGe, that is, by solving a minimization problem, to create the interpolation operator. We consider the construction of the extension operator and the respective minimization procedure to build the interpolation weights as our main contribution. We give examples of several extension operators and show how they relate to both classical AMG and other, more recently proposed algorithms. A two-grid model analysis of the properties of the resulting interpolation mappings is provided as well. In particular, we prove that they exhibit approximately "harmonic" properties as well as "partition of unity" properties, desirable in standard two-grid analyses of the AMG methods.

Numerical results are presented to demonstrate the method. We include both scalar problems and systems of PDEs in the form of elasticity problems. Finally, we draw some conclusions and comment on the direction that continued research will take.

It is important to note that while the choice of coarse-grid points, like the construction of the interpolation operator, is crucial to the success of the AMG method, we do not consider the coarse-grid selection here; rather, we leave that topic to future research while focusing on the interpolation problem here. Furthermore, we observe that neither AMGe nor the method proposed here are intended to replace or compete with classical AMG on problems characterized by simple *M*-matrices, such as the model Laplacian problem on a regular grid. Instead, they are intended for complicated problems, such as thin-body elasticity, posed typically on unstructured grids. Nonetheless, we apply the new method (and AMGe) to model problems because they illustrate, in simple fashion, the features of the methods. We therefore compare results of our method with AMGe on these problems but do not include comparisons to classical AMG, which would be used in practice.

Some notational conventions follow: to denote a vector we will use boldface, e.g., $\mathbf{v}, \mathbf{w}, \ldots$. The ith component of $\mathbf{v}$ will be denoted in different contexts as $\mathbf{v}(i)$, $v(i)$, or v_i. In the latter two cases v (i.e., not in boldface) will have a meaning of a "grid" function.

2. A framework for AMG interpolation.

Assume that the problem $A\mathbf{x} = \mathbf{f}$ is to be solved, where A is a sparse, symmetric, positive definite matrix. AMG is a multigrid method in which no geometric grid information is used (and often isn't available or doesn't even exist). Accordingly, all of the components of a multigrid algorithm, the hierarchy of grids, interpolation and restriction operators, and the coarse-grid versions of the original operator must be constructed using only the information contained in the entries of A. For any multigrid algorithm, several basic components are required. In the case of AMG, they can be described as follows:

- A fine grid is required. For AMG, this is generally a set D comprising the degrees of freedom of the original problem.
- A coarse grid D_c is necessary. This set of dofs is typically a subset of D.
- An interpolation operator P is necessary to map vector functions defined on the coarse grid D_c to the fine grid D, $P: D_c \longrightarrow D$.
- A restriction operator $R: D \longrightarrow D_c$, mapping fine-grid functions to the coarse grid, is needed. For AMG the restriction is frequently defined by $R = P^T$, and we will use that definition here.
- A coarse-grid version of the original operator A is needed. For AMG the coarse operators are generally defined by the Galerkin relation $A_c = P^TAP$.
- A smoothing iteration, G, is used. It is typical to use a point-relaxation method such as Gauss–Seidel or Jacobi relaxation.

The basic two-grid algorithm can then be described as follows: Begin with an initial approximation $\mathbf{x}_0$ to the solution of $A\mathbf{x} = \mathbf{f}$.

1. Smooth the error by $\mathbf{x}_0 \leftarrow G(A, \mathbf{f}, \mathbf{x}_0)$.
2. Compute the residual $\mathbf{r} = \mathbf{f} - A\mathbf{x}_0$.
3. Restrict the residual to the coarse grid $\mathbf{f}_c = R\mathbf{r}$.
4. Solve the coarse-grid residual equation $\mathbf{e}_c = A_c^{-1}\mathbf{f}_c$.
5. Interpolate the coarse-grid error to the fine grid and correct the fine-grid approximation $\mathbf{x}_0 \leftarrow \mathbf{x}_0 + P\mathbf{e}_c$.

For a multigrid method, a hierarchy of coarse grids $D \equiv D_0 \supset D_1 \supset D_2 \supset \cdots \supset D_J$ is present, and the two-grid algorithm is applied recursively, i.e., the two-grid algorithm is repeated each time step 4 is encountered, except that a direct solve is employed at the coarsest grid.

For the multigrid method an interpolation operator P_i is required mapping functions on each grid (level i) to the next finer grid (level $i-1$). Unlike many conventional (geometric) multigrid algorithms, in AMG the interpolation operators are rarely the same for different levels. Similarly, the Galerkin relation is employed to define versions of the original operator A on all levels, thusly: $A_{i+1} \equiv R_iA_iP_i$.

There are many ways in which to select the coarse-grid dofs in AMG [9, 11, 5]. Commonly, the coarse set D_c is a maximally independent subset of D, but this is not required. We will not discuss the question of coarse-grid selection further, except to note that each fine-grid dof i is connected to its nearest neighbors (e.g., j) by way of having a nonzero coefficient a_{ij}, and that the value of a interpolated function at i is typically a weighted average of the values of its nearest neighbors that are coarse-grid dofs. For the remainder of this paper, we shall simply assume that a coarse grid has been selected and that the coarse neighbors are known for any fine dof.

With this description of the basic components of AMG, we can describe a simple framework for computing the entries of the interpolation operator. Let $i \in D$ be a fine-grid dof whose value is to be interpolated. We first define a subset $\Omega(i) \subset D$ to be the *neighborhood* of i. For now we place no particular restrictions on what dofs can be in $\Omega(i)$. For example, the set $\Omega(i)$ could consist of i and all of its nearest neighbors, or i and its nearest coarse neighbors, or i, its neighbors, and all of their neighbors. Indeed, within the framework we describe here, the exact character of the interpolation operator will depend largely on what sort of neighborhood is defined. Since the value at i will be interpolated from coarse points in the neighborhood, it is useful to denote the set of coarse dofs in the neighborhood to be $\Omega_c(i)$.

To construct the interpolation for i, we examine the entries of the operator A in the following way. We begin, without loss of generality, by permuting the rows and columns of A and partitioning it so the first set of rows and columns corresponds to i and the fine dofs in the neighborhood, that is, to $\Omega(i)\backslash\Omega_c(i)$. The next set of rows and columns corresponds to the coarse neighbors $\Omega_c(i)$, while the final set of rows and columns corresponds to the rest of the grid $D\backslash\Omega(i)$. Hence the partitioning of A, along with the identity of the rows corresponding to the partitions, appears as

$$A = \begin{pmatrix} A_{ff} & A_{fc} & * \\ * & * & * \\ * & * & * \end{pmatrix} \begin{array}{l} \}\ \Omega(i)\backslash\Omega_c(i), \\ \}\ \Omega_c(i), \\ \}\ D\backslash\Omega(i). \end{array}$$

For our purposes we are concerned only with two blocks of the partitioned matrix. The block A_{ff} gives the connections among i and the fine-grid neighbors while the block A_{fc} links i and the fine neighbors to the coarse neighbors.

Armed with these concepts of neighborhood partitioning of the operator, we can examine classical AMG, AMGe, and our proposed method in terms of choices of neighborhood and definition of the neighborhood matrices.

3. Interpolation in classical AMG. For classical AMG [9], the interpolation is computed in the following fashion. The neighborhood $\Omega(i)$ is defined to be the dof i and all dofs connected to it (all j for which $a_{ij} \neq 0$). We then replace both A_{ff} and A_{fc} with modified versions, $\widehat{A}_{ff}$ and $\widehat{A}_{fc}$, respectively.

A_{ff} is modified in two ways. Let S_i denote the set of dofs that are *strongly connected* to the dof i, let C_i denote the set of C-points in the neighborhood $\Omega(i)$, and let W_i denote the dofs that are *weakly connected* to the dof i. (By strongly (weakly) connected we mean that the magnitude of a_{ij} is greater (smaller) than some predefined threshold. A common choice is that if the magnitude of a_{ij} is less than θ times the largest magnitude of all off-diagonal entries in the ith row, then j is considered to be weakly connected to i.) Then we modify the row of A_{ff} corresponding to the dof i (which we will hereafter refer to as the ith row, regardless of the actual numerical ordering) by

$$\widehat{a}_{ii} = a_{ii} + \sum_{j \in W_i} a_{ij}, \tag{3.1}$$

$$\widehat{a}_{i,j} = \begin{cases} 0, & j \in W_i, \\ a_{i,j}, & j \in S_i. \end{cases} \tag{3.2}$$

For $j \in W_i$ we replace the jth row of A_{ff} by a zero row and then place a 1 in column j and -1 in column i. For all other rows of A_{ff}, i.e., for $j \in S_i$, we zero out the off-diagonal entries, and replace the diagonal entry a_{jj} with

$$\widehat{a}_{jj} = -\sum_{k \in C_i} a_{j,k}.$$

The block A_{fc} is modified to $\widehat{A}_{fc}$ by zeroing the jth row for $j \in W_i$.

Once the modified blocks $\widehat{A}_{ff}$ and $\widehat{A}_{fc}$ are computed, the entries of the ith row of the interpolation matrix P are determined by taking the entries of the ith row of the matrix

$$-\left(\widehat{A}_{ff}^{-1}\widehat{A}_{fc}\right).$$

4. Interpolation in AMGe. For AMGe a similar description of the interpolation is easily given. In this setting, the neighborhood $\Omega(i)$ is defined naturally as the union of all finite elements having i as a vertex (Figure 4.1). In the figure, the set $\Omega(i)$ consists of all vertices in the shaded region, including i (the open circle in the center). The shaded region consists of the six triangular finite elements having i as a vertex. Members of $\Omega_c(i)$ are indicated by the square vertices. Since AMGe gives us access to the individual element stiffness matrices, we may create a neighborhood matrix $A_{\Omega(i)}$ simply by summing together all the individual element stiffness matrices of the elements in the neighborhood. In AMGe the interpolation operator for the dof i is determined by solving a constrained min-max problem, that is, by finding interpolation coefficients that minimize a certain measure from finite element theory. The solution to the min-max problem can be computed in several ways, one of which fits into the framework we are developing here. We partition the neighborhood matrix into the rows and columns associated with the fine dofs in the neighborhood and the rows and columns associated with the coarse dofs, as

$$A_{\Omega(i)} = \begin{pmatrix} A_{ff} & A_{fc} \\ * & * \end{pmatrix} \begin{array}{l} \} \; \Omega(i) \setminus \Omega_c(i), \\ \} \; \Omega_c(i). \end{array}$$

Again, our only interest is in the rows of the neighborhood matrix corresponding to the fine dofs, including i. With this partitioning, it turns out that one way to solve the min-max problem is to take, as the coefficients for the interpolation operator for i, the entries of the ith row of the matrix

$$-\left(A_{ff}^{-1}A_{fc}\right).$$

It is useful to note that, unlike the classical AMG case, there is no need to modify the matrix A_{ff} prior to computing the interpolation coefficients. Essentially, this is because the element stiffness matrices have built into them local versions of the null space and near-null space of the operator; we do not need to make alterations to the local matrices to ensure that these spaces are represented.

For many problems the AMGe method produces a superior interpolation and results in good convergence rates [4]. In the remainder of this paper our goal is to accomplish a superior interpolation without the knowledge (and hence, expense) of the individual stiffness matrices.

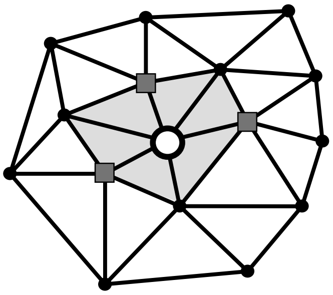


FIG. 4.1. *The neighborhood of the fine dof i (large open circle).*

5. Interpolation for element-free AMGe. The process we propose for building the interpolation operator is very similar to the processes described for AMG and AMGe. Once again, we will proceed by defining a neighborhood of the fine dofs and an associated neighborhood matrix. Let ψ be a set of fine dofs whose values we wish to interpolate. We define $\Omega(\psi)$ to be the neighborhood of ψ, which includes the coarse dofs that will be used to interpolate the dofs in ψ. The set of coarse dofs in the neighborhood we denote $\Omega_c(\psi)$.

Now, however, we define a third set of dofs:

$$\Omega_{\mathcal{X}}(\psi) = \{j \notin \Omega(\psi) \mid a_{ij} \neq 0 \text{ for some } i \in \Omega(\psi) \backslash \Omega_c(\psi)\}\,.$$

That is, $\Omega(\psi)$ can be viewed as the interior of the set $\overline{\Omega}(\psi) \equiv \Omega(\psi) \cup \Omega_{\mathcal{X}}(\psi)$. Figure 5.1 gives an example of such a neighborhood.

We begin the construction of a neighborhood matrix by examining the rows of the matrix A that correspond to the fine dofs in $\Omega(\psi)$; that is, we will be concerned with the following partitioning of A:

$$A = \begin{pmatrix} A_{ff} & A_{fc} & A_{f\mathcal{X}} & 0 \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{pmatrix} \begin{array}{l} \} \ \Omega(\psi)\ \backslash \Omega_c(\psi), \\ \} \ \Omega_c(\psi), \\ \} \ \Omega_{\mathcal{X}}(\psi), \\ \} \ \text{everything else on grid.} \end{array}$$

5.1. Local (neighborhood) quadratic form. Our next task is to define a matrix associated with ψ that yields a local version of the operator A, performing the same function as does the neighborhood matrix in AMGe. To do this we first build an extension mapping (matrix) $E(\psi)$ that maps a vector defined on $\Omega(\psi)$ to $\overline{\Omega}(\psi)$,

$$E(\psi) : \begin{pmatrix} \mathbf{v}_f \\ \mathbf{v}_c \end{pmatrix} \longrightarrow \begin{pmatrix} \mathbf{v}_f \\ \mathbf{v}_c \\ \mathbf{v}_{\mathcal{X}} \end{pmatrix},$$

using the relation

$$\mathbf{v}_{\mathcal{X}} = E_{\mathcal{X}f}(\psi)\mathbf{v}_f + E_{\mathcal{X}c}(\psi)\mathbf{v}_c.$$

That is, the extension operator looks like

$$E = \begin{pmatrix} I & 0 \\ 0 & I \\ E_{\mathcal{X}f}(\psi) & E_{\mathcal{X}c}(\psi) \end{pmatrix}.$$

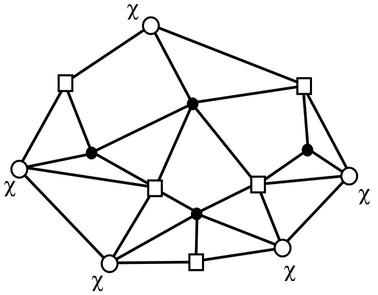


FIG. 5.1. *The extended neighborhood $\overline{\Omega}(\psi)$, including the fine dofs to be interpolated (solid circles), the coarse interpolatory set $\Omega_c(\psi)$ (squares), and the extension dofs (open circles marked $\mathcal{X}$).*

For now we will not be specific about the exact nature of the extension operator. Rather, we will describe how it may be used to develop an interpolation formula, after which we shall discuss desirable properties of the operator.

We construct a neighborhood matrix from the first block of rows of the partitioned matrix

$$\left(\widehat{A}_{ff}, \widehat{A}_{fc}\right) = (A_{ff}, A_{fc}, A_{f\mathcal{X}}) \begin{pmatrix} I & 0 \\ 0 & I \\ E_{\mathcal{X}f}(\psi) & E_{\mathcal{X}c}(\psi) \end{pmatrix}$$

so that

$$\widehat{A}_{ff} = A_{ff} + A_{f\mathcal{X}} E_{\mathcal{X}f}(\psi) \qquad \text{and} \qquad \widehat{A}_{fc} = A_{fc} + A_{f\mathcal{X}} E_{\mathcal{X}c}(\psi).$$

For any vector $[\begin{smallmatrix}\mathbf{v}_f \\ \mathbf{v}_c\end{smallmatrix}]$, consider its extension

$$\mathbf{v} = \begin{bmatrix} \mathbf{v}_f \\ \mathbf{v}_c \\ \mathbf{v}_{\mathcal{X}} \end{bmatrix},$$

where $\mathbf{v}_{\mathcal{X}}$ is given by $\mathbf{v}_{\mathcal{X}} = E_{\mathcal{X}f}(\psi)\mathbf{v}_f + E_{\mathcal{X}c}(\psi)\mathbf{v}_c$. Let

$$\widehat{\mathbf{v}} = \begin{bmatrix} -A_{ff}^{-1}\left(A_{fc}\mathbf{v}_c + A_{f\mathcal{X}}\mathbf{v}_{\mathcal{X}}\right) \\ \mathbf{v}_c \\ \mathbf{v}_{\mathcal{X}} \end{bmatrix}$$

be the so-called harmonic extension of $\mathbf{v}|_{\Omega_c(\psi)\cup\Omega_{\mathcal{X}}(\psi)}$ into $\Omega(\psi) \setminus \Omega_c(\psi)$. That is, one extends $\mathbf{v}$, restricted to the "boundary" $\Omega_c(\psi) \cup \Omega_{\mathcal{X}}(\psi)$, into the "interior" $\Omega(\psi) \setminus \Omega_c(\psi)$.

We use the $\mathbf{v}_f$ that minimizes the difference $\mathbf{v} - \widehat{\mathbf{v}}$ in energy norm in the interpolation procedure. Since

$$\mathbf{v} - \widehat{\mathbf{v}} = \begin{bmatrix} \mathbf{v}_f - (\widehat{\mathbf{v}})_f \\ 0 \\ 0 \end{bmatrix},$$

its energy norm is computable and equals

$$\begin{aligned} ||\mathbf{v} - \widehat{\mathbf{v}}||_A^2 &= (\mathbf{v}_f - (\widehat{\mathbf{v}})_f)^T A_{ff}(\mathbf{v}_f - (\widehat{\mathbf{v}})_f) \\ &= (\mathbf{v}_f + A_{ff}^{-1}\left(A_{fc}\mathbf{v}_c + A_{f\mathcal{X}}\mathbf{v}_{\mathcal{X}}\right))^T A_{ff}(\mathbf{v}_f + A_{ff}^{-1}\left(A_{fc}\mathbf{v}_c + A_{f\mathcal{X}}\mathbf{v}_{\mathcal{X}}\right)). \end{aligned}$$

Since A_{ff} is positive definite, this implies that if we solve the equation

$$\begin{aligned} 0 &= A_{ff}\mathbf{v}_f + (A_{fc}\mathbf{v}_c + A_{f\mathcal{X}}\mathbf{v}_{\mathcal{X}}) \\ &= (A_{ff} + A_{f\mathcal{X}}E_{\mathcal{X}f})\mathbf{v}_f + (A_{fc} + A_{f\mathcal{X}}E_{\mathcal{X}c})\mathbf{v}_c \\ &= \widehat{A}_{ff}\mathbf{v}_f + \widehat{A}_{fc}\mathbf{v}_c, \end{aligned}$$

the minimization of $||\mathbf{v}_f - (\widehat{\mathbf{v}})_f||_A$ is attained with zero minimum by $\mathbf{v}_f = -\widehat{A}_{ff}^{-1}\widehat{A}_{fc}\mathbf{v}_c$.

We can actually show (see Remark 7.1 and Lemma 7.1) that in the model finite element case considered in section 7 the minimization procedure is equivalent to a quadratic functional minimization involving Neumann assembled matrices, as in the AMGe method (cf. [4]).

It is natural to ask whether $\widehat{A}_{ff}$ is invertible. If $E_{\mathcal{X}f} = 0$, there is no difficulty, since then $\widehat{A}_{ff} = A_{ff}$. In general, if $E_{\mathcal{X}f}$ is sufficiently small in norm, $A_{ff} + A_{f\mathcal{X}}E_{\mathcal{X}f}$ will be invertible.

6. Examples of extension operators. We describe here four extension operators E that can be used to construct the interpolation operator in the element-free approach. These are by no means all the useful extensions that we could concoct; they form, however, a simple set of examples that will allow us to demonstrate the efficacy of the method and its underlying philosophy.

The first we call the L_2-*extension* because it is a simple averaging method. Given $\mathbf{v}$ defined on $\Omega(i)$, we wish to extend it to $\mathbf{v}_{\mathcal{X}}$, defined on $\Omega_{\mathcal{X}}(\psi)$. Suppose that $i_{\mathcal{X}}$ is an exterior dof, that is, a point from $\Omega_{\mathcal{X}}(\psi)$ whose value we wish to determine from the values of the dofs in $\Omega(i)$. Let $S = \{j \in \Omega(i) : a_{i_{\mathcal{X}},j} \neq 0\}$; that is, S comprises those dofs in $\Omega(i)$ to which the point $i_{\mathcal{X}}$ is connected. It seems natural to consider using a simple average over these dofs as the extension at $i_{\mathcal{X}}$. Thus, the extension formula, for the dof $i_{\mathcal{X}}$, is given by

$$\mathbf{v}_{\mathcal{X}}(i_{\mathcal{X}}) = \frac{1}{\sum\limits_{j\in S} 1} \sum_{j\in S} \mathbf{v}(j).$$

A somewhat more sophisticated extension we call the A-*extension* because it is a simple operator-induced method. The A-extension operator for the dof $i_{\mathcal{X}}$ is given by the formula

$$\mathbf{v}_{\mathcal{X}}(i_{\mathcal{X}}) = \frac{1}{\sum\limits_{j\in S} |a_{i_{\mathcal{X}},j}|} \sum_{j\in S} \left(|a_{i_{\mathcal{X}},j}|\mathbf{v}(j)\right).$$

It may be seen that in this case the extension to the exterior is a simple weighted average of the values of the neighborhood dofs to which the exterior point is connected. The weights in the average are given by the absolute values of the matrix coefficients.

The two methods just described share the property that they are computed point by point. That is, the extension formulas for the dofs in $\Omega_{\mathcal{X}}(\psi)$ are determined independently. A second feature shared by the methods is that if the neighborhood vector $\mathbf{v}$ is constant, then the extended values are also constant and have the same value as the entries of the neighborhood vector. This feature is clearly desirable for many elliptic PDEs, where the constant vector is in the null space or near-null space of the operator A.

The third example we describe is based on the minimization of a quadratic functional. Again, let $\mathbf{v}$ be a vector defined on $\Omega(i)$ that we wish to extend to $\Omega_{\mathcal{X}}(\psi)$. We construct the extension to be that operator which produces $\mathbf{v}_{\mathcal{X}}$ that minimizes the functional $Q(\mathbf{v}_{\mathcal{X}})$, where

$$Q(\mathbf{v}_{\mathcal{X}}) = \sum_{\substack{i_{\mathcal{X}} \in \Omega_{\mathcal{X}}(\psi) \\ j \in \Omega(i)}} |a_{i_{\mathcal{X}},j}| \left(v_{i_{\mathcal{X}}} - v_j\right)^2 . \tag{6.1}$$

It is evident that, like the previous extension operators, if $\mathbf{v}$ is constant on $\Omega(i)$ then the dofs in $\Omega_{\mathcal{X}}(\psi)$ will also have the same constant value. Unlike the previous extension operators, which are determined one dof at a time, this is a "simultaneous" extension, computing formulas for extending to all of the exterior dofs together. As such, it is necessarily more expensive to compute. We also note that this extension, and the interpolation it generates, is equivalent to the method recently proposed in [1].

A final example is given by minimizing the following "cut-off" quadratic functional:

$$(\theta \mathbf{v})^T A_{\overline{\Omega}(\psi)} (\theta \mathbf{v}) \mapsto \min,$$

where $\overline{\Omega}(\psi) \equiv \Omega(\psi) \cup \Omega_{\mathcal{X}}(\psi)$ subject to $\mathbf{v}_f,\ \mathbf{v}_c$ fixed. Here

$$\theta = \left[\begin{array}{cc} I & 0 \\ 0 & \theta_{\mathcal{X}} \end{array} \right] \begin{array}{l} \} \, \Omega(\psi), \\ \} \, \Omega_{\mathcal{X}}(\psi) \end{array}$$

is a diagonal matrix. A good choice is a diagonal matrix $\theta_{\mathcal{X}}$ formed from the vector

$$\underline{\theta}_{\mathcal{X}} = -(A_{\mathcal{X}\mathcal{X}})^{-1} \left[A_{\mathcal{X}f},\ A_{\mathcal{X}c} \right] \left[\begin{array}{c} (1)_f \\ (1)_c \end{array} \right] .$$

Here we used the blocks of A corresponding to its $\Omega_{\mathcal{X}}(\psi)$ rows.

It is easily seen that the extension mapping is actually defined as

$$\begin{aligned} \mathbf{v}_{\mathcal{X}} &= E_{\mathcal{X}c} \mathbf{v}_c + E_{\mathcal{X}f} \mathbf{v}_f \\ &= -\theta_{\mathcal{X}}^{-1} (A_{\mathcal{X}\mathcal{X}})^{-1} \left[A_{\mathcal{X}f},\ A_{\mathcal{X}c} \right] \left[\begin{array}{c} \mathbf{v}_f \\ \mathbf{v}_c \end{array} \right] . \end{aligned}$$

Note that this extension mapping is also a simultaneous extension operator and an averaging one; i.e., if $\mathbf{v}_c = (1)_c$ and $\mathbf{v}_f = (1)_f$, then $\mathbf{v}_{\mathcal{X}} = (1)_{\mathcal{X}}$.

6.1. Classical AMG as an extension method. The interpolation method of the classical AMG algorithm popularized by Ruge and Stüben [9] may be viewed as an extension method. Here the neighborhood is just the dof to be interpolated together with the dofs that will be used to compute the interpolated value. That is, $\Omega(i) = \{i\} \cup \Omega_c(i)$. The extended neighborhood then includes all fine dofs that are connected to i:

$$\Omega_{\mathcal{X}}(\psi) = \{ j \notin \Omega(i) \, : \, a_{ij} \neq 0 \} .$$

An A-extension is defined in the following manner. For each $i_{\mathcal{X}} \in \Omega_{\mathcal{X}}(\psi)$, set $v_{i_{\mathcal{X}}} = v_i$ if $i_{\mathcal{X}}$ is weakly connected to i. (Recall that in classical AMG, as developed for M-matrices, the dof i is said to be strongly connected to the dof j if

$$-a_{ij} > \theta \max_{k \neq i} (-a_{ik}),$$

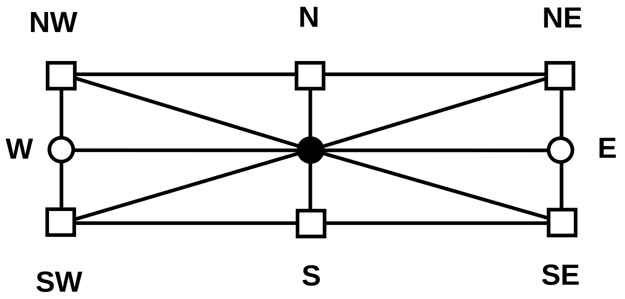


Fig. 6.1. *The neighborhood of the fine dof i (large solid circle) for the stretched quadrilateral element problem. The problem is semicoarsened; squares denote the coarse neighbors $\Omega_c(i)$ while the open circles are the exterior points $\Omega_{\mathcal{X}}(\psi)$.*

where θ is a user-specified parameter, and weakly connected otherwise.) If $i_{\mathcal{X}}$ is strongly connected to i, the extension is defined by

$$v_{i_{\mathcal{X}}} = \frac{1}{\displaystyle\sum_{j\in\Omega_c(\psi)} a_{i_{\mathcal{X}},j}} \sum_{j\in\Omega_c(\psi)} a_{i_{\mathcal{X}},j} v_j.$$

6.2. An example of the extensions. A simple example should suffice to illustrate these extension methods. Suppose the problem $-U_{xx} - U_{yy} = f(x,y)$ is discretized using a regular Cartesian grid of points making up the vertices of quadrilateral elements. Suppose further that the elements had dimension $h_x \times h_y$ where $h_x \gg h_y$. As $h_y/h_x \to 0$ the operator stencil tends toward

$$\begin{bmatrix} -1 & -4 & -1 \\ 2 & 8 & 2 \\ -1 & -4 & -1 \end{bmatrix}.$$

Since there is effectively no coupling between a given point and its neighbors to the east or west, the appropriate choice is to semicoarsen, selecting every other line of points with constant y-coordinate to be coarse points. Using the same logic, the natural interpolation is to have each fine dof interpolated using only the values to the north and south of it, each with equal weighting of 1/2. Consider the interpolation of one point, i, shown in the center of its neighborhood in Figure 6.1. For either the L_2- or A-extensions, we might select $\Omega(i) = \{i\} \cup \Omega_c(i)$, where, in this instance, $\Omega_c(i) = \{N, S, SW, NW, SE, NE\}$. Then $\Omega_{\mathcal{X}}(\psi) = \{W, E\}$. We see then that $A_{ff} = [8]$, $A_{fc} = [\ -4 \quad -4 \quad -1 \quad -1 \quad -1 \quad -1\ ]$, and $A_{f\mathcal{X}} = [\ 2 \quad 2\ ]$. For the A-extension it is easy to compute the extension operators

$$E_{\mathcal{X}c} = \frac{1}{12}\begin{pmatrix} 1 & 1 & 4 & 4 & & \\ 1 & 1 & & & 4 & 4 \end{pmatrix} \qquad \text{and} \qquad E_{\mathcal{X}f} = \frac{1}{12}\begin{pmatrix} 2 \\ 2 \end{pmatrix},$$

from which

$$\widehat{A}_{ff} = \left(\frac{104}{12}\right) \qquad \text{and} \qquad \widehat{A}_{fc} = \frac{1}{3}\left(\ -11 \quad -11 \quad -1 \quad -1 \quad -1 \quad -1\ \right),$$

which yields an interpolation operator

$$P_A = \frac{1}{26}\left(\ 11 \quad 11 \quad 1 \quad 1 \quad 1 \quad 1\ \right).$$

We see that the values to the north and south are used in the interpolation with weights $11/26 \approx 0.423$ and that the four points diagonally adjacent to i are all weighted $1/26 \approx 0.038$. The ideal weights, of course, are 0.5 and 0, respectively, so the interpolation weights computed by the A-extension method, while quite good, are not perfect.

A similar calculation for the weights using the L_2-extension yields the interpolation operator

$$P_{L_2} = \frac{1}{44}\left(\begin{array}{cccccc} 16 & 16 & 3 & 3 & 3 & 3 \end{array}\right).$$

Here the dofs to the north and south are weighted $16/44 \approx 0.364$ while the diagonally adjacent dofs are weighted by $3/44 \approx 0.068$. For this problem, then, the A-extension is significantly better than the L_2-extension.

By contrast, it is a straightforward calculation to show that classical AMG produces the interpolation operator

$$P_{AMG} = \frac{1}{12}\left(\begin{array}{cccccc} 4 & 4 & 1 & 1 & 1 & 1 \end{array}\right),$$

where the north and south dofs are weighted by $4/12 \approx 0.333$ and the diagonally adjacent dofs are weighted by $1/12 \approx 0.083$; these weights are farther from the ideal than the weights produced by either the A- or L_2-extension.

Finally, consider the extension operator based on minimizing the "cut-off" quadratic functional. The additional matrix blocks involved read

$$\begin{aligned} A_{\mathcal{X}\mathcal{X}} &= \begin{bmatrix} 8 & 0 \\ 0 & 8 \end{bmatrix}, \\ A_{\mathcal{X}f} &= \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \\ A_{\mathcal{X}c} &= \begin{bmatrix} -1 & -1 & -4 & -4 & 0 & 0 \\ -1 & -1 & 0 & 0 & -4 & -4 \end{bmatrix}. \end{aligned}$$

The vector $\underline{\theta}_{\mathcal{X}} = -A_{\mathcal{X}\mathcal{X}}^{-1}[A_{\mathcal{X}f}\ A_{\mathcal{X}c}][\begin{smallmatrix}(1)_f \\ (1)_c\end{smallmatrix}] = (1)_{\mathcal{X}}$. This is seen as follows:

$$A_{\mathcal{X}f}(1)_f = 2(1)_{\mathcal{X}}, \qquad A_{\mathcal{X}c}(1)_c = -10(1)_{\mathcal{X}},$$

and hence

$$A_{\mathcal{X}f}(1)_f + A_{\mathcal{X}c}(1)_c = -8(1)_{\mathcal{X}},$$

which implies

$$\underline{\theta}_{\mathcal{X}} = -A_{\mathcal{X}\mathcal{X}}^{-1}(A_{\mathcal{X}f}(1)_f + A_{\mathcal{X}c}(1)_c) = -\frac{1}{8}[-8(1)_{\mathcal{X}}] = (1)_{\mathcal{X}}.$$

That is, the diagonal matrix θ is the identity and hence the extension matrices then read

$$\begin{aligned} E_{\mathcal{X}f} &= -A_{\mathcal{X}\mathcal{X}}^{-1}A_{\mathcal{X}f} &&= -\tfrac{1}{4}\begin{bmatrix} 1 \\ 1 \end{bmatrix}, \\ E_{\mathcal{X}c} &= -A_{\mathcal{X}\mathcal{X}}^{-1}A_{\mathcal{X}c} &&= \tfrac{1}{8}\begin{bmatrix} 1 & 1 & 4 & 4 & 0 & 0 \\ 1 & 1 & 0 & 0 & 4 & 4 \end{bmatrix}. \end{aligned}$$

The modified matrices $\widehat{A}_{ff}$ and $\widehat{A}_{fc}$ take the form

$$\begin{aligned}
\widehat{A}_{ff} &= A_{ff} + A_{f\mathcal{X}} E_{\mathcal{X}f} &&= 8 - [2,\ 2]\tfrac{1}{4}\begin{bmatrix} 1 \\ 1 \end{bmatrix} \\
& &&= 7, \\
\widehat{A}_{fc} &= A_{fc} + A_{f\mathcal{X}} E_{\mathcal{X}c} &&= [-4,-4,-1,-1,-1,-1] + [2,\ 2]\tfrac{1}{8}\begin{bmatrix} 1 & 1 & 4 & 4 & 0 & 0 \\ 1 & 1 & 0 & 0 & 4 & 4 \end{bmatrix} \\
& &&= [-4,-4,-1,-1,-1,-1] + [\tfrac{1}{2},\tfrac{1}{2},1,1,1,1] \\
& &&= [-\tfrac{7}{2},-\tfrac{7}{2},0,0,0,0].
\end{aligned}$$

That is, the interpolation coefficients are the "perfect" ones (corresponding to semicoarsening):

$$\left((-\widehat{A}_{ff})^{-1}\widehat{A}_{fc}\right)_i = \left[\frac{1}{2},\frac{1}{2},0,0,0,0\right].$$

7. Two-grid analysis for a model finite element problem. Before providing numerical results, we present an analysis of the quality of the "element-free AMGe" interpolation. That is, we prove an "approximate" harmonic property of the interpolation mapping and show that it provides a partition of unity. Specifically, we assume that the problem is a standard finite element discretization of a second-order elliptic problem

$$a(u,\ v) \equiv \int a(x)\nabla u \cdot \nabla v \, dx = (f,\ v), \qquad v \in V,$$

where V is a finite element space of piecewise linear functions over quasi-uniform triangular elements that cover a given two-dimensional polygonal domain. For simplicity, we assume that homogeneous Neumann boundary conditions are imposed and that $(f,\ 1) = 0$ (to ensure solvability).

Let us denote, for any element e,

$$\varrho(e) = \sup_{x\in e} \max_{\underline{\xi}\in\mathbb{R}^2} \frac{\underline{\xi}^T a(x)\underline{\xi}}{\underline{\xi}^T\underline{\xi}}. \tag{7.1}$$

In the following, we assume that the differential operator coefficients are essentially constant in each element, so that $\varrho(e)$ gives rise to the local ellipticity constant.

Further, we assume (only for simplicity) that the neighborhood $\overline{\Omega}(i) \equiv \Omega(i) \cup \Omega_{\mathcal{X}}(i)$ for any fine dof i is formed by a union of triangles that share dof i as a common vertex. Thus we will use i instead of ψ to denote the neighborhoods ($\Omega(i)$, $\Omega_{\mathcal{X}}(i)$, and $\Omega_c(i)$) and the extension mappings. In particular, we denote $E_i = [E_{\mathcal{X}f},\ E_{\mathcal{X}c}]$ where for brevity $E_{\mathcal{X}f} = E_{\mathcal{X}f}(i)$ and $E_{\mathcal{X}c} = E_{\mathcal{X}c}(i)$. A closer look at the analysis to follow, however, shows that it applies as well to more general (i.e., larger) neighborhoods.

In what follows, for any subdomain (union of triangles) G, we let $a_G(.,.)$ denote the bilinear form a restricted to G. The corresponding subdomain matrix (assembled from the individual element matrices A_e) will be denoted by A_G^N. We omit the superscript N when there is no confusion between A_G^N and A_G, the submatrix of the original matrix A (corresponding to G). Note that in the latter case A_G corresponds to a matrix with homogeneous Dirichlet boundary conditions imposed on $\partial(G \cup \{\text{elements neighboring } G\})$.

For this discussion we assume that E_i, the local extension mapping used to build the interpolation coefficients, is based on averaging, although no specific rule is assumed. We do, however, assume that $E = E(i)$ has the particular form

$$\begin{bmatrix} I & 0 \\ 0 & I \\ 0 & E_{\mathcal{X}c} \end{bmatrix} \begin{matrix} \}\Omega(i), \\ \}\Omega_c(i), \\ \}\Omega_{\mathcal{X}}(i). \end{matrix}$$

That is, $E_{\mathcal{X}f} = 0$ and $E_i = [0,\ E_{\mathcal{X}c}]$.

REMARK 7.1. *The general case of $E_i = [E_{\mathcal{X}f},\ E_{\mathcal{X}c}]$ can be reduced to the particular case above by using the modified extension mapping $\widehat{E}_i = [0,\ \widehat{E}_{\mathcal{X}c}]$, where*

$$\widehat{E}_{\mathcal{X}c} = E_{\mathcal{X}f}\left(-\widehat{A}_{ff}^{-1}\widehat{A}_{fc}\right) + E_{\mathcal{X}c}.$$

To see this, recall that $\widehat{A}_{ff} = A_{ff} + A_{f\mathcal{X}}E_{\mathcal{X}f}$ and $\widehat{A}_{fc} = A_{fc} + A_{f\mathcal{X}}E_{\mathcal{X}c}$, and note that the modified extension mapping extends a constant vector defined on $\Omega_c(i)$ to be the same constant on $\Omega_{\mathcal{X}}(i)$, that is,

$$\begin{aligned} \widehat{E}_{\mathcal{X}c}(1)_c &= -E_{\mathcal{X}f}\widehat{A}_{ff}^{-1}\widehat{A}_{fc}(1)_c + E_{\mathcal{X}c}(1)_c \\ &= E_{\mathcal{X}f}(1)_f + E_{\mathcal{X}c}(1)_c \\ &= (1)_{\mathcal{X}}. \end{aligned}$$

Here we have used the fact that since (for the model second-order elliptic problem) $A_{ff}(1)_f + A_{f\mathcal{X}}(1)_{\mathcal{X}} + A_{fc}(1)_c = 0$, then $A_{ff}(1)_f + A_{f\mathcal{X}}\left(E_{\mathcal{X}f}(1)_f + E_{\mathcal{X}c}(1)_c\right) + A_{fc}(1)_c = 0$. That is, $\widehat{A}_{ff}(1)_f + \widehat{A}_{fc}(1)_c = 0$, implying that $(1)_f = -\widehat{A}_{ff}^{-1}\widehat{A}_{fc}(1)_c$.

We still must show that the modified extension mapping $\widehat{E}_i$ leads to the same interpolation as does E_i, i.e., that

$$-A_{ff}^{-1}\left(A_{f\mathcal{X}}\widehat{E}_{\mathcal{X}c} + A_{fc}\right) = -\widehat{A}_{ff}^{-1}\widehat{A}_{fc}.$$

For this we observe that

$$\begin{aligned} -A_{ff}^{-1}(A_{f\mathcal{X}}\widehat{E}_{\mathcal{X}c} + A_{fc}) &= -A_{ff}^{-1}\left[A_{f\mathcal{X}}\left(E_{\mathcal{X}f}(-\widehat{A}_{ff}^{-1}\widehat{A}_{fc}) + E_{\mathcal{X}c}\right) + A_{fc}\right] \\ &= -A_{ff}^{-1}\left[A_{f\mathcal{X}}E_{\mathcal{X}c} + A_{fc} - A_{f\mathcal{X}}E_{\mathcal{X}f}\widehat{A}_{ff}^{-1}\widehat{A}_{fc}\right] \\ &= -A_{ff}^{-1}\left[\widehat{A}_{fc} - A_{f\mathcal{X}}E_{\mathcal{X}f}\widehat{A}_{ff}^{-1}\widehat{A}_{fc}\right] \\ &= -A_{ff}^{-1}\left[\widehat{A}_{ff} - A_{f\mathcal{X}}E_{\mathcal{X}f}\right]\widehat{A}_{ff}^{-1}\widehat{A}_{fc} \\ &= -A_{ff}^{-1}\left(A_{ff}\right)\widehat{A}_{ff}^{-1}\widehat{A}_{fc} \\ &= -\widehat{A}_{ff}^{-1}\widehat{A}_{fc}. \end{aligned}$$

Consider the minimization problem

$$\text{(7.2)} \qquad \text{find } v_f \text{ such that } \begin{bmatrix} v_f \\ v_c \\ E_i v \end{bmatrix}^T A^N_{\overline{\Omega}(i)} \begin{bmatrix} v_f \\ v_c \\ E_i v \end{bmatrix} = \inf_{\substack{w:\ w_c = v_c \\ w_{\mathcal{X}} = E_i w}} a_{\overline{\Omega}(i)}(w,\ w).$$

Thus we seek v_f, the value of w on $\Omega(i) \setminus \Omega_c(i)$, which minimizes the quadratic form $a_{\overline{\Omega}(i)}(w,\ w)$ when the values of w are fixed at the coarse points and are "slave" at the exterior points ($\Omega_{\mathcal{X}}(i)$); that is, they are extrapolated from the interior $\Omega(i)$ and the coarse points $\Omega_c(i)$ by $E_i w$.

LEMMA 7.1. *The solution to the minimization problem* (7.2) *produces the same interpolation coefficients as does element-free AMGe, namely, those given by* $-A_{ff}^{-1}(A_{f\mathcal{X}}E_{\mathcal{X}c}+A_{fc})$. *That is, the minimizer is given by* $w_f = v_f \equiv -A_{ff}^{-1}(A_{f\mathcal{X}}E_{\mathcal{X}c}+A_{fc})v_c$.

Proof. Consider the Neumann matrix

$$A^N_{\overline{\Omega}(i)} = \begin{bmatrix} A_{ff} & A_{fc} & A_{f\mathcal{X}} \\ A_{cf} & A^N_{cc} & A^N_{c\mathcal{X}} \\ A_{\mathcal{X}f} & A^N_{\mathcal{X}c} & A^N_{\mathcal{X}\mathcal{X}} \end{bmatrix}.$$

We use the superscript N for the blocks which differ from the corresponding blocks of $A_{\overline{\Omega}(i)}$, the principal submatrix of the original matrix A corresponding to the subdomain $\overline{\Omega}(i)$. Note that the "N" blocks are not accessible (available) and not used in our algorithm. We have $E_i v|_{\Omega_{\mathcal{X}}(i)} = E_{\mathcal{X}c}v_c$. Hence, $a_{\overline{\Omega}(i)}(w,\ w)$ for $w_c = v_c$ and $w_{\mathcal{X}} = E_i w|_{\Omega_{\mathcal{X}}(i)}$ leads to the following matrix expression:

$$\begin{aligned} a_{\overline{\Omega}(i)}(w,\ w) &= \begin{bmatrix} w_f \\ v_c \\ E_{\mathcal{X}c}v_c \end{bmatrix}^T \begin{bmatrix} A_{ff} & A_{fc} & A_{f\mathcal{X}} \\ A_{cf} & A^N_{cc} & A^N_{c\mathcal{X}} \\ A_{\mathcal{X}f} & A^N_{\mathcal{X}c} & A^N_{\mathcal{X}\mathcal{X}} \end{bmatrix} \begin{bmatrix} w_f \\ v_c \\ E_{\mathcal{X}c}v_c \end{bmatrix} \\ &= \begin{bmatrix} w_f \\ v_c \end{bmatrix}^T \begin{bmatrix} A_{ff} & A_{fc} + A_{f\mathcal{X}}E_{\mathcal{X}c} \\ A_{cf} + E^T_{\mathcal{X}c}A_{\mathcal{X}f} & A^N_{cc} + A^N_{c\mathcal{X}}E_{\mathcal{X}c} + E^T_{\mathcal{X}c}(A^N_{\mathcal{X}c} + A^N_{\mathcal{X}\mathcal{X}}E_{\mathcal{X}c}) \end{bmatrix} \\ &\quad \times \begin{bmatrix} w_f \\ v_c \end{bmatrix}. \end{aligned}$$

Minimizing this symmetric positive semidefinite quadratic form with respect to w_f is equivalent to solving the equation

$$A_{ff}w_f + (A_{fc} + A_{f\mathcal{X}}E_{\mathcal{X}c})v_c = 0,$$

which is the same equation that specifies v_f in the element-free AMGe interpolation procedure. □

In the next lemma we will remove the constraint on v being fixed at the $\Omega_{\mathcal{X}}(i)$ points.

LEMMA 7.2. *The following quadratic forms are spectrally equivalent:*

$$q_1(v_c,\ v_c) \equiv \inf_{v:\ v|_{\Omega_c(i)}=v_c} a_{\overline{\Omega}(i)}(v,\ v), \qquad \textit{and} \qquad q_2(v_c, v_c) \equiv \inf_{\substack{v:\ v_c\ \textit{fixed} \\ v_{\mathcal{X}}=E_i v}} a_{\overline{\Omega}(i)}(v,\ v).$$

That is, there exists a positive constant η *such that*

$$q_1(v_c, v_c) \le q_2(v_c, v_c) \le \eta\, q_1(v_c, v_c) \quad \textit{for all } v_c.$$

Proof. We show the proof for two-dimensional domains. For other domains, we must scale ϱ by the local mesh size appropriately; otherwise, the proof remains the same. It suffices to show that the two quadratic forms have the same null-space. The null-space of q_1 is $v_c = \text{const}$ and the null-space of q_2 is the same as that of $a_{\overline{\Omega}(i)}(v,\ v)$

with $v:\ v_c = \text{const}$ and $E_i v = $ const on $\Omega_{\mathcal{X}}(i)$. Note that $a_{\overline{\Omega}(i)}(v,\ v) = 0$ implies v_f is the same constant as v_c. Then $E_i v$ is also the same constant, since it is an averaging operator based on the values of v_c and v_f. Hence the forms q_1 and q_2 both vanish only for constant v_c. In order to show that the constant η is bounded independently of E_i, one first easily sees that

$$a_{\overline{\Omega}(i)}(v,v) \le C \sum_{e\subset\overline{\Omega}(i)} \varrho(e) \sum_{l,\ k\in e} (v(l) - v(k))^2.$$

The constant C depends only on the number of points used in the averaging procedure (E_i), i.e., it is bounded by the total number of coarse points $\Omega_c(i)$ (plus the interior point i). The dofs l and k in the summation are either coarse dofs or i, and $\varrho(e)$ is defined in (7.1) to be the maximal value of the local ellipticity bound associated with the original elliptic operator coefficient $a(x)$. More specifically, for each $i_{\mathcal{X}} \in \Omega_{\mathcal{X}}(i)$

$$v(i_{\mathcal{X}}) \equiv (E_i v)(i_{\mathcal{X}}) = \sum_{k\in\Omega_c(i)\cup\{i\}} \alpha_{i_{\mathcal{X}},\ k} v(k),$$

where

$$\sum_{k\in\Omega_c(i)\cup\{i\}} \alpha_{i_{\mathcal{X}},\ k} = 1, \quad \text{and} \quad \alpha_{i_{\mathcal{X}},\ k} \ge 0.$$

Then, for any $j \in \Omega_c(i) \cup \{i\}$,

$$(v(i_{\mathcal{X}}) - v(j)) = \sum_{k\in\Omega_c(i)\cup\{i\}} \alpha_{i_{\mathcal{X}},\ k}(v(k) - v(j)),$$

and hence

$$\begin{aligned}(v(i_{\mathcal{X}}) - v(j))^2 \ &\le \sum_{k\in\Omega_c(i)\cup\{i\}} \alpha^2_{i_{\mathcal{X}},\ k} \sum_{k\in\Omega_c(i)\cup\{i\}} (v(k) - v(j))^2 \\ &\le \sum_{k\in\Omega_c(i)\cup\{i\}} (v(k) - v(j))^2.\end{aligned}$$

As a result we see that, for $v_{\mathcal{X}} = E_i v$,

$$\begin{aligned} q_2(v_c,\ v_c)\ &\le a_{\overline{\Omega}(i)}(v,\ v) \\ &\le C \sum_{e\subset\overline{\Omega}(i)} \varrho(e) \sum_{l,k\in e} (v(l) - v(k))^2 \\ &\le C\, \frac{\max\limits_{e\in\overline{\Omega}(i)} \varrho(e)}{\min\limits_{e\in\overline{\Omega}(i)} \varrho(e)} \sum_{e\subset\overline{\Omega}(i)} \varrho(e) \sum_{k,j\in e\cap(\Omega_c(i)\cup\{i\})} (v(k) - v(j))^2.\end{aligned}$$

Finally, since v_f is arbitrary on the right-hand side of this inequality,

$$q_2(v_c,\ v_c) \le C\, \frac{\max\limits_{e\in\overline{\Omega}(i)} \varrho(e)}{\min\limits_{e\in\overline{\Omega}(i)} \varrho(e)} \inf_{v_f} \left(\sum_{e\subset\Omega_c(i)} \varrho(e) \sum_{k,j\in e\cap(\Omega_c(i)\cup\{i\})} (v(k) - v(j))^2 \right).$$

It is also true that

$$q_1(v_c, v_c) \simeq \inf_{v:\, v_c \text{ fixed}} \left(\sum_{e \subset \overline{\Omega}(i)} \varrho(e) \sum_{l,\, k \in e} (v(l) - v(k))^2 \right).$$

This shows that η can be chosen bounded independently of the actual averaging extension mapping E_i. □

The above estimates involve the factor

$$\frac{\max\limits_{e \in \overline{\Omega}(i)} \varrho(e)}{\min\limits_{e \in \overline{\Omega}(i)} \varrho(e)}.$$

Whether it is large or small depends on the selection of the coarse grid (the coarse grid reflects the form of the neighborhood $\Omega(i)$), which we do not consider in the present paper.

Then the following corollary, involving the element-free AMGe interpolated vector Pv_c, is proved in the same way as Lemma 7.2.

COROLLARY 7.3. *Consider the extended neighborhood of i,* $\widehat{\Omega}(i) = \cup\{e,\ e \subset \overline{\Omega}(i)$ *or* $e \subset \overline{\Omega}(j)$ *for all* $j \in \Omega_{\mathcal{X}}(i)\}$. *There is a constant* $\kappa = \kappa_{\widehat{\Omega}(i)} > 0$, *locally estimated, such that the following bound holds:*

$$a_{\overline{\Omega}(i)}(Pv_c,\ Pv_c) \le \kappa \inf_{w:\ w_c = v_c} a_{\hat{\Omega}(i)}(w,\ w).$$

Proof. Let v be defined on $\widehat{\Omega}(i)$ as follows:

$$v(k) = \begin{cases} (Pv_c)(k), & k \in \overline{\Omega}(i), \\ v_c(k), & k \text{ is a coarse dof outside } \overline{\Omega}(i), \\ (E_j v)(k), & k \in \Omega_{\mathcal{X}}(j), \text{ for some } j \in \Omega_{\mathcal{X}}(i). \end{cases}$$

We see that v at every fine dof k in $\widehat{\Omega}(i)$ is an average value of some neighboring coarse dofs from $\widehat{\Omega}(i)$. Hence, in the same way as in the proof of Lemma 7.2, we establish the inequality

$$a_{\hat{\Omega}(i)}(v, v) \le \kappa \inf_{w:\ w_c = v_c} a_{\hat{\Omega}(i)}(w,\ w).$$

Since $a_{\overline{\Omega}(i)}(Pv_c,\ Pv_c) \le a_{\hat{\Omega}(i)}(v, v)$, the desired result follows. □

For each fine dof i, define $\mathcal{Z}(i)$ to be the number of overlapping domains on $\hat{\Omega}(i)$, that is, the number of domains $\hat{\Omega}(j)$ such that $\hat{\Omega}(j) \cap \hat{\Omega}(i) \ne \emptyset$. Then we may state the following theorem.

THEOREM 7.4. *The element-free interpolation mapping P exhibits the following approximate harmonic property:*

$$a(Pv_c,\ Pv_c) \le \kappa \inf_{w:\ w_c = v_c} a(w,\ w),$$

where the constant $\kappa = \max\limits_{i = \text{fine dof}} \kappa_{\hat{\Omega}(i)} \mathcal{Z}(i)$, *and the* $\kappa_{\hat{\Omega}(i)}$ *are the local constants from Corollary* 7.3.

Proof. The proof simply follows from the fact that

$$a(Pv_c,\ Pv_c) \le \sum_{i = \text{fine dof}} a_{\overline{\Omega}(i)}(Pv_c,\ Pv_c)$$

and by summation of the local estimates from Corollary 7.3. □

Another important property of the element-free interpolation mapping P is that it partitions unity, as we show in the following theorem.

THEOREM 7.5. *P provides a partition of unity. Specifically, the row sums of P are* 1.

Proof. Let $v_f = Pv_c$ be given by $v_f = \sum_{i_c \in \Omega_c(i)} \alpha_{i,\, i_c} v_c(i_c)$. Assume that $v_c(i_c) = 1$ on $\Omega_c(i)$. Now, P uses the formula that minimizes (7.2) and the minimum (zero) is achieved for $E_i v(j) = 1$ at $\Omega_{\mathcal{X}}(i)$ and $v_f = 1$. That is, we find that

$$1 = \sum_{i_c \in \Omega_c(i)} \alpha_{i,\, i_c},$$

which is the desired unity row-sum property of P. □

REMARK 7.2. *Theorems* 7.4 *and* 7.5 *are the main goals of many two-grid convergence analyses and they imply convergence of the respective two-grid AMG methods, cf., e.g.,* [14], [8], [13], *and* [7].

8. Numerical experiments. We describe here several sets of numerical experiments designed to test the efficacy of the element-free AMGe methods described above. For each of several problems, we apply a set of interpolation rules within an AMG code. The problems are then solved using a CG solver, preconditioned with one V-cycle of AMG.

The interpolation rules are

- the AMGe rule [7] for the finite element problems;
- three element-free AMGe rules from section 6:
  1. L_2-extension;
  2. A-extension;
  3. (only for scalar PDE) the simultaneous extension based on minimizing the quadratic functional (6.1) described in section 6.

For system problems the unknowns are split into physical variables. That is, for scalar problems the rule is as described in section 6, while for two-dimensional elasticity, with physical variables u and v (displacement in the x- and y-directions, respectively), we perform the extensions (and associated interpolation) of exterior dofs of type u using only neighborhood dofs of type u; similarly, the extension to exterior dofs of type v are carried out using neighborhood dofs of type v; this applies both to L_2- and A-extensions. The local neighborhood about a point is defined by the sparsity pattern of the matrix about that point, and the averaging involves only dofs from the sparsity pattern set S (see section 6).

8.1. An elliptic problem on a triangular element mesh. We apply the various interpolation rules to a second-order elliptic PDE

$$-\nabla \cdot (A(x,y)\nabla u) = f(x,y) \quad \text{on } G, \tag{8.1}$$

$$u(x,y) = g(x,y) \quad \text{on } \partial G, \tag{8.2}$$

where G is the unit square. The matrix of diffusion coefficients includes functions with relatively benign characteristics—there are both spatial variability and jump discontinuity in the coefficients, but the jumps are of relatively small magnitude and the variation is mild. The discretization is by a finite element method on an unstructured triangular mesh. The coarsening algorithm is one of element agglomeration. That is, the coarse grids are the vertices of coarse elements produced by an agglomeration algorithm proposed in [7]. Figure 8.1 displays the coarsening sequence for a typical

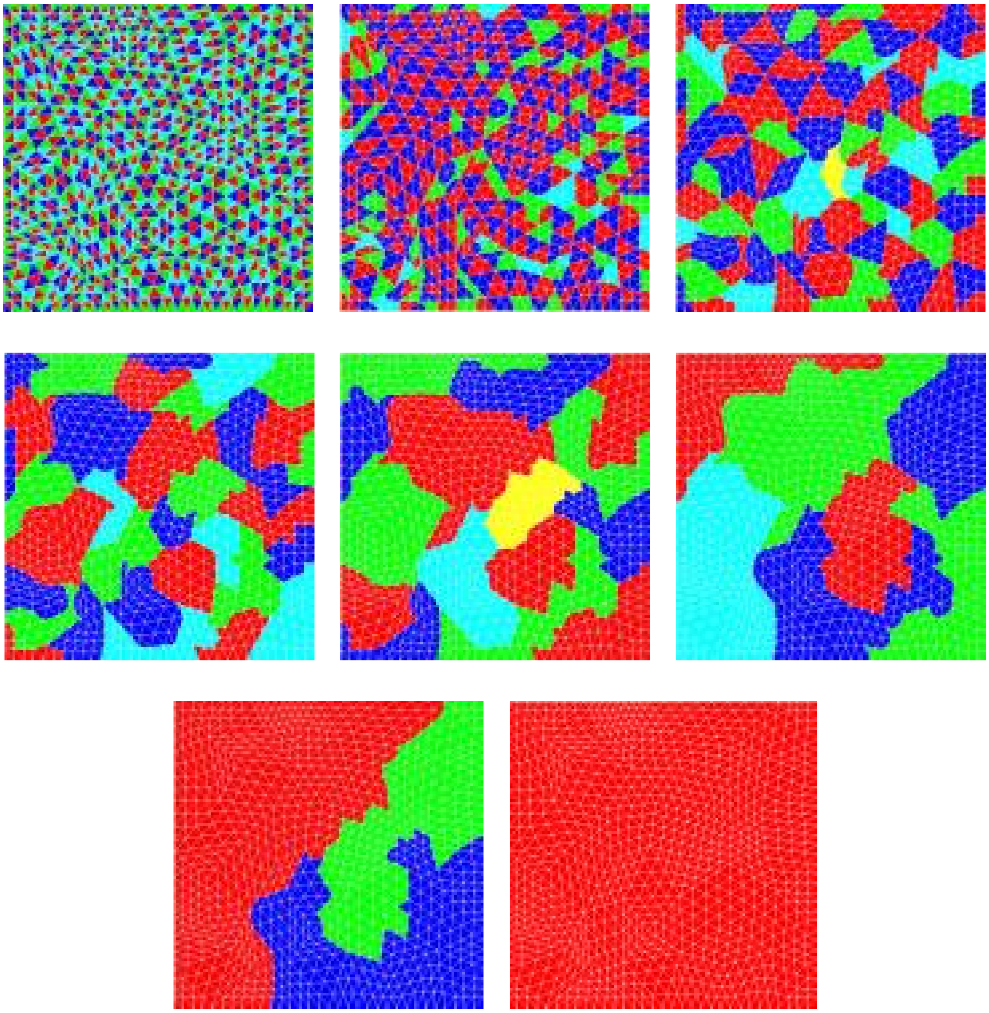


Fig. 8.1. *Sequence increasingly coarse elements, formed by element agglomeration.*

problem. Here the fine grid comprises 1600 elements, the first coarse grid has 382 elements, and the remaining grids have 93, 33, 15, 7, 3, and 1 elements. Table 8.1 gives the coarsening details for four different versions of this problem. It may be seen that the number of elements decreases by about 75% at each coarsening for the first few coarsenings, after which it decreases by about 50% per level. The number of nonzero entries in the matrix decreases by approximately 50% per level, while the number of dofs tends to decrease by 50–60% with each successive level.

For each of the four interpolation rules, the problem is solved using a preconditioned CG method, where the preconditioning consists of a single V(1,1)-cycle of AMG, with a Gauss–Seidel smoother. The iteration is run until the residual is less than 10^{-8} in norm. We report the results in Table 8.2. For each problem size, we display, for each interpolation rule, the number of preconditioned CG iterations required to achieve the desired residual size and ϱ, the average convergence factor over the iterations.

Table 8.1

Coarsening history for the problem $-\nabla \cdot A(x.y)\nabla u = f$ on an unstructured triangular fine grid. For each level of each problem size, "nz" is the number of nonzero entries in the operator matrix, "dofs" gives the number of degrees of freedom, and "elts" gives the number of finite elements in the agglomerated grid.

		No. of elements			
Level		25600	6400	1600	400
0	nz	90321	22761	5781	1491
	dofs	13041	3321	861	231
	elts	25600	6400	1600	400
1	nz	32898	9540	2602	1094
	dofs	4108	1152	330	114
	elts	6013	1427	382	76
2	nz	14305	4361	1397	470
	dofs	1507	451	143	50
	elts	1489	374	93	26
3	nz	7193	2098	634	199
	dofs	643	198	64	23
	elts	392	117	33	11
4	nz	3458	975	304	88
	dofs	302	91	32	12
	elts	158	47	15	5
5	nz	1580	453	126	36
	dofs	140	45	16	6
	elts	70	22	7	2
6	nz	714	188	46	16
	dofs	68	22	8	4
	elts	33	10	3	1
7	nz	274	84	16	
	dofs	30	12	4	
	elts	14	5	1	
8	nz	120	30		
	dofs	16	6		
	elts	7	2		
9	nz	42	16		
	dofs	8	4		
	elts	3	1		
10	nz	16			
	dofs	4			
	elts	1			

Table 8.2

CG convergence results; unstructured triangular fine grid; second-order elliptic problem; $V(1,1)$-cycle MG, Gauss–Seidel smoother used as preconditioner.

Interp. rule		400 elts	1600 elts	6400 elts	25600 elts
AMGe	iterations	14	16	21	23
	ϱ	0.115	0.172	0.252	0.289
A-extension	iterations	13	15	19	20
	ϱ	0.118	0.158	0.218	0.247
L_2-extension	iterations	13	16	19	21
	ϱ	0.119	0.161	0.227	0.249
Quadratic funct.	iterations	13	15	19	19
min.	ϱ	0.105	0.152	0.222	0.231

Examination of the results reveals that all three of the extension methods, A-extension, L_2-extension, and quadratic functional minimization, perform at least as well on this problem as does AMGe. In some cases the performance of the extension methods is marginally better than AMGe. The amount of work entailed for the A-extension and the L_2-extension methods is comparable to that of AMGe, provided that the neighborhoods are selected to be of comparable size to the element neighborhoods (which is the case in these experiments). For the quadratic functional minimization the work is somewhat greater but still comparable. The advantage of the element-free methods is, of course, that there is no requirement to have the actual individual stiffness matrices that are required in AMGe. For this experiment this represents a considerable savings in storage.

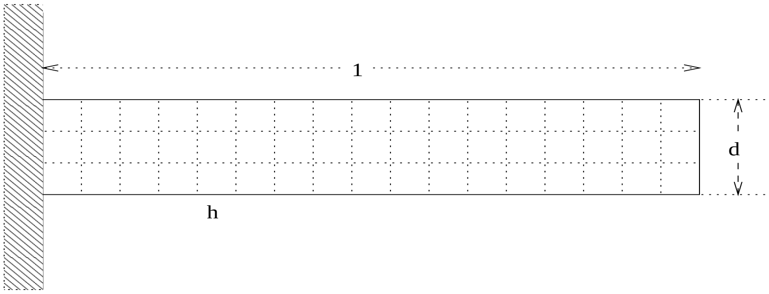


FIG. 8.2. *The thin-beam elasticity problem domain. Homogeneous Dirichlet boundary conditions are applied at $x = 0$.*

8.2. Two-dimensional elasticity: The thin beam. We consider next the two-dimensional plane-stress elasticity problem on a cantilevered beam, fixed at one end (see Figure 8.2). The domain of the problem is $G = (0,1) \times (0,\ d)$ with $d \le 1$. For $d \ll 1$ this is the thin beam problem. The problem is

$$\begin{aligned} u_{xx} + \frac{1-\nu}{2}\, u_{yy} + \frac{1+\nu}{2}\, v_{xy} &= f_1, \\ \frac{1+\nu}{2}\, u_{xy} + \frac{1-\nu}{2}\, v_{xx} + v_{yy} &= f_2, \end{aligned}$$

where u and v are displacements in the x and y directions, respectively. This can be a difficult problem for standard multigrid methods, especially when the domain is long and thin [6]. The problem is discretized using uniform square finite elements of size h. Nodal coarsening is used, with the coarse nodes being the vertices of elements created by the agglomeration algorithm from [7]. After certain levels of coarsening the algorithm agglomerates only along the x direction.

We present results in both the thick beam ($d = 1.0$) and thin beam ($d = 0.05$) cases. In both cases we use $\nu = 1/3$. For each case we present results for three sizes of the discretization parameter: $h = 0.05, 0.025$, and 0.0125 for the thick beam and $h = 0.025, 0.0125$, and 0.00625 for the thin beam. The coarsening histories of the agglomeration algorithm are shown in Table 8.3. Table 8.4 shows the results of the experiments for the beam problem. As in section 8.1, preconditioned CG is used as the solver, with a single V(1,1)-cycle of AMG as the preconditioner, with a Gauss–Seidel smoother. For this problem we show the number of iterations required to achieve a residual norm less than 10^{-8}, and also the convergence factor of the final iteration. For this problem we do not implement the quadratic minimization method

Table 8.3
Coarsening history; structured rectangular fine grid; two-dimensional elasticity, $d = 1$.

Level		Thick beam $d = 1.0$			Thin beam $d = 0.05$		
		$h = 0.050$	$h = 0.025$	$h = 0.0125$	$h = 0.025$	$h = 0.0125$	$h = 0.00625$
0	nz	14884	58564	232324	3388	12532	48100
	dofs	882	3362	13122	246	810	2898
1	nz	10440	40880	161760	1664	7328	30656
	dofs	264	924	3444	88	252	820
2	nz	4128	17248	70488	784	3744	10152
	dofs	84	264	924	44	132	252
3	nz	1000	4956	19056	384	1152	3816
	dofs	32	94	284	24	48	132
4	nz	256	1404	6128	144	384	1152
	dofs	16	38	104	12	24	48
5	nz	64	324	1668	64	144	384
	dofs	8	18	42	8	12	24
6	nz		144	576		64	144
	dofs		12	24		8	12
7	nz		64	144			64
	dofs		8	12			8
8	nz			64			
	dofs			8			

Table 8.4
CG convergence results; structured rectangular fine grid; two-dimensional elasticity, $d = 1$, $V(1,1)$-cycle MG, Gauss–Seidel smoother used as preconditioner.

Thick beam, $d = 1.0$				
Interp. rule		$h = 0.050$	$h = 0.025$	$h = 0.0125$
AMGe	iterations ϱ	16 0.172	18 0.206	20 0.234
A-extension	iterations ϱ	12 0.099	12 0.098	12 0.097
L_2-extension	iterations ϱ	13 0.101	13 0.102	13 0.104

Thin beam, $d = 0.05$				
Interp. rule		$h = 0.025$	$h = 0.0125$	$h = 0.00625$
AMGe	iterations ϱ	17 0.180	18 0.198	19 0.22
A-extension	iterations ϱ	20 0.227	23 0.286	22 0.280
L_2-extension	iterations ϱ	18 0.203	20 0.243	27 0.254

described in section 6. That method is for scalar problems, while this problem is a system of PDEs. We use the AMGe method described in [4] and compare it with the A- and L_2-extension methods described above. Our expectation is that AMGe should outperform the element-free methods, at least on the thin beam problem; this is the problem for which AMGe was originally developed. We observe, however, that for the thick beam problems the element-free methods both outperform AMGe. First, we note that it takes fewer iterations to reach the tolerance. It is also apparent that the element-free methods are more scalable, in that the number of iterations does not grow with the problem size. The AMGe method requires more iterations for larger problems.

For the thin beam problem, we observe the results we naturally expect. That is, AMGe outperforms the element-free methods, requiring fewer iterations. Further, AMGe appears to be more scalable on this problem than the extension methods. The L_2-extension method exhibits a distinct lack of scalability as the problem grows larger.

9. Conclusions. In this paper we propose a general rule for building interpolation weights in AMG, thus extending the applicability of AMG to more general settings than the traditional M-matrix case. The applications include elliptic problems on unstructured finite element grids, where both scalar problems and systems (like elasticity) are considered. The element-free AMGe method seems as competitive as the AMGe methods but entails much less overhead. The element information and the element matrices, in particular, are essential for the AMGe methods but are not required for element-free AMGe. If we assume more information is available (such as the rigid body modes in the case of elasticity) it may be incorporated into the construction of the extension mappings. Thus element-free AMGe can be made to reproduce the extra modes in the interpolation from their coarse values. This property is important in the AMG methods for elasticity problems (cf. [12]), and incorporating it into element-free AMGe is a subject of ongoing research.

REFERENCES

[1] A. Brandt, *Generally highly accurate algebraic coarsening*, Electron. Trans. Numer. Anal., 10 (2000), pp. 1–20.

[2] A. Brandt, S. F. McCormick, and J. W. Ruge, *Algebraic Multigrid (AMG) for Automatic Multigrid Solutions with Application to Geodetic Computations*, Report, Inst. for Computational Studies, Fort Collins, CO, 1982.

[3] A. Brandt, S. F. McCormick, and J. W. Ruge, *Algebraic multigrid (AMG) for sparse matrix equations*, in Sparsity and Its Applications, D. J. Evans, ed., Cambridge University Press, Cambridge, UK, 1985, pp. 257–284.

[4] M. Brezina, A. J. Cleary, R. D. Falgout, V. E. Henson, J. E. Jones, T. A. Manteuffel, S. F. McCormick, and J. W. Ruge, *Algebraic multigrid based on element interpolation (AMGe)*, SIAM J. Sci. Comput., 22 (2000), pp. 1570–1592.

[5] A. J. Cleary, R. D. Falgout, V. E. Henson, and J. E. Jones, *Coarse-grid selection for parallel algebraic multigrid*, in Proceedings of the Fifth International Symposium on Solving Irregularly Structured Problems in Parallel, Lecture Notes in Comput. Sci. 1457, Springer-Verlag, New York, 1998, pp. 104–115.

[6] A. J. Cleary, R. D. Falgout, V. E. Henson, J. E. Jones, T. A. Manteuffel, S. F. McCormick, G. N. Miranda, and J. W. Ruge, *Robustness and scalability of algebraic multigrid*, SIAM J. Sci. Comput., 21 (2000), pp. 1886–1908.

[7] J. E. Jones and P. S. Vassilevski, *AMGe based on element agglomeration*, SIAM J. Sci. Comput., 23 (2001), pp. 109–133.

[8] J. Mandel, M. Brezina, and P. Vaněk, *Energy optimization of algebraic multigrid bases*, Computing, 62 (1999), pp. 205–228.

[9] J. W. Ruge and K. Stüben, *Algebraic multigrid (AMG)*, in Multigrid Methods, Frontiers Appl. Math. 3, S. F. McCormick, ed., SIAM, Philadelphia, 1987, pp. 73–130.

[10] K. Stüben, *Algebraic multigrid (AMG): Experiences and comparisons*, Appl. Math. Comput., 13 (1983), pp. 419–452.

[11] K. Stüben, *A Review of Algebraic Multigrid*, GMD report 69, Sankt Augustin, Germany, 1999.

[12] P. Vanĕ k, M. Brezina, and R. Tezuar, *Two-grid method for linear elasticity on unstructured meshes*, SIAM J. Sci. Comput., 21 (1999), pp. 900–923.

[13] P. Vaněk, J. Mandel, and M. Brezina, *Convergence of Algebraic Multigrid Based on Smoothed Aggregation*, UCD/CCM report 126, Center for Computational Mathematics, University of Colorado at Denver, Denver, CO, 1998; also available from http://www-math.cudenver.edu/ccmreports/rep126.ps.gz.

[14] W. L. Wan, T. F. Chan, and B. Smith, *An energy-minimizing interpolation for robust multigrid methods*, SIAM J. Sci. Comput., 21 (2000), pp. 1632–1649.

SIAM J. Sci. Comput.
Vol. 23, No. 2, pp. 651–671

ON THREE-GRID FOURIER ANALYSIS FOR MULTIGRID[*]

ROMAN WIENANDS[†] AND CORNELIS W. OOSTERLEE[†]

Abstract. In this paper, we present three-grid Fourier analysis for multigrid methods. Due to the recursive structure of a multigrid iteration, this analysis can be deduced from the well-known two-grid Fourier analysis. The coarse grid correction part of multigrid algorithms can be more accurately evaluated with the three-grid analysis. We apply the analysis to several scalar equations and discretizations with an emphasis on problems with a multigrid coarse grid correction difficulty like upwind discretizations of the convection diffusion equation. The main focus lies on possible improvements by carefully chosen Galerkin operators and/or by an additional acceleration with restarted GMRES, GMRES(m). Numerical test calculations validate the theoretical predictions.

Key words. Fourier analysis, multigrid, coarse grid correction, restarted GMRES

AMS subject classifications. 65N55, 65N12, 65F10

PII. S106482750037367X

1. Introduction.

Fourier one-grid (smoothing) and two-grid analysis [2, 20, 21] are well-known tools in the multigrid community. The two-grid analysis is the basis for classical asymptotic multigrid convergence estimates [20, 21]. Moreover, it is the main analysis tool for nonsymmetric problems.

For several multigrid components or cycle variants, however, the asymptotic multigrid convergence factor cannot be predicted accurately by the Fourier two-grid factors. For example, one may use different discretizations on different grids. It can be beneficial to replace the direct $2h$-, $4h$-, etc. discretizations on the coarser grids by other discretizations. The most prominent example of this kind is the Galerkin coarse grid operator. As the entries of the Galerkin coarse grid discretizations are in general not known in advance, they may not be favorable for the smoothing method applied. Investigations of the two-grid iteration cannot display possible smoothing difficulties on coarser grids induced by the different discretizations, since the direct solution of the $2h$-problem is assumed. Furthermore, if one is interested in the influence on the asymptotic convergence factor of V-cycles versus W-cycles, of different numbers of pre- and postsmoothing, or of different smoothers on different grids, one needs to consider at least three grids.

To investigate these additional phenomena we carry out a *three*-grid Fourier analysis, which is usually sufficient to obtain a comprehensive insight into a *multi*grid method. In section 2, we outline the theoretical background of the three-grid analysis. Instead of (4×4)-blocks for the two-grid iteration matrix [2, 20, 21, 24] in the case of standard grid coarsening, the three-grid iteration matrix is transformed into a (16×16)-block matrix by Fourier analysis in the two-dimensional scalar case. This means that the calculation of three-grid asymptotic convergence factors is reduced to the calculation of the spectral radii of certain (16×16)-matrices.

In section 3, we apply the three-grid analysis to several equations and multigrid components. We focus on singular perturbation problems like the convection diffusion equation discretized by first or higher order difference schemes and the rotated

[*] Received by the editors June 13, 2000; accepted for publication (in revised form) November 29, 2000; published electronically July 10, 2001.
http://www.siam.org/journals/sisc/23-2/37367.html

[†] GMD–Institute for Algorithms and Scientific Computing (SCAI), D-53754 Saint Augustin, Germany (wienands@gmd.de, oosterlee@gmd.de).

anisotropic diffusion equation. For these problems a variety of nonstandard coarse grid discretizations is evaluated. Some of them are discussed in [26]. For example, Galerkin coarsenings based on the transfer operators from [7] and [27] are analyzed.

This analysis can be generalized to the situation where multigrid is a preconditioner for GMRES(m) [15] in the same way as it was done in [24] for the two-grid analysis. The different multigrid algorithms are not only evaluated as a solver but also as a preconditioner for GMRES(m).

2. Fourier analysis of multigrid.

In this section we introduce the three-grid Fourier analysis of multigrid as a solver and as a preconditioner for GMRES(m); see sections 2.3 and 2.4, respectively. We restrict ourselves to the two-dimensional scalar case and standard coarsening, i.e., the grid coarsening is performed by doubling the mesh size in each direction, in order to keep the presentation as simple as possible. However, the generalization to three dimensions or to systems of equations is obvious but somewhat more technically involved, as explained in Remark 2. Other coarsening techniques like semicoarsening can be treated in a similar way by some appropriate changes concerning the coarse grid correction in the multigrid process.

2.1. Notation and basic principles.

The rigorous theoretical foundations for the Fourier analysis of multigrid, which is also commonly called *local mode analysis*, can be found, for example, in [4] and [18]. Basically, the local mode analysis is valid if the influence of a domain boundary is negligible [4]. Often, this requirement can be fulfilled by performing some extra local relaxations near and at the boundary.

For a k-grid cycle, we consider k discrete linear operators L_n $(n = 1, \dots, k)$ with constant coefficients on k infinite grids G_n with mesh sizes $h_n = 2^{k-n}h$:

$$\begin{gathered} L_n u_n(\boldsymbol{x}) = f_n(\boldsymbol{x}) \\ \text{on } \; G_n := \{\boldsymbol{x} = h_n \boldsymbol{j} = (h_n j_x, h_n j_y) = (x, y) \; \text{ with } \; \boldsymbol{j} \in \mathbb{Z}^2\}\,. \end{gathered} \tag{2.1}$$

Obviously, the grids become finer with an increasing index n, and L_k is defined on the finest grid with mesh size $h_k = h$. In stencil notation [20], (2.1) looks like

$$L_n u_n(\boldsymbol{x}) = \sum_{\boldsymbol{\kappa} \in J} (l_n)_{\boldsymbol{\kappa}} u_n(\boldsymbol{x} + \boldsymbol{\kappa} h_n) = f_n(\boldsymbol{x}) \;\text{ on }\; G_n \tag{2.2}$$

with stencil coefficients $(l_n)_{\boldsymbol{\kappa}}$ and a finite index set $J \subset \mathbb{Z}^2$. For compact 9-point stencils $[L]$ we have, for example,

$$J := \{\boldsymbol{\kappa} = (\kappa_x, \kappa_y) \;\text{ with }\; \kappa_x, \kappa_y \in \{-1, 0, 1\}\,\} \quad \text{and} \quad [L] = \begin{bmatrix} l_{-1,1} & l_{0,1} & l_{1,1} \\ l_{-1,0} & l_{0,0} & l_{1,0} \\ l_{-1,-1} & l_{0,-1} & l_{1,-1} \end{bmatrix}.$$

From (2.2), it can be deduced that the continuous eigenfunctions, the *Fourier components*, of the fine grid operator L_k are given by

$$\phi(\boldsymbol{\theta}, \boldsymbol{x}) := e^{i\boldsymbol{x}\boldsymbol{\theta}/h} = e^{i\boldsymbol{j}\boldsymbol{\theta}} = e^{i(j_x\theta_x + j_y\theta_y)} \;\text{ with }\; \boldsymbol{x} \in G_k\,,$$

where the *Fourier frequencies* $\boldsymbol{\theta} = (\theta_x, \theta_y)$ vary continuously in $\mathbb{R}^2$. The corresponding eigenvalues or *Fourier symbols* of L_k read as

$$\widetilde{L}_k(\boldsymbol{\theta}) = \sum_{\boldsymbol{\kappa} \in J} (l_k)_{\boldsymbol{\kappa}} e^{i\boldsymbol{\theta}\boldsymbol{\kappa}}\,. \tag{2.3}$$

On G_k, we introduce the scaled Euclidean inner product [20]

$$\langle v_k, w_k \rangle := \lim_{m\to\infty} \frac{1}{4m^2} \sum_{|\boldsymbol{\kappa}|\le m} v_k(\boldsymbol{\kappa}h)\,\overline{w_k(\boldsymbol{\kappa}h)} \quad \text{with}$$
$$|\boldsymbol{\kappa}| = \max\{|\kappa_1|, |\kappa_2|\} \quad \text{and} \quad v_k, w_k : \; G_k \longrightarrow \mathbb{C},$$

leading to a norm $||v_k|| := \sqrt{\langle v_k, v_k\rangle}$. Note that the Fourier components are orthonormal with respect to this inner product [20]. We define the space of bounded infinite grid functions by

$$\mathcal{F}(G_k) := \{v_k \mid \; v_k(\,.\,) : \; G_k \longrightarrow \mathbb{C} \; \text{ with } \; ||v_k|| < \infty\}.$$

For each $v_k \in \mathcal{F}(G_k)$, there exists a Fourier transformation, i.e., each v_k can be written as a linear combination of Fourier components [4, 11, 23]. Fourier components with $|\hat{\boldsymbol{\theta}}| := \max\{|\hat{\theta_x}|, |\hat{\theta_y}|\} \ge \pi$ are not visible on G_k, since they coincide with components $e^{i\boldsymbol{j}\boldsymbol{\theta}}$, where $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}(\operatorname{mod} 2\pi)$, due to the periodicity of the exponential function. Therefore, the *Fourier space*

$$\mathcal{F} := \operatorname{span}\{e^{i\boldsymbol{j}\boldsymbol{\theta}} : \boldsymbol{\theta} \in \Theta = (-\pi, \pi]^2\} \tag{2.4}$$

contains any bounded infinite grid-function.

It is convenient to explain the three-grid analysis (or, more generally, the k-grid analysis) by a recursive adaptation of the two-grid case. The discrete fine grid solution u_k and a current approximation u^i can be represented by linear combinations of Fourier components $e^{i\boldsymbol{j}\boldsymbol{\theta}} \in \mathcal{F}$ because of $\mathcal{F}(G_k) \subset \mathcal{F}$. The same holds for the error $v^{i-1} = u^{i-1} - u_k$ before and $v^i = u^i - u_k$ after the ith k-grid cycle. It can be easily established by induction that the error transformation by a k-grid cycle is given by the following recursion [10, 20, 21]:

$$M_2^1 = S_2^{\nu_2} K_2^1 S_2^{\nu_1} = S_2^{\nu_2}(I_2 - P_1^2 (L_1)^{-1} R_2^1 L_2) S_2^{\nu_1}, \tag{2.5}$$

$$\begin{aligned} M_{\ell+2}^1 &= S_{\ell+2}^{\nu_2} K_{\ell+2}^1 S_{\ell+2}^{\nu_1} \\ &= S_{\ell+2}^{\nu_2}(I_{\ell+2} - P_{\ell+1}^{\ell+2}(I_{\ell+1} - (M_{\ell+1}^1)^\gamma)(L_{\ell+1})^{-1} R_{\ell+2}^{\ell+1} L_{\ell+2}) S_{\ell+2}^{\nu_1} \\ &\text{for} \quad \ell = 1, \dots, k-2\,, \end{aligned} \tag{2.6}$$

where the sub- and superscripts of the different operators are abbreviations for the related mesh sizes of the k involved grids. S_n is a smoothing operator on G_n, ν_1 and ν_2 indicate the number of pre- and postsmoothing iterations, K_n^1 is the coarse grid correction operator, I_n is the G_n-identity, L_n is the approximation of L_k on a coarse grid G_n, P_{n-1}^n and R_n^{n-1} are transfer operators from coarse to fine grids and reversed, and γ is the cycle index (for example, $\gamma = 1$ denotes a V-cycle and $\gamma = 2$ denotes a W-cycle). Of course, it is possible to vary the number of pre- and postsmoothing steps on the different grids leading to $\nu_1(n)$ and $\nu_2(n)$. L_{n-1} may be defined by Galerkin coarsening, $L_{n-1} = \check{R}_n^{n-1} L_n \check{P}_{n-1}^n$, or simply by a straightforward application of L_k on G_{n-1}. Note that the transfer operators in the Galerkin process do not necessarily have to match with R_n^{n-1} and P_{n-1}^n from the multigrid iteration.

For the coarse grid discretization operators, the prolongation, and the restriction, similar stencil notations as in (2.2) exist. The corresponding Fourier symbols, denoted by a tilde $\tilde{}$, are calculated as in (2.3) with a suitable index set J related to the operator under consideration; see (2.8). A detailed representation of the Fourier symbols for many prolongations, restrictions, and discretizations is given, for instance, in [10, 20, 21, 23].

In the following, we often use a more instructive notation for the operators from (2.5), (2.6), the related Fourier symbols, and the infinite grids G_n. For example, their indices are replaced by the corresponding mesh sizes $h_n = 2^{k-n}h$. For example, we write P^h_{2h}, $\widetilde{P}^h_{2h}(\boldsymbol{\theta})$, and G_h instead of P^k_{k-1}, $\widetilde{P}^k_{k-1}(\boldsymbol{\theta})$, and G_k, respectively; see, e.g., (2.7), (2.8), or (2.11). Some basic elements of the three-grid analysis already appear in the two-grid analysis. Here, the sub- or superscript $2g$ always stands for two-grid. The index $3g$ is used accordingly for three-grid in section 2.3.

2.2. Two-grid Fourier analysis. If standard coarsening is selected, it is convenient to divide the Fourier space (2.4) into the following four-dimensional subspaces.

DEFINITION 2.1 ($2h$-harmonics). *The $2h$-harmonics $\mathcal{F}^{2g}_{\boldsymbol{\theta}}$ are given by*

$$\mathcal{F}^{2g}_{\boldsymbol{\theta}} := span\{\phi(\boldsymbol{\theta}^{\alpha_x\alpha_y}, \boldsymbol{x}) \quad with\ \ \alpha_x, \alpha_y \in \{0,1\}\ \}, \quad where$$
$$\boldsymbol{\theta} = \boldsymbol{\theta}^{00} \in \Theta_{2g} := (-\pi/2, \pi/2]^2 \quad and\ \ \boldsymbol{\theta}^{\alpha_x\alpha_y} := \boldsymbol{\theta}^{00} - (\,\alpha_x sign(\theta_x), \alpha_y sign(\theta_y)\,)\,\pi\,.$$

This distinction is motivated by the fact that each *low-frequency* $\boldsymbol{\theta}^{00} \in \Theta_{2g}$ is coupled with three *high-frequencies* $\boldsymbol{\theta}^{\boldsymbol{\alpha}}$ with $\boldsymbol{\alpha} \neq (00)$ in the transition from G_h to G_{2h}. For example, the three high-frequency components are not visible on the coarse grid as they coincide with the corresponding low-frequency component.

In order to ensure that we deal with nonsingular Fourier symbols $\widetilde{L}_h(\boldsymbol{\theta})$ and $\widetilde{L}_{2h}(2\boldsymbol{\theta})$, we restrict our considerations to the following slightly shrunken subspace of the Fourier space (2.4), as in [20]:

$$\mathcal{F}^{2g} := \mathcal{F}\backslash \bigcup_{\boldsymbol{\theta}\in\Psi_{2g}} \mathcal{F}^{2g}_{\boldsymbol{\theta}} \quad \text{with} \quad \Psi_{2g} := \{\boldsymbol{\theta} \in \Theta_{2g} :\ \widetilde{L}_{2h}(2\boldsymbol{\theta}^{00}) = 0 \ \text{ or } \ \widetilde{L}_h(\boldsymbol{\theta}^{\boldsymbol{\alpha}}) = 0\}.$$

The crucial observation is that the coarse grid correction operator K^{2h}_h (see (2.5)) leaves the spaces of $2h$-harmonics invariant for an arbitrary Fourier frequency $\boldsymbol{\theta} \in \widetilde{\Theta}_{2g} := \Theta_{2g}\backslash\Psi_{2g}$. The same invariance property holds for many well-known smoothing methods, e.g., Jacobi point- or line-relaxation, lexicographical Gauss–Seidel point- or line-relaxation, and certain pattern relaxation methods such as red-black Gauss–Seidel or zebra line Gauss–Seidel. Especially for pattern relaxations, the calculation of the related Fourier symbols is not as simple as it is for the different operators of the coarse grid correction. In general, certain Fourier components within the spaces of $2h$-harmonics may be coupled by the smoothing operator, which means that the calculation of the corresponding Fourier symbol cannot be done separately for each component $\phi(\boldsymbol{\theta}, \boldsymbol{x})$, as in (2.3). On the contrary, it is represented by a general (4×4)-matrix $S^{2g}(\boldsymbol{\theta}, h) = S^{2g}(\boldsymbol{\theta}^{00}, \boldsymbol{\theta}^{11}, \boldsymbol{\theta}^{10}, \boldsymbol{\theta}^{01}, h) \in \mathbb{C}^{4\times 4}$. For the explicit representation of several relaxation methods, we refer to [10, 20, 21, 23].

Summarizing, we have that the two-grid operator (see (2.5)) leaves the spaces of $2h$-harmonics invariant, i.e., for each $\boldsymbol{\theta} \in \widetilde{\Theta}_{2g}$ it holds that

$$\begin{aligned} &M^{2h}_h|_{\mathcal{F}^{2g}_{\boldsymbol{\theta}}} \overset{\wedge}{=} M^{2g}(\boldsymbol{\theta}, h) \\ &= (S^{2g}(\boldsymbol{\theta}, h))^{\nu_2}(\,I^{2g} - P^{2g}(\boldsymbol{\theta}, h)(\mathcal{L}^{2g}(\boldsymbol{\theta}, h))^{-1}R^{2g}(\boldsymbol{\theta}, h)L^{2g}(\boldsymbol{\theta}, h)\,)(S^{2g}(\boldsymbol{\theta}, h))^{\nu_1}\,. \end{aligned} \tag{2.7}$$

The representation of the coarse grid correction block matrices is given by

$$\begin{aligned} I^{2g} &= \operatorname{diag}\{1,1,1,1\} \in \mathbb{C}^{4\times 4}\,, \quad \mathcal{L}^{2g}(\boldsymbol{\theta},h) = \widetilde{L}_{2h}(2\boldsymbol{\theta}^{00}) \in \mathbb{C}^{1\times 1}\,,\\ L^{2g}(\boldsymbol{\theta},h) &= \operatorname{diag}\{\widetilde{L}_h(\boldsymbol{\theta}^{00}), \widetilde{L}_h(\boldsymbol{\theta}^{11}), \widetilde{L}_h(\boldsymbol{\theta}^{10}), \widetilde{L}_h(\boldsymbol{\theta}^{01})\} \in \mathbb{C}^{4\times 4}\,,\\ R^{2g}(\boldsymbol{\theta},h) &= (\, \widetilde{R}_h^{2h}(\boldsymbol{\theta}^{00})\; \widetilde{R}_h^{2h}(\boldsymbol{\theta}^{11})\; \widetilde{R}_h^{2h}(\boldsymbol{\theta}^{10})\; \widetilde{R}_h^{2h}(\boldsymbol{\theta}^{01})\,) \in \mathbb{C}^{1\times 4}\,,\\ P^{2g}(\boldsymbol{\theta},h) &= (\, \widetilde{P}_{2h}^{h}(\boldsymbol{\theta}^{00})\; \widetilde{P}_{2h}^{h}(\boldsymbol{\theta}^{11})\; \widetilde{P}_{2h}^{h}(\boldsymbol{\theta}^{10})\; \widetilde{P}_{2h}^{h}(\boldsymbol{\theta}^{01})\,)^T \in \mathbb{C}^{4\times 1}\,. \end{aligned} \tag{2.8}$$

Using the simple block representation from (2.7), the spectral radius of the two-grid iteration matrix and thus the asymptotic two-grid convergence factor can be approximated by

$$\rho_{2g}(h) := \sup_{\boldsymbol{\theta}\in\tilde{\Theta}_{2g}} \rho(M^{2g}\,(\boldsymbol{\theta},h))\,. \tag{2.9}$$

Remark 1 (boundedness of $\rho_{2g}(h)$). In all the examples considered in section 3, we have $\tilde{\Theta}_{2g} = (-\pi/2,\pi/2] \setminus \{(0,0)\}$, as only $\widetilde{L}_h((0,0))$ and $\widetilde{L}_{2h}(2(0,0))$ are zero. However, the suprenum in (2.9) remains finite, since $\widetilde{R}_h^{2h}((0,0))\widetilde{L}_h((0,0))$ is rank deficient too, in such a way that $\lim_{\boldsymbol{\theta}\to(0,0)} \rho(M^{2g}(\boldsymbol{\theta},h)\,)$ is bounded; see [4].

The smoothing or one-grid convergence factor $\rho_{1g}(h)$, based on the "ideal" coarse grid correction operator Q_h^{2h} with $Q_h^{2h}|_{\mathcal{F}_{\boldsymbol{\theta}}^{2g}} \overset{\wedge}{=} Q = \operatorname{diag}\{0,1,1,1\}$ from [20], reads as

$$\rho_{1g}(h) := \sup_{\boldsymbol{\theta}\in\tilde{\Theta}_{2g}} \rho(\, Q(S^{2g}(\boldsymbol{\theta},h))^{\nu_1+\nu_2}\,)\,. \tag{2.10}$$

Q_h^{2h} annihilates the low-frequency error components and leaves the high-frequency components unchanged. $\rho_{1g}(h)$ yields reasonable convergence estimates as long as Q_h^{2h} is a good approximation of the real coarse grid correction operator.

2.3. Three-grid Fourier analysis. From (2.6), the error transformation by a three-grid cycle is given by $v_i = M_h^{4h} v_{i-1}$ with

$$\begin{aligned} M_h^{4h} &= S_h^{v_2} K_h^{4h} S_h^{v_1}\\ &= S_h^{v_2}(I_h - P_{2h}^h(I_{2h} - (M_{2h}^{4h})^\gamma)(L_{2h})^{-1} R_h^{2h} L_h) S_h^{v_1}\,, \end{aligned} \tag{2.11}$$

where M_{2h}^{4h}, defined by (2.5), reads as

$$M_{2h}^{4h} = S_{2h}^{\nu_2}(I_{2h} - P_{4h}^{2h}(L_{4h})^{-1} R_{4h}^{2h} L_{2h}) S_{2h}^{\nu_1}\,. \tag{2.12}$$

Instead of inverting L_{2h}, as is done in the two-grid cycle (2.5), the $2h$-equation is solved approximately by performing γ two-grid iterations M_{2h}^{4h} with zero initial approximation. This is reflected by the replacement of $(L_{2h})^{-1}$ from (2.5) by the expression

$$(\mathcal{L}_{2h}^{4h})^{-1} = (I_{2h} - (M_{2h}^{4h})^\gamma)(L_{2h})^{-1} \tag{2.13}$$

in (2.11). To see this, consider an arbitrary nonsingular system $Lu = f$ which is approximately solved by γ steps of an iterative method, $Cu^j = (C-L)u^{j-1} + f$, based on the splitting $L = C + (L - C)$. If a multigrid method is applied, we obtain $u^j = Mu^{j-1} + C^{-1}f$ with $C^{-1} = (I - M)L^{-1}$. M denotes the error transformation matrix by one multigrid cycle; see (2.6). Starting with $u^0 = 0$, the γth iterate can easily be written as $u^\gamma = (I - M^\gamma)L^{-1}f$. In a numerical algorithm, however, L^{-1} (and, in particular, $(L_{2h})^{-1}$) is, of course, not applied explicitly.

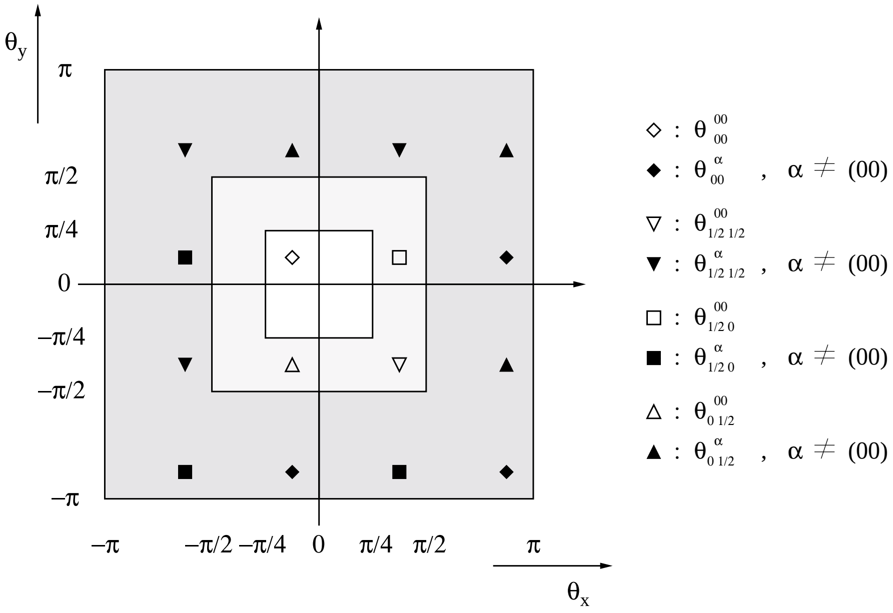

θ_y
π
$\pi/2$
$\pi/4$
0
$-\pi/4$
$-\pi/2$
$-\pi$
$-\pi$ $-\pi/2$ $-\pi/4$ 0 $\pi/4$ $\pi/2$ π
θ_x
◇ : θ^{00}_{00}
◆ : θ^{α}_{00} , $\alpha \neq (00)$
▽ : $\theta^{00}_{1/2\,1/2}$
▼ : $\theta^{\alpha}_{1/2\,1/2}$, $\alpha \neq (00)$
□ : $\theta^{00}_{1/2\,0}$
■ : $\theta^{\alpha}_{1/2\,0}$, $\alpha \neq (00)$
△ : $\theta^{00}_{0\,1/2}$
▲ : $\theta^{\alpha}_{0\,1/2}$, $\alpha \neq (00)$

FIG. 2.1. *A set of Fourier frequencies that are coupled by a three-grid iteration; see Definition* 2.2.

Considering three-grid cycles, it is appropriate to divide the Fourier space $\mathcal{F}$ into a direct sum of the following 16-dimensional subspaces.

DEFINITION 2.2 ($4h$-harmonics). *The $4h$-harmonics are defined by*

$$\begin{aligned} \mathcal{F}^{3g}_{\boldsymbol{\theta}} := span\Big\{ \phi(\boldsymbol{\theta}^{\boldsymbol{\alpha}}_{\boldsymbol{\beta}}, \boldsymbol{x}) \quad & with \;\; \boldsymbol{\alpha} = (\alpha_x\, \alpha_y)\,, \;\; \boldsymbol{\beta} = (\beta_x\, \beta_y), \\ & and \;\; \alpha_x, \alpha_y \in \{0,1\}\,, \;\; \beta_x, \beta_y \in \Big\{0, \frac{1}{2}\Big\} \;\Big\}, \\ where \quad \boldsymbol{\theta} = \boldsymbol{\theta}^{00}_{00} & \in \Theta_{3g} := (-\pi/4, \pi/4]^2 \quad and \\ \boldsymbol{\theta}^{00}_{\beta_x\beta_y} &= \boldsymbol{\theta}^{00}_{00} - (\, \beta_x sign(\theta_x), \beta_y sign(\theta_y)\,)\pi\,, \\ \boldsymbol{\theta}^{\alpha_x\alpha_y}_{\boldsymbol{\beta}} &= \boldsymbol{\theta}^{00}_{\boldsymbol{\beta}} - (\, \alpha_x sign(\theta_{\beta_x}), \alpha_y sign(\theta_{\beta_y})\,)\pi\,. \end{aligned}$$

Figure 2.1 illustrates this somewhat technical definition by indicating the location of the 16 different frequencies $\boldsymbol{\theta}^{\boldsymbol{\alpha}}_{\boldsymbol{\beta}}$. It can be motivated in the same way as it was done in the two-grid analysis concerning the $2h$-harmonics. In the transition from G_{2h} to G_{4h}, a low frequency $\boldsymbol{\theta}^{00}_{00} \in \Theta_{3g}$ is coupled with three high frequencies $\boldsymbol{\theta}^{00}_{\boldsymbol{\beta}}$ with $\boldsymbol{\beta} \neq (00)$. We collect four such components in the following subspaces.

DEFINITION 2.3.

$$\mathcal{F}^{\boldsymbol{\beta}}_{\boldsymbol{\theta}} := span\Big\{ \phi(\boldsymbol{\theta}^{00}_{\beta_x\beta_y}, \boldsymbol{x}) \quad with \;\; \beta_x, \beta_y \in \Big\{0, \frac{1}{2}\Big\} \;\Big\} \quad for \;\; \boldsymbol{\theta}^{00}_{00} \in \Theta_{3g}\,.$$

Furthermore, each $\boldsymbol{\theta}^{00}_{\boldsymbol{\beta}}$ is coupled with three high-frequency components $\boldsymbol{\theta}^{\boldsymbol{\alpha}}_{\boldsymbol{\beta}}$ with $\boldsymbol{\alpha} \neq (00)$ in the transition from G_h to G_{2h}; see Figure 2.1. It follows that a three-grid cycle couples 16 frequencies, i.e., the 15 high-frequency components alias on G_{4h} with the low-frequency component $\phi(\boldsymbol{\theta}^{00}_{00}, \boldsymbol{x})$.

Again, we exclude certain frequencies from the calculation to obtain a well-defined three-grid operator and consider the following slightly smaller spaces:

$$\mathcal{F}^{3g} := \mathcal{F} \backslash \bigcup_{\boldsymbol{\theta} \in \Psi_{3g}} \mathcal{F}^{3g}_{\boldsymbol{\theta}} \quad \text{and} \quad \tilde{\Theta}_{3g} := \Theta_{3g} \backslash \Psi_{3g} \quad \text{with}$$

$$\Psi_{3g} := \left\{ \boldsymbol{\theta} \in \Theta_{3g} : \ \widetilde{L}_{4h}(4\boldsymbol{\theta}^{00}_{00}) = 0 \ \text{ or } \ \widetilde{L}_{2h}(2\boldsymbol{\theta}^{00}_{\boldsymbol{\beta}}) = 0 \ \text{ or } \ \widetilde{L}_{h}(\boldsymbol{\theta}^{\boldsymbol{\alpha}}_{\boldsymbol{\beta}}) = 0 \right\}.$$

Comparing Definitions 2.1 and 2.2, it immediately follows that each space of $4h$-harmonics consists of four spaces of $2h$-harmonics, i.e., for an arbitrary $\boldsymbol{\theta} \in \tilde{\Theta}_{3g}$ we have

$$\mathcal{F}^{3g}_{\boldsymbol{\theta}} = \mathcal{F}^{2g}_{\boldsymbol{\theta}^{00}_{00}} \cup \mathcal{F}^{2g}_{\boldsymbol{\theta}^{00}_{\frac{1}{2}\frac{1}{2}}} \cup \mathcal{F}^{2g}_{\boldsymbol{\theta}^{00}_{\frac{1}{2}0}} \cup \mathcal{F}^{2g}_{\boldsymbol{\theta}^{00}_{0\frac{1}{2}}} . \tag{2.14}$$

As a fundamental statement for the three-grid analysis, we find that the three-grid operator M_h^{4h} (2.11) leaves the spaces of $4h$-harmonics invariant. Using Definition 2.3, (2.14), and the considerations from the last subsection, it follows for every $\boldsymbol{\theta} \in \tilde{\Theta}_{3g}$ that

$$\begin{aligned} &S_h \ : \ \mathcal{F}^{3g}_{\boldsymbol{\theta}} \longrightarrow \mathcal{F}^{3g}_{\boldsymbol{\theta}} \ , \quad L_h \ : \ \mathcal{F}^{3g}_{\boldsymbol{\theta}} \longrightarrow \mathcal{F}^{3g}_{\boldsymbol{\theta}} \ , \\ &R_h^{2h} \ : \ \mathcal{F}^{3g}_{\boldsymbol{\theta}} \longrightarrow \mathcal{F}^{\boldsymbol{\beta}}_{\boldsymbol{\theta}} \ , \quad \mathcal{L}^{4h}_{2h} \ : \ \mathcal{F}^{\boldsymbol{\beta}}_{\boldsymbol{\theta}} \longrightarrow \mathcal{F}^{\boldsymbol{\beta}}_{\boldsymbol{\theta}} \ , \quad P^{h}_{2h} \ : \ \mathcal{F}^{\boldsymbol{\beta}}_{\boldsymbol{\theta}} \longrightarrow \mathcal{F}^{3g}_{\boldsymbol{\theta}} \ . \end{aligned} \tag{2.15}$$

The relation for $\mathcal{L}^{4h}_{2h}$ reads in more detail as (see (2.13) and (2.12))

$$\begin{aligned} &L_{2h} \ : \ \mathcal{F}^{\boldsymbol{\beta}}_{\boldsymbol{\theta}} \longrightarrow \mathcal{F}^{\boldsymbol{\beta}}_{\boldsymbol{\theta}} \ , \quad S_{2h} \ : \ \mathcal{F}^{\boldsymbol{\beta}}_{\boldsymbol{\theta}} \longrightarrow \mathcal{F}^{\boldsymbol{\beta}}_{\boldsymbol{\theta}} \ , \quad R^{4h}_{2h} \ : \ \mathcal{F}^{\boldsymbol{\beta}}_{\boldsymbol{\theta}} \longrightarrow \operatorname{span}\{\phi(\boldsymbol{\theta}^{00}_{00}, \boldsymbol{x})\} \ , \\ &L_{4h} \ : \ \operatorname{span}\{\phi(\boldsymbol{\theta}^{00}_{00}, \boldsymbol{x})\} \longrightarrow \operatorname{span}\{\phi(\boldsymbol{\theta}^{00}_{00}, \boldsymbol{x})\} \ , \quad P^{2h}_{4h} \ : \ \operatorname{span}\{\phi(\boldsymbol{\theta}^{00}_{00}, \boldsymbol{x})\} \longrightarrow \mathcal{F}^{\boldsymbol{\beta}}_{\boldsymbol{\theta}} \ . \end{aligned}$$

As a consequence of (2.15), one obtains the following (16×16)-block matrices:

$$\begin{aligned} M_h^{4h}|_{\mathcal{F}^{3g}_{\boldsymbol{\theta}}} &\overset{\wedge}{=} M^{3g}(\boldsymbol{\theta}, h) \\ = (S^{3g}(\boldsymbol{\theta}, h))^{\nu_2} (I^{3g} &- P^{3g}(\boldsymbol{\theta}, h)(\mathcal{L}^{3g}(\boldsymbol{\theta}, h))^{-1} R^{3g}(\boldsymbol{\theta}, h) L^{3g}(\boldsymbol{\theta}, h) \ )(S^{3g}(\boldsymbol{\theta}, h))^{\nu_1} \ . \end{aligned} \tag{2.16}$$

The different operators of the block-matrices $M^{3g}(\boldsymbol{\theta}, h)$ can be expressed by the two-grid representations from section 2.2; see $S^{2g}(\boldsymbol{\theta}, h)$ and (2.8):

$$\begin{aligned} I^{3g} &= \operatorname{diag}\{I^{2g}, I^{2g}, I^{2g}, I^{2g}\} \in \mathbb{C}^{16\times 16} \ , \\ S^{3g}(\boldsymbol{\theta}, h) &= \operatorname{diag}\{S^{2g}(\boldsymbol{\theta}_{00}, h), S^{2g}(\boldsymbol{\theta}_{\frac{1}{2}\frac{1}{2}}, h), S^{2g}(\boldsymbol{\theta}_{\frac{1}{2}0}, h), S^{2g}(\boldsymbol{\theta}_{0\frac{1}{2}}, h)\} \in \mathbb{C}^{16\times 16} \ , \\ L^{3g}(\boldsymbol{\theta}, h) &= \operatorname{diag}\{L^{2g}(\boldsymbol{\theta}_{00}, h), L^{2g}(\boldsymbol{\theta}_{\frac{1}{2}\frac{1}{2}}, h), L^{2g}(\boldsymbol{\theta}_{\frac{1}{2}0}, h), L^{2g}(\boldsymbol{\theta}_{0\frac{1}{2}}, h)\} \in \mathbb{C}^{16\times 16} \ , \\ R^{3g}(\boldsymbol{\theta}, h) &= \operatorname{diag}\{R^{2g}(\boldsymbol{\theta}_{00}, h), R^{2g}(\boldsymbol{\theta}_{\frac{1}{2}\frac{1}{2}}, h), R^{2g}(\boldsymbol{\theta}_{\frac{1}{2}0}, h), R^{2g}(\boldsymbol{\theta}_{0\frac{1}{2}}, h)\} \in \mathbb{C}^{4\times 16} \ , \\ P^{3g}(\boldsymbol{\theta}, h) &= \operatorname{diag}\{P^{2g}(\boldsymbol{\theta}_{00}, h), P^{2g}(\boldsymbol{\theta}_{\frac{1}{2}\frac{1}{2}}, h), P^{2g}(\boldsymbol{\theta}_{\frac{1}{2}0}, h), P^{2g}(\boldsymbol{\theta}_{0\frac{1}{2}}, h)\} \in \mathbb{C}^{16\times 4} \ , \\ (\mathcal{L}^{3g}(\boldsymbol{\theta}, h))^{-1} &= \ (\ I^{2g} - (M^{2g}(2\boldsymbol{\theta}, 2h))^{\gamma} \ ) \\ &\quad (\ \operatorname{diag}\{\mathcal{L}^{2g}(\boldsymbol{\theta}_{00}, h), \mathcal{L}^{2g}(\boldsymbol{\theta}_{\frac{1}{2}\frac{1}{2}}, h), \mathcal{L}^{2g}(\boldsymbol{\theta}_{\frac{1}{2}0}, h), \mathcal{L}^{2g}(\boldsymbol{\theta}_{0\frac{1}{2}}, h)\} \ )^{-1} \in \mathbb{C}^{4\times 4} \ . \end{aligned} \tag{2.17}$$

Of course, $M^{2g}(2\boldsymbol{\theta}, 2h)$ can be calculated using $S^{2g}(\boldsymbol{\theta}, h)$ and (2.8) if we replace h by $2h$ and $\boldsymbol{\theta}$ by $2\boldsymbol{\theta}$.

Analogous to the two-grid factor $\rho_{2g}(h)$ from (2.9), we obtain the three-grid convergence factor as the suprenum of the spectral radii from certain block-matrices:

$$\rho_{3g}(h) := \sup_{\boldsymbol{\theta} \in \tilde{\Theta}_{3g}} \rho(M^{3g}(\boldsymbol{\theta}, h)) \,. \tag{2.18}$$

The considerations from Remark 1 concerning the boundedness of $\rho_{2g}(h)$ carry over to the three-grid factors calculated in section 3.

Remark 2 (generalization to d dimensions, k grids, and systems of equations). In the most general case, we consider a d-dimensional problem and apply k-grid cycles to a system of q equations. Then, every low-frequency $\boldsymbol{\theta} \in \Theta_{kg} := (-2^{1-k}\pi, 2^{1-k}\pi]^d$ is coupled with $2^{d(k-1)} - 1$ high frequencies. Accordingly, the related $2^{k-1}h$-harmonics are of dimension $2^{d(k-1)}$. Each operator of the k-grid cycle acts on the whole system and the dimensions of the corresponding block-matrices (see (2.8) and (2.17) for the two-dimensional scalar case with two- and three-grid cycles, respectively) are given by

$$I^{kg},\ S^{kg}(\boldsymbol{\theta}, h),\ L^{kg}(\boldsymbol{\theta}, h) \in \mathbb{C}^{2^{d(k-1)}q \times 2^{d(k-1)}q}, \quad R^{kg}(\boldsymbol{\theta}, h) \in \mathbb{C}^{2^{d(k-2)}q \times 2^{d(k-1)}q},$$
$$P^{kg}(\boldsymbol{\theta}, h) \in \mathbb{C}^{2^{d(k-1)}q \times 2^{d(k-2)}q}, \quad \mathcal{L}^{kg}(\boldsymbol{\theta}, h) \in \mathbb{C}^{2^{d(k-2)}q \times 2^{d(k-2)}q}.$$

The evaluation of k-grid cycles appears to be quite complicated and costly for many-level cycles, but one should take into account that the k-grid operators (2.6) are recursively defined and can be expressed in terms of two-grid operators. This means that the entries of a k-grid Fourier matrix representation are given by certain two-grid Fourier symbols; see (2.17). *In practice, however, it should usually be enough to perform a three-grid analysis to obtain sufficient insight into a multigrid method.*

Remark 3 (finite-dimensional Fourier space). Note that the Fourier space (2.4) has a nondenumerable basis as $\boldsymbol{\theta}$ varies continuously in $(\pi, \pi]^2$. The use of infinite-dimensional spaces and operators leads to some technical simplifications in the analysis; see [20]. However, in general, the suprema from (2.9), (2.10), and (2.18) cannot be calculated analytically. Therefore, we restrict our practical computations in section 3 to a finite-dimensional Fourier space which is related to the mesh size h under consideration:

$$\mathcal{F}_{finite} := \text{span}\{e^{i\boldsymbol{j}\boldsymbol{\theta}} : \boldsymbol{\theta} \in \Theta_{finite} := (-\pi, \pi]^2 \cap G_{h_{\boldsymbol{\theta}}}\}$$
$$\text{with } G_{h_{\boldsymbol{\theta}}} := \{\boldsymbol{\theta} = h_{\boldsymbol{\theta}}\boldsymbol{j} = (h_{\boldsymbol{\theta}}j_x, h_{\boldsymbol{\theta}}j_y) \text{ with } h_{\boldsymbol{\theta}} = 2\pi h \,, \boldsymbol{j} \in \mathbb{Z}^2\} \,.$$

The definitions of $\tilde{\Theta}_{2g}$, $\tilde{\Theta}_{3g}$, $\mathcal{F}^{2g}$, and $\mathcal{F}^{3g}$ have to be changed accordingly. Hence the suprema are replaced by maxima, which can easily be calculated numerically. Using this finite-dimensional Fourier space, the infinite Fourier analysis becomes an exact analysis for certain model problems on rectangular domains with periodic boundary conditions. Pure periodic boundary conditions lead to a singular boundary value problem in general. This necessitates a compatibility condition for every iterative solution method, which directly corresponds to the exclusion of the "zero" frequency in the analysis; see Remark 1. More details can be found, for example, in [20, 21].

2.4. Generalization for multigrid as a preconditioner for GMRES(m). In this section, we briefly describe the generalization of the above analysis to multigrid as a right preconditioner for GMRES(m). For a detailed discussion with respect to the two-grid analysis, we refer to [24].

As Krylov subspace methods are commonly described using matrix and vector notation, we consider the linear system $Lu = f$, related to (2.1) with $n = k$. A three-grid (or, more generally, a multigrid) cycle can be described by the matrix splitting, $Cu_j + (L - C)u_{j-1} = f$, where u_j and u_{j-1} represent a new and a previous approximation. This formulation is equivalent to

$$u_j = u_{j-1} + C^{-1}(f - Lu_{j-1}) \quad \text{and} \quad r_j = (I - LC^{-1})r_{j-1}$$

with the residual vectors r_j, r_{j-1} and the residual transformation matrix

$$I - LC^{-1} = LML^{-1}, \tag{2.19}$$

where M denotes the three-grid iteration matrix. GMRES(m) searches for a new approximation u_j with corresponding residual r_j in the Krylov subspace

$$K^m(LC^{-1}, r_{j-m}) := \text{span}[r_{j-m}, (LC^{-1})r_{j-m}, \dots, (LC^{-1})^{m-1}r_{j-m}] \ .$$

It selects the new solution by minimizing the residual in the discrete Euclidean 2-norm

$$||r_j||_2 = \min\{||P_m(LC^{-1})r_{j-m}||_2 \ | \ P_m \in \Pi_m\}, \tag{2.20}$$

where Π_m denotes the set of all polynomials of degree at most m with $P_m(0) = 1$. For convenience, $j \geq m$ is assumed. Since we are interested in the asymptotic convergence of multigrid preconditioned GMRES with a restart after m iterations, we focus on the residuals $r_m, r_{2m}, \dots, r_{i \cdot m}$. Then a "complete" iteration with iteration index i consists of m multigrid preconditioned GMRES(m) steps. The GMRES(m)-polynomial which characterizes the ith complete iteration is denoted by P_m^i, leading to the following recursion for the corresponding residual:

$$r_{i \cdot m} = P_m^i(LC^{-1})r_{(i-1)\cdot m}. \tag{2.21}$$

As unitary transformations do not affect the convergence properties of GMRES, we consider the Fourier representations $\widetilde{M}^{3g}$ and $\widetilde{L}^{3g}$ instead of the representations M and L with respect to the Euclidean basis. More precisely, we use the finite-dimensional variants

$$\widetilde{M}^{3g} := [M^{3g}(\boldsymbol{\theta}, h)]_{\boldsymbol{\theta} \in \tilde{\Theta}_{3g} \cap G_{h_{\boldsymbol{\theta}}}} \quad \text{and} \quad \widetilde{L}^{3g} := [L^{3g}(\boldsymbol{\theta}, h)]_{\boldsymbol{\theta} \in \tilde{\Theta}_{3g} \cap G_{h_{\boldsymbol{\theta}}}}$$

to allow an explicit calculation; see Remark 3. Assuming a repeated application of preconditioned GMRES(m), the following function g has to be minimized in order to find the coefficients α_k^i ($k = 1, ..., m$) of the ith GMRES(m)-polynomial P_m^i (see (2.19), (2.20), and (2.21)):

$$g(\alpha_1^i, \dots, \alpha_m^i) := (P_m^i(I - \widetilde{L}^{3g}\widetilde{M}^{3g}(\widetilde{L}^{3g})^{-1})\tilde{r}_{(i-1)m}, P_m^i(I - \widetilde{L}^{3g}\widetilde{M}^{3g}(\widetilde{L}^{3g})^{-1})\tilde{r}_{(i-1)m}).$$

The α_k^i are obtained by solving the linear system

$$\begin{aligned} \frac{\partial g}{\partial \alpha_\ell^i} &= 2\sum_{k=1}^{m} \alpha_k^i((I - \widetilde{L}^{3g}\widetilde{M}^{3g}(\widetilde{L}^{3g})^{-1})^\ell \tilde{r}_{(i-1)m}, (I - \widetilde{L}^{3g}\widetilde{M}^{3g}(\widetilde{L}^{3g})^{-1})^k \tilde{r}_{(i-1)m}) \\ &+ 2((I - \widetilde{L}^{3g}\widetilde{M}^{3g}(\widetilde{L}^{3g})^{-1})^\ell \tilde{r}_{(i-1)m}, \tilde{r}_{(i-1)m}) = 0 \qquad \text{for } \ell = 1, \dots, m. \end{aligned} \tag{2.22}$$

The solution of (2.22) can easily be computed for every iteration index i due to the sparse block structure of $(I - \widetilde{L}^{3g}\widetilde{M}^{3g}(\widetilde{L}^{3g})^{-1})^\ell$ ($\ell = 1, \dots, m$) if the previous

Fourier transformed residual $\tilde{r}_{(i-1)m}$ is given. We simply prescribe a randomly chosen initial residual $\tilde{r}_0$. This allows the calculation of $\alpha^1_\ell(\ell = 1,\dots,m)$ by (2.22) and gives $\tilde{r}_{1\cdot m} = P^1_m(I - \widetilde{L}^{3g}\widetilde{M}^{3g}(\widetilde{L}^{3g})^{-1})\tilde{r}_0$. Then the computation of $\tilde{r}_{i\cdot m}$ for $i > 1$ is straightforward due to its recursive definition; see (2.21). This leads to an average reduction factor $\rho^{acc}_{3g}(i,m)$ for a complete iteration, which can be obtained by the three-grid Fourier analysis

$$\rho^{acc}_{3g}(i,m) := \left[\left(\frac{||\tilde{r}_{i\cdot m}||_2}{||\tilde{r}_0||_2}\right)^{1/m}\right]^{1/i} . \tag{2.23}$$

The superscript "acc" is used as an abbreviation for "accelerated," since the combination of multigrid and GMRES(m) can be interpreted as an acceleration of the multigrid convergence speed by an additional application of GMRES(m).

In all tests, presented in section 3, $\rho^{acc}_{3g}(i,m)$ tends to a constant for $i \geq 20$. The particular choice of the initial residual $\tilde{r}_0$ does not influence the average reduction factors for $i \gg 1$, which is confirmed by systematic test calculations. Thus it is expected that $\rho^{acc}_{3g}(m) := \rho^{acc}_{3g}(20,m)$ matches well with numerical reference values.

If we use the above Fourier representation $I - \widetilde{L}^{3g}\widetilde{M}^{3g}(\widetilde{L}^{3g})^{-1}$, the corresponding spectrum can be calculated numerically. The common way (see, for example, [15, 16]) to analyze the convergence of GMRES is to exploit information about the spectrum σ of the iteration matrix LC^{-1}.

Suppose that all eigenvalues of $I - \widetilde{L}^{3g}\widetilde{M}^{3g}(\widetilde{L}^{3g})^{-1}$ are located in an ellipse $E(c,d,a)$ which excludes the origin. $(c,0)$ denotes the center, d denotes the focal distance, and a denotes the major semiaxis. Note that σ is always symmetric with respect to the real axis, so we only consider ellipses which are aligned with the axes and where the ordinate of the center equals zero. Then it is known [16] that the absolute value of the polynomial

$$t_m(z) := T_m\Big(\frac{c}{d} - \frac{1}{d}\,z\Big)\Big/T_m\Big(\frac{c}{d}\Big) = T_m(\widehat{z})\Big/T_m\Big(\frac{c}{d}\Big) \quad \text{with} \quad z,\, \widehat{z} := \Big(\frac{c}{d} - \frac{1}{d}\,z\Big) \in \mathbb{C}$$

is small on the spectrum of $I - \widetilde{L}^{3g}\widetilde{M}^{3g}(\widetilde{L}^{3g})^{-1}$. Here T_m represents the Chebychev polynomial of degree m of the first kind; see [16]. If $I - \widetilde{L}^{3g}\widetilde{M}^{3g}(\widetilde{L}^{3g})^{-1}$ is diagonalizable, $I - \widetilde{L}^{3g}\widetilde{M}^{3g}(\widetilde{L}^{3g})^{-1} = XDX^{-1}$, then (2.20) yields

$$\begin{aligned} ||\tilde{r}_{i\cdot m}||_2 &\leq ||t_m(I - \widetilde{L}^{3g}\widetilde{M}^{3g}(\widetilde{L}^{3g})^{-1})||_2\; ||\tilde{r}_{(i-1)m}||_2 \\ &\leq [||t_m(I - \widetilde{L}^{3g}\widetilde{M}^{3g}(\widetilde{L}^{3g})^{-1})||_2]^i\; ||\tilde{r}_0||_2 \;\leq\; \left[\kappa_2(X)\, T_m\Big(\frac{a}{d}\Big)/T_m\Big(\frac{c}{d}\Big)\right]^i ||\tilde{r}_0||_2, \end{aligned}$$

where $\kappa_2(X)$ denotes the spectral condition number of the transformation matrix X [16]. Using these inequalities, we obtain for an arbitrary complete iteration i

$$\rho^{acc}_{3g}(i,m) \;\leq\; N^E_m \;\leq\; (\kappa_2(X))^{1/m}\, T^E_m \quad \text{with} \tag{2.24}$$

$$N^E_m := (||t_m(I - \widetilde{L}^{3g}\widetilde{M}^{3g}(\widetilde{L}^{3g})^{-1})||_2)^{1/m} \quad \text{and} \quad T^E_m := \left(T_m\Big(\frac{a}{d}\Big)/T_m\Big(\frac{c}{d}\Big)\right)^{1/m} \tag{2.25}$$

as approximations for the average reduction factors of m multigrid preconditioned GMRES(m) steps (see (2.23)).

Remark 4 (determination of ellipses). There seems to be no simple way to determine the "optimal" ellipse for a given spectrum σ which minimizes T_m^E and N_m^E from (2.25). Thus it is proposed in [8] to compute a rectangle containing the spectrum σ and then to calculate the ellipse of smallest area which encloses this rectangle. Both steps of this two-stage strategy are rather simple and straightforward (see [8]), but the resulting ellipses may cover much too large an area due to the first "auxiliary" step. The resulting estimates reflect the qualitative convergence behavior of GMRES(m), but the explicit values are far too pessimistic in general. Therefore, we tune the ellipses by hand. It is well known that each ellipse is uniquely defined if we prescribe three points that should lie on the ellipse. As a first guess, we select those three eigenvalues contained in σ with maximal real part (λ_{max}^{Re}), with minimal real part (λ_{min}^{Re}), and with maximal imaginary part (λ_{max}^{Im}). Then the semiaxis in the x-direction a_x, the semiaxis in the y-direction a_y, and the center $(c, 0)$ of the ellipse are given by

$$a_x = \frac{\mathrm{Re}(\lambda_{max}^{Re}) - \mathrm{Re}(\lambda_{min}^{Re})}{2}, \quad c = \mathrm{Re}(\lambda_{min}^{Re}) + a_x, \quad \text{and} \quad a_y = \frac{\mathrm{Im}(\lambda_{max}^{Im})}{\sqrt{1 - \frac{\mathrm{Re}(\lambda_{max}^{Im}) - c}{a_x^2}}}.$$

$\mathrm{Re}(\lambda)$ and $\mathrm{Im}(\lambda)$ denote the real and imaginary part of the complex eigenvalue λ. (In this way, the ellipse from Figure 3.2b is calculated.) If this first guess does not contain the whole spectrum, the ellipse is carefully enlarged until it covers σ. (An example is given in Figure 3.2a.) Although this strategy might not be very satisfactory from a mathematical viewpoint, it yields sharper estimates which can be easily calculated.

In [17] it is stated that (2.24) is an asymptotic result and that the actual residual norm should rather behave like T_m^E without $\kappa_2(X)$. This presumption, in connection with the two-grid Fourier analysis, is validated in [24], where it is found that the heuristic estimate T_m^E gives a certain insight into the asymptotic accelerated two-grid convergence, whereas the upper bound N_m^E is too pessimistic in general. Therefore, some explicit values for T_m^E based on the spectra of three-grid iteration matrices are given in section 3 and compared with asymptotic numerical convergence results. However, the main focus lies on $\rho_{3g}^{acc}(m)$.

3. Applications of three-grid Fourier analysis.

In order to demonstrate the benefits of the three-grid analysis, we consider several equations, discretizations, and multigrid components. The applications range from the nicely elliptic Poisson equation in connection with standard components to singular perturbation problems with more advanced multigrid components. By considering a large number of smoothers and transfer operators, we intend to show the large range of applicability of the three-grid analysis. In each of the following three subsections, we consider coarse grid correction problems which can often be solved by algebraic multigrid (AMG) [19], because AMG implicitly selects the "correct" coarsening strategy for the problem under consideration, or by a special relaxation method. Of course, it is possible to adapt the Fourier analysis from section 2 to nonstandard coarsenings or to sophisticated smoothers. However, we try to keep the multigrid method simple and search for possible improvements by carefully chosen transfer operators and Galerkin coarse grid discretizations. Furthermore, we investigate the use of an additional acceleration with GMRES(m) of nonoptimal but easy-to-program multigrid methods. The main purpose is to evaluate the additional insights coming from the three-grid Fourier analysis.

The theoretical estimates are compared with numerical experiments whose convergence is indicated by $\rho_n(k\mathrm{g})$, which denotes the average defect reduction after 100

Table 3.1
*MG*1 *applied to the Poisson equation,* $h = 1/128$.

Cycle	$\rho_{1g}(h)$	$\rho_{2g}(h)$	$\rho_{3g}(h)$	ρ_n(3g)	ρ_n(7g)
V(1,1)	0.063	0.074	0.106	0.105	0.119
V(2,0)	0.063	0.074	0.133	0.132	0.170
V(0,2)	0.063	0.074	0.140	0.138	0.179
W(1,1), W(2,0), W(0,2)	0.063	0.074	0.074	0.073	0.073

iterations for the corresponding solution method involving k grids. We choose such a large number of iterations because the theoretical values $\rho_{2g}(h)$ and $\rho_{3g}(h)$ refer to asymptotic convergence factors. For nonsymmetric problems like the convection-dominated examples from section 3.3, it might take a large number of iterations before the asymptotic behavior is observed. An alternative is to consider norms of the three-grid operator, like the defect reduction norm

$$\sigma_{3g}(h) := \sup_{\boldsymbol{\theta}\in\tilde{\Theta}_{3g}} \sqrt{\rho(L^{3g}(\boldsymbol{\theta},h)M^{3g}(\boldsymbol{\theta},h)(L^{3g}(\boldsymbol{\theta},h))^{-1}(L^{3g}(\boldsymbol{\theta},h)M^{3g}(\boldsymbol{\theta},h)(L^{3g}(\boldsymbol{\theta},h))^{-1})^{*})}, \tag{3.1}$$

which can be obtained straightforwardly by the above analysis. The star * in (3.1) denotes, as usual, the adjoint of the matrix.

3.1. Poisson equation. The first example deals with the well-known 5-point discretization of the Poisson equation,

$$-\Delta u(\boldsymbol{x}) = 1 \quad \text{on } \Omega = (0,1)^2, \qquad u(\boldsymbol{x}) = 0 \quad \text{on } \Gamma = [0,1]^2 \setminus \Omega. \tag{3.2}$$

An efficient multigrid method [20] for this problem, denoted by $MG1$, consists of

- direct $2h$-, $4h$-, etc. coarse grid discretizations,
- bilinear interpolation of coarse grid corrections and full weighting of residuals, and
- red-black Gauss–Seidel relaxation.

Table 3.1 compares the analytical predictions from the one-, two-, and three-grid analysis with numerical reference values ρ_n(3g) and ρ_n(7g) for several 3- and 7-grid cycles. This table illustrates that even for such a simple and well-understood problem there is a difference between the performance of a V-cycle and a W-cycle and pre- and postsmoothing which cannot be displayed by Fourier two-grid analysis, whereas the different behavior of the cycle variants is very accurately predicted by the three-grid estimates $\rho_{3g}(h)$.

Remark 5 (maintaining the two-grid convergence). As it is seen in Table 3.1, one has to choose the multigrid W-cycle to obtain the two-grid convergence factor. This is indicated by the Fourier analysis. The theoretical predictions for the two- and three-grid factors are equal only for the W-cycle. If we replace the 5-point discretizations on the coarse grids by operators based on Galerkin coarsening with full weighting and bilinear interpolation, the V-cycle also leads to k-independent fast convergence. Using a four-color Gauss–Seidel relaxation for the resulting symmetric 9-point operators, we get for a V(1,1)-cycle $\rho_{1g}(h) = \rho_{2g}(h) = \rho_{3g}(h) = 0.063$. This value is validated by the corresponding numerical calculation for a 7-grid method.

A second multigrid variant, $MG2$ [1], with a simplified coarse grid correction is given by

TABLE 3.2
MG2 [1] *applied to the Poisson equation, $h = 1/128$.*

$V(1,1)$			$W(1,1)$		
$\rho_{1g}(h)$	$\rho_{2g}(h)$	$\rho_{3g}(h)$	$\rho_{1g}(h)$	$\rho_{2g}(h)$	$\rho_{3g}(h)$
0.06	0.50	0.75	0.06	0.50	0.62

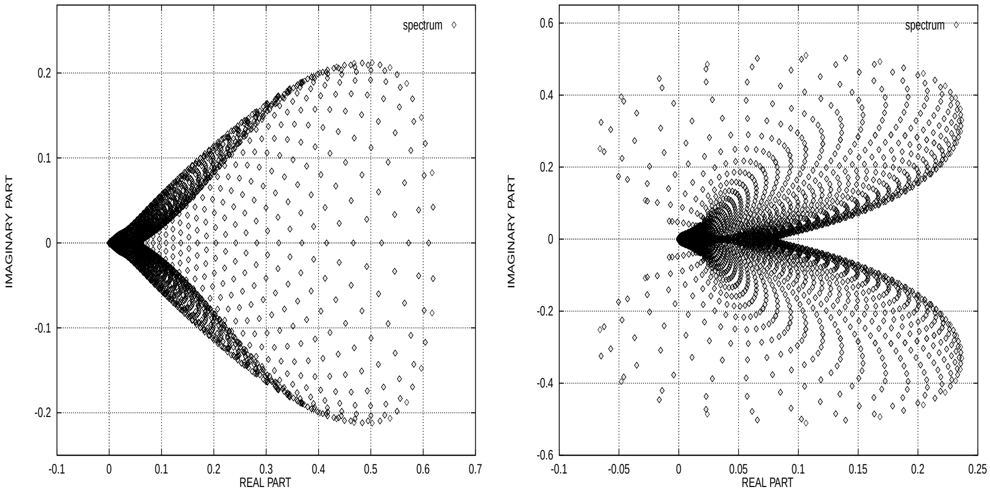


FIG. 3.1a. $\alpha = 1$

FIG. 3.1b. $\alpha = 2$

FIG. 3.1. *Eigenvalue spectra from Fourier analysis for three-grid $W(1,1)$-cycles of MG2* [1] *with and without rescaling applied to the Poisson equation, $h = 1/128$.*

- Galerkin coarse grid discretizations,
- piecewise constant interpolation and its transpose as transfer operators, and
- symmetric lexicographical Gauss–Seidel relaxation.

Modified versions of this strategy are used in the so-called aggregation-type AMG approaches, which might be useful for problems on unstructured grids; see, for example, [1, 22]. A direct application of this method to the Poisson equation leads to inefficient and strongly h-dependent V- (and W-) cycles. It can be easily shown that each of the coarse grid operators is off by a factor of 2 compared to the discretization on the next finer grid [1, 19]. A simple recursive argument yields that the V-cycle convergence on k grids is limited by $\rho_{kg}(h) \geq 1 - 2^{-k+1}$ ($\rho_{2g}(h) \geq 0.5$, $\rho_{3g}(h) \geq 0.75, \dots$) [19]. Table 3.2 shows that these limiting values can be confirmed by Fourier two- and three-grid analysis.

The above considerations motivate us to rescale the coarse grid Galerkin operators by $1/\alpha$ with α close to 2, $L_{n-1} = 1/\alpha \, \check{R}_n^{n-1} L_n \check{P}_{n-1}^n$, as it is proposed in [1]. For $\alpha = 2$, the Galerkin operators coincide with the standard 5-point discretizations on all grids. Figure 3.1 demonstrates the strong influence of this rescaling on the eigenvalue distribution of a $W(1,1)$ three-grid cycle from Fourier analysis.

Comparing Tables 3.2 and 3.3, one observes the improvement of the two- and three-grid factors due to the rescaling. However, the coarse grid correction problem is not completely solved for V-cycles which can be deduced from the big differences between the one-, two-, and three-grid estimates and which is validated by the further increasing numerical reference value $\rho_n(7\text{g})$ for a 7-grid cycle. This behavior might refer to the fact that the order of the transfer operators (see, for example, [10, 12, 23])

Table 3.3
$MG2$ [1] with a rescaling by $\alpha = 2$ applied to the Poisson equation, $h = 1/128$.

	$MG2$ as a solver				
Cycle	$\rho_{1g}(h)$	$\rho_{2g}(h)$	$\rho_{3g}(h)$	ρ_n(3g)	ρ_n(7g)
V(1,1)	0.06	0.44	0.68	0.66	0.82
W(1,1)	0.06	0.44	0.52	0.50	0.51

	$MG2$ as a preconditioner, $m = 5$		
Cycle	$\rho_{3g}^{acc}(m)$	ρ_n(3g)	ρ_n(7g)
V(1,1)	0.44	0.42	0.53
W(1,1)	0.36	0.34	0.35

in the multigrid process is too low compared to the order of the partial differential equation.

By combining $MG2$ with GMRES(m), one finds satisfactory convergence factors with a rather small Krylov subspace. The actual improvement for a three-grid method can be accurately predicted by $\rho_{3g}^{acc}(m)$; see Table 3.3.

Remark 6 (acceleration on coarse grids, F-cycle convergence). As the coarse grid difficulty occurs on all coarser grids, it seems reasonable to incorporate the Krylov acceleration into the multigrid cycle, like in [14], or to apply it *only on the coarse grids.* In the present example, one obtains ρ_n(7g)=0.52 for a V(1,1)-cycle if GMRES($m = 5$) is applied only on the coarse grids. In this way, it is possible to reduce the storage because on coarser grids much less storage is needed for a Krylov subspace. Furthermore, the multigrid F-cycle yields very similar convergence factors as the W-cycle, both as a solver and as a preconditioner.

Of course, these simple transfer operators are not at all an optimal choice for problem (3.2), especially compared to $MG1$ (see Table 3.1) or to AMG [19]. But it is a first example of a multigrid method with a coarse grid correction difficulty already for the Poisson equation and therefore an illustrative starting example. Furthermore, the method is easy to program and can be seen as a basis for more advanced aggregation-type AMG methods.

3.2. Rotated anisotropic diffusion equation. Next we discuss the standard 9-point discretization (see, for example, [23]) of the rotated anisotropic diffusion equation

$$-(c^2 + \varepsilon s^2)\, u_{xx}(\boldsymbol{x}) - 2(\varepsilon - 1)cs\, u_{xy}(\boldsymbol{x}) - (\varepsilon c^2 + s^2)\, u_{yy}(\boldsymbol{x}) = 1 \quad \text{on} \;\; \Omega = (0,1)^2$$
$$\text{with} \quad c = \cos(\beta),\;\; s = \sin(\beta), \quad u(\boldsymbol{x}) = 0 \quad \text{on} \;\; \Gamma = [0,1]^2 \setminus \Omega.$$

This differential operator corresponds to $-u_{\xi\xi} - \varepsilon u_{\eta\eta}$ in a (ξ, η)-coordinate system which can be obtained by rotating the (x, y)-system by an angle of β [23]. We set $\beta = 45°$. For $\varepsilon \to 0$ this equation is no longer elliptic. Using standard grid coarsening and Gauss–Seidel smoothing, this leads to coarse grid correction difficulties which limit the two-grid convergence to a factor of 0.75; see, for instance, [24]. The same recursive argument as in the previous subsection yields a lower bound for the V-cycle convergence on k grids which is given by $\rho_{kg}(h) \geq 1 - 4^{-k+1}$ ($\rho_{2g}(h) \geq 0.75$, $\rho_{3g}(h) \geq 0.9375, \dots$). These bounds can be established by Fourier two- and three-grid analysis as can be seen from Table 3.4, where $MG1$ is applied to the rotated anisotropic diffusion equation. Switching to a W-cycle leads to a slight improvement, which is well predicted by the three-grid analysis.

If $MG1$ is used as a preconditioner, it is possible to obtain an acceptable W-cycle

TABLE 3.4
$MG1$ applied to the rotated anisotropic diffusion equation, $\beta = 45°, \varepsilon = 10^{-5}$, $h = 1/128$.

	$MG1$ as a solver				
Cycle	$\rho_{1g}(h)$	$\rho_{2g}(h)$	$\rho_{3g}(h)$	$\rho_n(3\text{g})$	$\rho_n(7\text{g})$
$V(1,1)$	0.35	0.76	0.94	0.92	0.95
$W(1,1)$	0.35	0.76	0.90	0.87	0.89

	$MG1$ as a preconditioner, $m = 5$			
Cycle	T_m^E	$\rho_{3g}^{acc}(m)$	$\rho_n(3\text{g})$	$\rho_n(7\text{g})$
$V(1,1)$	0.79	0.68	0.63	0.73
$W(1,1)$	0.67	0.58	0.52	0.55

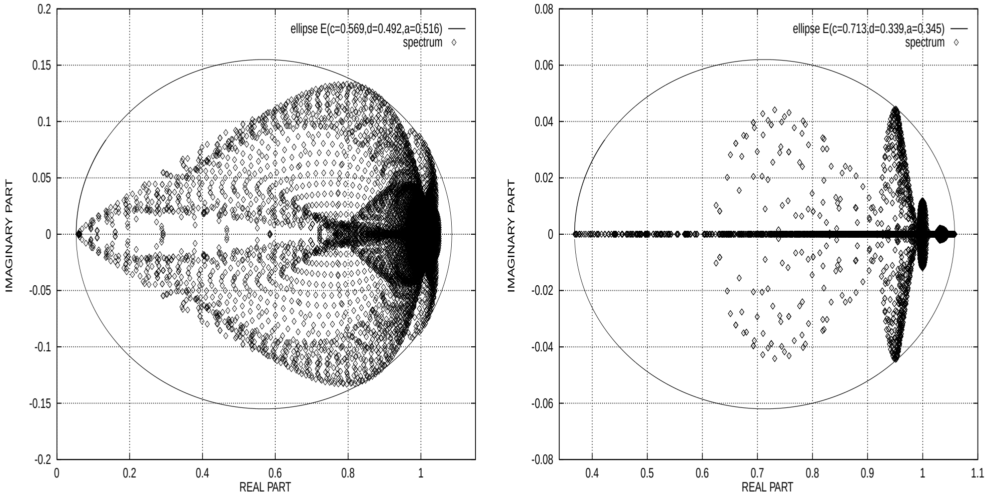


FIG. 3.2a. $MG1$ FIG. 3.2b. $MG3$

FIG. 3.2. *Eigenvalue spectra of $I - \widetilde{L}^{3g}\widetilde{M}^{3g}(\widetilde{L}^{3g})^{-1}$ for three-grid $V(1,1)$-cycles of $MG1$ and $MG3$ applied to the rotated anisotropic diffusion equation, $\beta = 45°, \varepsilon = 10^{-5}$, $h = 1/128$.*

convergence, whereas the V-cycle convergence for a 7-grid method remains unsatisfactory. Again, the analytical values $\rho_{3g}^{acc}(m)$ yield reliable predictions that are more accurate than those indicated by T_m^E. As an example, Figure 3.2a shows the $V(1,1)$-cycle spectrum of $I - \widetilde{L}^{3g}\widetilde{M}^{3g}(\widetilde{L}^{3g})^{-1}$ from Fourier analysis for $MG1$ and the corresponding ellipse which is used to calculate T_m^E. For a more detailed discussion of such values and spectra, see [24], as the main focus in this paper lies on the three-grid analysis.

In the context of Galerkin coarsening for the rotated anisotropic diffusion equation, it is interesting to investigate the multigrid method, $MG3$,

- Galerkin coarse grid discretizations,
- matrix-dependent prolongation and restriction by de Zeeuw [27], and
- four-color Gauss–Seidel relaxation.

Comparing Tables 3.4 and 3.5, we see a remarkable improvement of the two-grid convergence factor. For this example, however, Fourier one- and two-grid analysis yield somewhat misleading results. At first sight, the coarse grid correction problem seems to be solved since the two-grid value nearly recovers the smoothing factor. But if we look at the increased three-grid values, one has to expect that the multigrid convergence deteriorates further, which is validated by the numerical reference values

TABLE 3.5
MG3 applied to the rotated anisotropic diffusion equation, $\beta = 45°, \varepsilon = 10^{-5}$, $h = 1/128$.

	*MG*3 as a solver				
Cycle	$\rho_{1g}(h)$	$\rho_{2g}(h)$	$\rho_{3g}(h)$	ρ_n(3g)	ρ_n(7g)
V(1,1)	0.28	0.36	0.63	0.61	0.87
W(1,1)	0.28	0.36	0.47	0.48	0.61

	*MG*3 as a preconditioner, $m = 5$			
Cycle	T_m^E	$\rho_{3g}^{acc}(m)$	ρ_n(3g)	ρ_n(7g)
V(1,1)	0.31	0.29	0.28	0.55
W(1,1)	0.21	0.19	0.19	0.27

for the related 7-grid iterations.

On the other hand, we see that the three-grid methods can be nicely accelerated by GMRES(m) with a small Krylov subspace. Thus it can be expected also that the corresponding multigrid iterations are appropriate preconditioners, which is confirmed by the numerical values. The actual performance of the accelerated three-grid methods is very accurately estimated by $\rho_3^{acc}(m)$ and T_m^E. The V(1,1)-cycle spectrum of $I - \widetilde{L}^{3g}\widetilde{M}^{3g}(\widetilde{L}^{3g})^{-1}$ from Fourier analysis for $MG3$ and the related ellipse are pictured in Figure 3.2b.

Remark 7 (other Galerkin coarsenings). If the prolongation and restriction from [27] in $MG3$ are replaced by the transfer operators from the nonsymmetric blackbox multigrid by Dendy [7], we find very similar results. These transfer operators are investigated in the next subsection. The application of $MG2$ to this problem cannot be recommended. $MG2$ is mainly developed for elliptic problems and does not converge well for the rotated anisotropic diffusion equation.

Finally, we would like to mention that there are other multigrid components like ILU-type smoothers or nonstandard coarsening to overcome the coarse grid correction difficulty efficiently. De Zeeuw [27], for example, combines his matrix-dependent Galerkin coarsening with a powerful smoother, the incomplete line LU decomposition (ILLU), and obtains very fast multigrid convergence. These improvements might be verified by a straightforward adaption of the presented three-grid analysis.

3.3. Convection diffusion equation. As a third example, we discuss the convection diffusion equation with dominant convection in some detail. Here, it is important to distinguish between *entering* flows with an inflow and outflow boundary and *recirculating* flows for which no real inflow and outflow boundary exist and where the boundary information is mainly diffusing into the domain. In principle, efficient multigrid iterations can be constructed for upstream discretizations if relaxation methods are used with a downstream ordering of grid points. Then the relaxation acts not only as a smoother but also partly as a solver and takes care of problematic characteristic low-frequency error components; see [5]. For entering flows such smoothers can be found among standard relaxation methods, whereas for recirculating flows it is very difficult to construct a smoother with the desired property. Channel-type flows employing higher order upwind discretizations are treated successfully in [13].

For convection-dominated rotating flow problems like

$$-\varepsilon\Delta u(\boldsymbol{x}) + a(x,y)\,u_x(\boldsymbol{x}) + b(x,y)\,u_y(\boldsymbol{x}) = 1 \quad \text{on } \ \Omega = (0,1)^2 \tag{3.3}$$

$$\text{with } \ \varepsilon = 10^{-5}, \ \ a(x,y) = -\sin(\pi x)\cos(\pi y), \ \text{ and } \ b(x,y) = \sin(\pi y)\cos(\pi x)\,,$$

$$u(\boldsymbol{x}) = \sin(\pi x) + \sin(13\pi x) + \sin(\pi y) + \sin(13\pi y) \quad \text{on } \ \Gamma = [0,1]^2 \setminus \Omega,$$

the situation changes if standard smoothers are used, and we observe a similar coarse grid correction difficulty as in the previous subsection. If the direct $2h$-, $4h$-, etc. discretizations are applied on the coarser grids combined with standard coarsening, the two-grid convergence is limited by the factor

$$\rho_{2g}(h) \geq 1 - 2^{-p}, \tag{3.4}$$

where p denotes the order of the discretization [6]. This results in a deterioration of the V-cycle multigrid convergence on k grids given by $\rho_{kg}(h) \geq 1 - 2^{-(k-1)\cdot p}$; see above.

Remark 8 (reliability of the Fourier analysis). Dirichlet boundary effects are neglected by the Fourier analysis as it is presented in this paper. For entering flows, high-frequency boundary data may propagate far into the domain, and thus it should be taken into account by a reliable analysis. This is done by the so-called *half-space* full multigrid (FMG) analysis in [5]. For recirculating flows, however, the influence of the domain boundary is negligible in the limit of small mesh size, and the (*infinite-space*) Fourier analysis is again useful [6].

Remark 9 (Fourier analysis for operators with variable coefficients). A direct application of the Fourier analysis is not possible if we deal with operators $L_h(\boldsymbol{x})$ that are characterized by variable coefficients. However, the analysis can be applied to the locally frozen operator at a fixed grid point $\boldsymbol{\xi}$. Replacing the variable $\boldsymbol{x}$ by a constant $\boldsymbol{\xi}$, one obtains an operator $L_h(\boldsymbol{\xi})$ with *constant frozen* coefficients. In [3] it is motivated that the smoothing or two-grid factor for $L_h(\boldsymbol{x})$ can be bounded by the suprenum over the smoothing or two-grid factors for the locally frozen operators. Thus one may define the following convergence factors in the case of variable coefficients:

$$\rho_{kg}\left(h, L_h(\boldsymbol{x})\right) := \sup_{\boldsymbol{\xi}\in\Omega} \rho_{kg}\left(h, L_h(\boldsymbol{\xi})\right) \quad \text{for} \quad k = 1, 2, 3. \tag{3.5}$$

This means for (3.3) that one has to investigate the whole range of convection angles that occur in the problem. For an explicit calculation, we approximate (3.5) by a repeated application of the Fourier analysis to discretizations of $-\varepsilon\Delta u + a\,u_x + b\,u_y$ with fixed $a = \cos\beta$ and $b = \sin\beta$ for multiples of 3° until the range of possible convection angles $\beta \in [0^\circ, 360^\circ]$ is passed through, as is proposed in [23]. Then, the maximal values for $\rho_{1g}(h,\beta)$, $\rho_{2g}(h,\beta)$, $\rho_{3g}(h,\beta)$, and $\sigma_{3g}(h,\beta)$ are assumed to be upper bounds for problem (3.3).

In this section, we do not consider the acceleration by GMRES(m) in the Fourier analysis. For problems with varying coefficients the analysis from section 2.4 can be adapted in the same way as it is explained in Remark 9, but it has to be interpreted with care, as it has a more qualitative character. In the numerical experiments, however, we often observe a considerable improvement, especially for rotating flow problems.

3.3.1. First order discretization. A first order upwind discretization of (3.3) is given by the following stencil:

$$\frac{\varepsilon}{h^2}\begin{bmatrix} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{bmatrix} + \frac{1}{2h}\begin{bmatrix} -a-|a| & 2|a| & a-|a| \end{bmatrix} + \frac{1}{2h}\begin{bmatrix} b-|b| \\ 2|b| \\ -b-|b| \end{bmatrix}. \tag{3.6}$$

This discretization is studied in many papers, for instance, in [5, 14, 19, 26], where the coarse grid correction difficulty for geometric multigrid with direct coarse grid

Table 3.6
$MG4$ applied to the convection diffusion equation discretized by a first order upwind scheme, $\varepsilon = 10^{-5}$, $h = 1/256$.

Cycle	$\rho_{1g}(h,\beta)$	$\rho_{2g}(h,\beta)$	$\rho_{3g}(h,\beta)$	$\sigma_{3g}(h,\beta)$	$\rho_n(8g)$
$W(1,1)$	0.29 ($\beta = 3°$)	0.22 ($\beta = 6°$)	0.22 ($\beta = 6°$)	0.24 ($\beta = 6°$)	0.20
$W(2,1)$	0.15 ($\beta = 3°$)	0.17 ($\beta = 6°$)	0.18 ($\beta = 6°$)	0.20 ($\beta = 6°$)	0.17

discretizations is overcome by different remedies, like an overweighting of residuals [5], an additional Krylov acceleration [14], an application of AMG [19], or a special higher order choice of coarse grid discretizations [26]. Many of these approaches can be well analyzed by the Fourier three-grid analysis from section 2.

The convergence factors of the multigrid method by de Zeeuw [27] based on matrix-dependent transfer operators gets worse for discretization (3.6) with an increasing number of grids, even with a powerful relaxation like ILLU. This can be confirmed by the Fourier three-grid analysis. It is possible to improve the convergence properties with another Galerkin coarse grid operator applied in $MG4$:

- Galerkin coarse grid discretizations,
- transfer operators from the nonsymmetric blackbox multigrid by Dendy [7], and
- damped ($\omega = 0.7$) alternating zebra line Gauss–Seidel relaxation.

It was already stated in [26] that the application of this Galerkin coarsening should be useful. The symmetric prolongation, based on the symmetric part of the respective discretization operator $1/2(L_n + L_n^*)$, is similar to the well-known matrix-dependent prolongations for jumping coefficients; see, for instance, [10, 23, 27]. The nonsymmetric restriction, however, is defined as the transpose of a prolongation operator that is based on L_n^* leading to an upstream restriction. This is particularly useful because the coarse-grid operators remain upstream as well and tend to a second order compact upstream discretization. This agrees with the observation that the coarse grid operators must become higher order, at least in the cross-stream direction, to provide a good coarse grid approximation; see [26].

Table 3.6 shows the maximal one-, two-, and three-grid factors for $MG4$ with the corresponding convection angles in brackets; see Remark 9. As the two- and three-grid factors are very similar or even equal, it can be expected that the multigrid convergence for discretization (3.6) is close to these estimates, which is confirmed by the numerical reference for an 8-grid method. The maximal norm values $\sigma_{3g}(h,\beta)$ differ only slightly from the corresponding $\rho_{3g}(h,\beta)$, which indicates that the convergence speed for a single iteration is not much larger than the asymptotic convergence factor. Thus, $\rho_{3g}(h,\beta)$ is a "satisfactory" prediction for the multigrid convergence. This is observed in the numerical convergence history. Furthermore, we see that the damped alternating zebra line relaxation which is a robust smoother for the fine grid discretization [23] keeps this property for the Galerkin coarse grid discretizations resulting from the blackbox transfer operators.

3.3.2. Fourth order compact discretization. We continue with the fourth order compact discretization of (3.3) from [9]. With respect to the Fourier analysis, it is convenient to investigate the difference scheme for constant coefficients; see Remark

Table 3.7

$MG4$ is applied to the convection diffusion equation discretized by a compact fourth order scheme [9], $\varepsilon = 10^{-5}$, $h = 1/128$.

Cycle	$\rho_{1g}(h,\beta)$	$\rho_{2g}(h,\beta)$	$\rho_{3g}(h,\beta)$	$\sigma_{3g}(h,\beta)$
W(1,1)	0.53 ($\beta = 6°$)	0.59 ($\beta = 3°$)	0.77 ($\beta = 6°$)	> 20 ($\beta = 45°$)
W(2,2)	0.27 ($\beta = 6°$)	0.54 ($\beta = 9°$)	0.74 ($\beta = 9°$)	0.85 ($\beta = 18°$)

Table 3.8

The convection diffusion equation is discretized by a compact fourth order scheme [9] *using Galerkin coarsening based on the restriction from nonsymmetric blackbox multigrid* [7] *and biquintic interpolation, alternating zebra line Gauss–Seidel relaxation* ($\omega = 0.7$), $\varepsilon = 10^{-5}$, $h = 1/128$.

Cycle	$\rho_{1g}(h,\beta)$	$\rho_{2g}(h,\beta)$	$\rho_{3g}(h,\beta)$	$\sigma_{3g}(h,\beta)$
W(1,1)	0.53 ($\beta = 6°$)	0.46 ($\beta = 6°$)	0.60 ($\beta = 9°$)	0.69 ($\beta = 9°$)
W(2,2)	0.27 ($\beta = 6°$)	0.29 ($\beta = 3°$)	0.46 ($\beta = 9°$)	0.53 ($\beta = 9°$)

Cycle	$\rho_{3g}(h,\beta)$	$\sigma_{3g}(h,\beta)$
W(1,1), $\nu_1(2h) = \nu_2(2h) = 3$	0.46 ($\beta = 6°$)	0.50 ($\beta = 6°$)
W(2,2), $\nu_1(2h) = \nu_2(2h) = 4$	0.31 ($\beta = 9°$)	0.35 ($\beta = 9°$)

9, which reads in stencil notation [25]

$$\frac{\varepsilon}{6h^2}\begin{bmatrix} -1 & -4 & -1 \\ -4 & 20 & -4 \\ -1 & -4 & -1 \end{bmatrix} + \frac{1}{12h}\begin{bmatrix} a-b & -4b & -a-b \\ 4a & 0 & -4a \\ a+b & 4b & -a+b \end{bmatrix} + \frac{1}{24\varepsilon}\begin{bmatrix} ab & -2b^2 & -ab \\ -2a^2 & 4a^2+4b^2 & -2a^2 \\ -ab & -2b^2 & ab \end{bmatrix}.$$

Here, a two-grid convergence factor of 0.9375 is predicted by (3.4) for the direct coarse grid discretizations. Applying $MG4$ to this discretization, we find a remarkable improvement of the two-grid factor compared to the standard variant, but the coarse grid problem is not really solved, as is indicated by the increase of the two- and three-grid factors compared to the one-grid values; see Table 3.7. This can be explained by considering the orders of the transfer operators in a Galerkin process for singularly perturbed problems [26]. The sum of these orders ($m_r + m_p$, where the subscripts are abbreviations for restriction and prolongation, respectively) should be greater than the order of the differential equation M plus the order of the discretization p, i.e.,

$$m_r + m_p > M + p. \tag{3.7}$$

This requirement is fulfilled for the multigrid method from Table 3.6 for the first order discretization of (3.3) ($m_r = 1, m_p = 2, M = 1, p = 1$) but is violated in Table 3.7 ($m_r = 1, m_p = 2, M = 1, p = 4$). Note that we set $M = 1$ in this situation because of the dominating convection term. It is already indicated by the large value of σ_{3g} for the W(1,1)-cycle from Table 3.7 that this multigrid method has to be handled with care.

We keep the restriction from $MG4$ in order to maintain the upstream properties of the discretizations on the coarser grids (see [7]) but replace the prolongation by biquintic interpolation ($m_p = 6$). Thus, the above rule (3.7) is satisfied, which leads to a considerable improvement of the multigrid method, as is shown in Table 3.8. Regarding the norm values $\sigma_{3g}(h,\beta)$, one obtains a reliable multigrid method. However, although the one- and two-grid factors are similar, the three-grid factors still increase in the upper part of Table 3.8. This is due to a deterioration of the smoothing property on the coarse grids because there we have larger stencils resulting from the very accurate interpolation in the Galerkin process. The three-grid convergence factors can be improved if we increase the numbers $\nu_1(2h)$ and $\nu_2(2h)$ of pre- and postsmoothing steps on G_{2h}, as is shown in the lower part of Table 3.8. In this way, it is possible to recover the two-grid factors, which indicates that the coarse grid correction difficulty is solved. From section 3.3.1, it can be expected that the analytical estimates match closely with numerical test calculations, and also for the fourth order discretization. Because of the large stencils, however, this multigrid method cannot be recommended for a practical implementation despite its good (regarding the highly accurate discretization) convergence behavior.

4. Conclusions. We have presented Fourier three-grid analysis for multigrid as a solver and as a preconditioner for GMRES(m). Applying this analysis, it is possible to obtain accurate convergence estimates for elliptic operators and to predict the performance of V- and W-cycles or pre- and postsmoothing. For singularly perturbed problems with coarse grid correction difficulties, the three-grid analysis yields additional valuable insight into the nature of the multigrid convergence problem and allows for an investigation of the benefits of possible remedies, for example, based on certain choices of Galerkin coarse grid discretizations. One can verify the qualitative rule concerning the order of the transfer operators in a Galerkin process for singularly perturbed problems from [26] by quantitative asymptotic convergence and norm estimates.

REFERENCES

[1] D. Braess, *Towards algebraic multigrid for elliptic problems of second order*, Computing, 55 (1995), pp. 379–393.

[2] A. Brandt, *Multi-level adaptive solutions to boundary-value problems*, Math. Comp., 31 (1977), pp. 333–390.

[3] A. Brandt, *Rigorous local mode analysis of multigrid*, in Preliminary Proceedings of the 4th Copper Mountain Conference on Multigrid Methods, Copper Mountain, CO, 1989, pp. 55–133.

[4] A. Brandt, *Rigorous quantitative analysis of multigrid,* I. *Constant coefficients two-level cycle with L_2-norm*, SIAM J. Numer. Anal., 31 (1994), pp. 1695–1730.

[5] A. Brandt and I. Yavneh, *On multigrid solution of high-Reynolds incompressible entering flows*, J. Comput. Phys., 101 (1992), pp. 151–164.

[6] A. Brandt and I. Yavneh, *Accelerated multigrid convergence and high-Reynolds recirculating flows*, SIAM J. Sci. Comput., 14 (1993), pp. 607–626.

[7] J. E. Dendy, Jr., *Blackbox multigrid for nonsymmetric problems*, Appl. Math. Comput., 13 (1983), pp. 261–283.

[8] B. Fischer, A. Ramage, D. J. Silvester, and A. J. Wathen, *On parameter choice and iterative convergence for stabilised discretisations of advection-diffusion problems*, Comput. Methods Appl. Mech. Engrg., 179 (1999), pp. 179–195.

[9] M. M. Gupta, R. P. Manohar, and J. W. Stephenson, *A single cell high order scheme for the convection-diffusion equation with variable coefficients*, Internat. J. Numer. Methods Fluids, 4 (1984), pp. 641–651.

[10] W. Hackbusch, *Multi-Grid Methods and Applications*, Springer-Verlag, Berlin, 1985.

[11] P. W. Hemker, *Fourier Analysis of Grid Functions, Prolongations and Restrictions*, Report NW 98, CWI, Amsterdam, 1980.

[12] P. W. Hemker, *On the order of prolongations and restrictions in multigrid procedures*, J. Comput. Appl. Math., 32 (1990), pp. 423–429.

[13] C. W. Oosterlee, F. J. Gaspar, T. Washio, and R. Wienands, *Multigrid line smoothers for higher order upwind discretizations of convection-dominated problems*, J. Comput. Phys., 139 (1998), pp. 274–307.

[14] C. W. Oosterlee and T. Washio, *Krylov subspace acceleration of nonlinear multigrid with application to recirculating flows*, SIAM J. Sci. Comput., 21 (2000), pp. 1670–1690.

[15] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[16] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing, Boston, 1996.

[17] Y. Saad, *Further analysis of minimum residual iterations*, Numer. Linear Algebra Appl., 7 (2000), pp. 67–93.

[18] R. Stevenson, *On the Validity of Local Mode Analysis of Multi-Grid Methods*, Ph.D. thesis, Rijks University, Utrecht, the Netherlands, 1990.

[19] K. Stüben, *Algebraic Multigrid (AMG): An Introduction with Applications*, Tech. report 53, GMD, St. Augustin, Germany, 1999. This report appeared as an appendix in [21].

[20] K. Stüben and U. Trottenberg, *Multigrid methods: Fundamental algorithms, model problem analysis and applications*, in Multigrid Methods, Lecture Notes in Math. 960, W. Hackbusch and U. Trottenberg, eds., Springer-Verlag, Berlin, 1982, pp. 1–176.

[21] U. Trottenberg, C. W. Oosterlee, and A. Schüller, *Multigrid*, Academic Press, New York, 2001.

[22] P. Vanek, J. Mandel, and M. Brezina, *Algebraic Multigrid on Unstructured Meshes*, UCD/CCM Report 34, University of Colorado, Denver, CO, 1994.

[23] P. Wesseling, *An Introduction to Multigrid Methods*, John Wiley, Chichester, UK, 1992.

[24] R. Wienands, C. W. Oosterlee, and T. Washio, *Fourier analysis of GMRES(m) preconditioned by multigrid*, SIAM J. Sci. Comput., 22 (2000), pp. 582–603.

[25] I. Yavneh, *Analysis of a fourth-order compact scheme for convection-diffusion*, J. Comput. Phys., 133 (1997), pp. 361–364.

[26] I. Yavneh, *Coarse-grid correction for nonelliptic and singular perturbation problems*, SIAM J. Sci. Comput., 19 (1998), pp. 1682–1699.

[27] P. M. de Zeeuw, *Matrix-dependent prolongations and restrictions in a blackbox multigrid solver*, J. Comput. Appl. Math., 33 (1990), pp. 1–27.

SIAM J. SCI. COMPUT.
Vol. 23, No. 2, pp. 672–682

SOLUTION METHODS FOR THE POISSON EQUATION WITH CORNER SINGULARITIES: NUMERICAL RESULTS*

ZHIQIANG CAI†, SEOKCHAN KIM‡, AND BYEONG-CHUN SHIN§

Abstract. In [Z. Cai and S. Kim, *SIAM J. Numer. Anal.,* 39 (2001), pp. 286–299], we developed a new finite element method using singular functions for the Poisson equation on a polygonal domain with re-entrant angles. Such a method first computes the regular part of the solution, then the stress intensity factor, and finally the solution itself. This paper studies solution methods for solving the system of linear equations arising from the discretization and focuses on numerical results including the finite element accuracy and the multigrid performance.

Key words. corner singularity, finite element, multigrid

AMS subject classifications. 65F10, 65F30, 65N55

PII. S1064827500372778

1. Introduction.

Solutions of elliptic boundary value problems on a domain with corners have singular behavior near the corners. This occurs even when data of the underlying problem are very smooth. Such singular behavior affects the accuracy of the finite element method throughout the whole domain. In [5], we developed a new finite element method using singular and dual singular functions for the Poisson equation with homogeneous Dirichlet boundary conditions on a polygonal domain with re-entrant angles. By using the dual singular function and a cut-off function with a bigger support, we derived a new extraction formula for the stress intensity factor λ in terms of the regular part w of the solution u. This enables us to deduce a well-posed problem for w, which is then approximated by the continuous piecewise linear finite element method. Approximations to the intensity factor λ and the solution u are straightforward, so we concentrate in this paper on the computation of w. It was shown in [5] that we achieve $O(h)$ optimal accuracy for w and u in the H^1 norm. We established the $O(h^{1+\frac{\pi}{\omega}})$ error bound for w in L^2, which, in turn, implies the same error bounds for u in L^2 and for λ in the absolute value, where ω is the internal angle. This error bound for w in L^2 is not optimal. The reason is the simplified adjoint problem used in the duality argument, which does not have full regularity. But w is H^2 regular; it is then interesting to see if such error bound is sharp numerically.

The problem determining w is no longer the nice Poisson equation but is perturbed by integral terms which are only nonzero on a strip away from the corner. Because of such a perturbation, the problem is nonsymmetric and probably indefinite. To solve the nonsymmetric algebraic equations arising from the discretization, we observe that the perturbation is rank one and that its pseudodifferential order is -1. The

*Received by the editors May 31, 2000; accepted for publication (in revised form) October 25, 2000; published electronically July 10, 2001.

http://www.siam.org/journals/sisc/23-2/37277.html

†Department of Mathematics, Purdue University, 1395 Mathematical Science Building, West Lafayette, IN 47907-1395 (zcai@math.purdue.edu). This author was sponsored in part by the National Science Foundation under grants DMS-9619792 and INT-9910010.

‡Department of Applied Mathematics, Changwon National University, Changwon, 641-773, Republic of Korea (sckim@sarim.changwon.ac.kr). This author was supported by the basic research program of the Korea Science and Engineering Foundation under grant KOSEF 1999-2-103-0002-3.

§Department of Mathematics, Ajou University, Suwon 442-749, Korea (bcshin@math.purdue.edu). This auther was supported by the postdoctoral fellowships program of the Korea Science and Engineering Foundation (KOSEF).

former leads to the first approach which uses the Sherman–Morrison (SM) formula (see section 4.1). This approach requires *two* (approximate) inversions of the discrete Laplace operator, which will be done by multigrid (MG) methods. It is well known (see, e.g., [10, 1]) that the MG for the discrete Poisson problem has a fast convergence rate. The latter suggests that the nonsymmetric perturbation is well controlled by the Laplace operator whose pseudodifferential order is 2, and, hence, MG can be applied efficiently. In particular, the second approach adopted in this paper is the V-cycle MG that uses the exact coarsest grid solver and in which smoothing operators depend only on the discrete Laplace operator. Application of the convergence theory in [6] guarantees that the MG $V(1, 0)$ cycle and, hence, the $V(\nu_1, \nu_2)$ cycles for $\nu_1 \geq 1$ and $\nu_2 \geq 0$ converge uniformly in the finest mesh size and the number of refinement levels, provided that the coarsest mesh size is sufficiently small. This condition indicates that the direct MG is probably expensive because it requires us to solve a relatively large coarse grid problem. The coarsest mesh size used in our experiments for both approaches is 1/4. Hence, it is clear that such a condition is not essential for our problem.

The present paper attempts to confirm theories numerically on the accuracy of finite element approximation and the performance of solution methods. For the Dirichlet problem on the L-shape domain, we show here that finite element approximations using continuous piecewise linear elements converge to the exact solution with $O(h^2)$ errors in the *discrete* H^1 seminorm and L^2 norm, respectively. The theoretically predicted error bounds are $O(h)$ in the H^1 norm and $O(h^{\frac{5}{3}})$ in the L^2 norm. We appear to have obtained superconvergence in the discrete H^1 seminorm for this particular case. We seem to achieve the optimal L^2 accuracy, which is better than what is predicted by our theory. We show that the SM formula using the MG V-cycles and the direct MG V-cycle converge independently of the finest mesh size h and the number of refinement levels. We also show that the full MG computes a final approximation with accuracy on the order of a discretization error in a total amount of work equal to about nine relaxation sweeps on the finest grid.

The paper is organized as follows. The finite element method and related results from [5] are introduced in sections 2 and 3, and the solution methods are described in section 4. We present the results of numerical experiments in section 5. Some conclusions and final remarks are contained in section 6.

2. The problem and preliminaries.

As a model problem, we consider the Poisson equation with homogeneous Dirichlet boundary conditions

$$\left\{ \begin{array}{rcll} -\Delta u & = & f & \text{in } \Omega, \\ u & = & 0 & \text{on } \partial\Omega, \end{array} \right. \tag{2.1}$$

where f is a given function in $L^2(\Omega)$ and Ω is an open, bounded polygonal domain in $\mathcal{R}^2$ with one re-entrant angle. Extension to the domain with the finite number of re-entrant angles is straightforward. We use the standard notation and definition for the Sobolev spaces $H^t(B)$ for $t \geq 0$; the standard associated inner products are denoted by $(\cdot, \cdot)_{t,B}$, and their respective norms and seminorms are denoted by $\| \cdot \|_{t,B}$ and $| \cdot |_{t,B}$. The space $L^2(B)$ is interpreted as $H^0(B)$, in which case the inner product and norm will be denoted by $(\cdot, \cdot)_B$ and $\| \cdot \|_B$, respectively.

Let ω be the internal angle of Ω satisfying $\pi < \omega < 2\pi$. Without loss of generality, assume that the corresponding vertex is at the origin. Define the *singular* and the *dual singular* functions by

$$s = r^{\frac{\pi}{\omega}} \sin \frac{\pi\theta}{\omega} \quad \text{and} \quad s_{-} = r^{-\frac{\pi}{\omega}} \sin \frac{\pi\theta}{\omega}, \tag{2.2}$$

respectively, in the polar coordinates (r, θ). The coordinates are chosen at the origin so that the internal angle ω is spanned by the two half-lines $\theta = 0$ and $\theta = \omega$. Set

$$B(r_1; r_2) = \{(r, \theta) : \ r_1 < r < r_2 \text{ and } 0 < \theta < \omega\} \cap \Omega \quad \text{and} \quad B(r_1) = B(0; r_1).$$

Define a family of cut-off functions of r, $\eta_\rho(r)$ as follows:

$$\eta_\rho(r) = \begin{cases} 1 & \text{in } B(\frac{\rho R}{2}), \\ \frac{15}{16}\Big\{\frac{8}{15} - \big(\frac{4r}{\rho R} - 3\big) + \frac{2}{3}\big(\frac{4r}{\rho R} - 3\big)^3 - \frac{1}{5}\big(\frac{4r}{\rho R} - 3\big)^5\Big\} & \text{in } \bar{B}(\frac{\rho R}{2};\, \rho R), \\ 0 & \text{in } \Omega \setminus \bar{B}(\rho R), \end{cases} \tag{2.3}$$

where ρ is a parameter in (0, 2] and $R \in \mathcal{R}$ is a fixed number so that the $\eta_2 s$ vanishes identically on $\partial\Omega$. It is well known (see, e.g., [8, 9]) that the solution of (2.1) has the singular function representation

$$u = w + \lambda \eta_\rho s, \tag{2.4}$$

where $w \in H^2(\Omega) \cap H_0^1(\Omega)$ is the regular part of the solution and $\lambda \in \mathcal{R}$ is the so-called *stress intensity factor*.

In [5], by using a new *extraction formula* in terms of w,

$$\lambda = \frac{1}{\pi}(w, \Delta(\eta_2 s_{-}))_{B(R;2R)} + \frac{1}{\pi}(f, \eta_2 s_{-})_{B(2R)}, \tag{2.5}$$

we are able to deduce a well-posed integro-differential equation of w,

$$-\Delta w - \frac{1}{\pi}(w, \Delta(\eta_2 s_{-}))_{B(R;2R)} \Delta(\eta_\rho s) = f + \frac{1}{\pi}(f, \eta_2 s_{-})_{B(2R)} \Delta(\eta_\rho s) \quad \text{in } \Omega, \tag{2.6}$$

where $0 < \rho \le 1$. Its variational formulation is to find $w \in H_0^1(\Omega)$ such that

$$a(w, v) = g(v) \quad \forall\, v \in H_0^1(\Omega), \tag{2.7}$$

where the bilinear form $a(\cdot, \cdot)$ and linear form $g(\cdot)$ are defined by

$$a(w, v) = a^s(w, v) + b(w, v), \quad b(w, v) = -\frac{1}{\pi}(w, \Delta(\eta_2 s_{-}))_{B(R;2R)} (\Delta(\eta_\rho s), v)_{B(\frac{\rho R}{2};\rho R)},$$
$$a^s(w, v) = (\nabla w, \nabla v)_\Omega, \quad \text{and} \quad g(v) = (f, v)_\Omega + \tfrac{1}{\pi}(f, \eta_2 s_{-})_{B(2R)} (\Delta(\eta_\rho s), v)_{B(\frac{\rho R}{2};\rho R)}.$$

Note that the bilinear form $b(\cdot, \cdot)$ is not symmetric. It was shown in [5] that (2.7) has a unique solution $w \in H_0^1(\Omega) \cap H^2(\Omega)$. The following continuity and coercivity bounds were also established:

$$|a^s(w, v)| \le C\, |w|_{1,\Omega}\, |v|_{1,\Omega}, \quad |b(w, v)| \le C\, \|w\|_\Omega\, \|v\|_\Omega \quad \forall\, w,\ v \in H_0^1(\Omega), \tag{2.8}$$

$$\text{and} \quad |v|_{1,\Omega}^2 - C\|v\|_\Omega^2 \le a(v, v) \quad \forall\, v \in H_0^1(\Omega). \tag{2.9}$$

Here and henceforth, we use C with or without script to denote a generic positive constant independent of the mesh size h and the number of levels J introduced in section 4.

3. Finite element approximation. Let $\mathcal{T}_h$ be a partition of the domain Ω into triangular finite elements; i.e., $\Omega = \cup_{K\in\mathcal{T}_h} K$ with $h = \max\{\operatorname{diam} K\colon\ K \in \mathcal{T}_h\}$. Assume that the triangulation $\mathcal{T}_h$ is regular. Set

$$\mathcal{V}_h = \{\phi_h \in C^0(\Omega)\colon\ \phi_h|_K \text{ is linear, } \forall\ K \in \mathcal{T}_h,\ \phi_h = 0 \text{ on } \partial\Omega\} \subset H_0^1(\Omega).$$

Then the finite element approximation to (2.7) in $H_0^1(\Omega)\cap H^2(\Omega)$ becomes as follows: find $w_h \in \mathcal{V}_h$ such that

$$a(w_h, v) = g(v) \quad \forall\ v \in \mathcal{V}_h. \tag{3.1}$$

Approximations to the stress intensity factor and the solution of (2.1) can be computed according to (2.5) and (2.4) as follows:

$$\lambda_h = \frac{1}{\pi}(w_h, \Delta(\eta_2 s_-))_{B(R;2R)} + \frac{1}{\pi}(f, \eta_2 s_-)_{B(2R)} \tag{3.2}$$

$$\text{and} \quad u_h = w_h + \lambda_h \eta_\rho s. \tag{3.3}$$

The following error estimates were established in [5].

THEOREM 3.1. *Assume that* $w \in H^2(\Omega)$ *and* u *are the solutions of* (2.7) *and* (2.1), *respectively. Then the following error estimates hold:*

$$\|w - w_h\|_{1,\Omega},\ \|u - u_h\|_{1,\Omega} \le C\, h \|f\|_\Omega, \tag{3.4}$$

$$\textit{and} \quad \|w - w_h\|_\Omega,\ \|u - u_h\|_\Omega,\ |\lambda - \lambda_h| \le C\, h^{1+\frac{\pi}{\omega}} \|f\|_\Omega. \tag{3.5}$$

4. Solution methods. Let $\{\phi_j(x)\}_{j=1}^N$ be nodal bases for $\mathcal{V}_h$, write $w_h = \sum_{j=1}^N w_j^h \phi_j(x)$, and then the matricial form of (3.1) is

$$\sum_{j=1}^{N} w_j^h a(\phi_j,\ \phi_i) = g(\phi_i) \tag{4.1}$$

for $i = 1, \dots, N$. Let A_h^s and B_h be $N \times N$ matrices with $(i,\ j)$ entries $a^s(\phi_j,\ \phi_i)$ and $b(\phi_j,\ \phi_i)$, respectively, let W_h be the unknown vector with components w_i^h, and let F_h be the right-hand side vector with components $g(\phi_i)$. Then (4.1) may be written as

$$A_h W_h = F_h, \tag{4.2}$$

where $A_h = A_h^s + B_h$. Note that B_h is a nonsymmetric and rank-one matrix. In this section, we discuss two approaches for solving the system of linear equations in (4.2). The first approach is to use the SM formula which requires two inversions of the discrete Laplace operator A_h^s. The second approach is the application of the MG V-cycle with smoothers depending only on A_h^s.

4.1. The SM formula. Any rank-one matrix B_h can be written as the product of the column and row vectors. In particular, we have

$$B_h = -UV^t,$$

where U and V are column vectors with ith components

$$U_i = \frac{1}{\pi}(\Delta(\eta_\rho s), \phi_i)_{B(\frac{\rho R}{2};\rho R)} \quad \text{and} \quad V_i = (\phi_i,\ \Delta(\eta_2 s_-))_{B(R;\,2R)},$$

respectively. By the SM formula, the inverse of A_h has the form

$$A_h^{-1} = (A_h^s)^{-1} + \alpha\,(A_h^s)^{-1}UV^t(A_h^s)^{-1}, \quad \text{where} \quad \alpha = \frac{1}{1 - V^t(A_h^s)^{-1}U}.$$

Therefore, the solution of the system of linear equations in (4.2), $W_h = A_h^{-1}F_h$, can be computed by the following algorithm.

1. Solve $A_h^s X = F_h$ for X.
2. Solve $A_h^s Y = U$ for Y.
3. Compute $\alpha = \frac{1}{1-V^tY}$ and $\beta = V^tX$.
4. Set $W_h = X + \alpha\,\beta\,Y$.

Note that the above algorithm requires *two* (approximate) inversions of the discrete Poisson equations, which will be done by using MG. It is well known that MG applied to the discrete Poisson equation has a fast convergence rate.

4.2. MG algorithms. In this subsection, we describe the MG for solving the discrete problem in (4.2); the MG uses the exact coarsest grid solver, and smoothing operators on the fine grids depend only on A_h^s. To this end, we start with an intentionally coarse triangulation $\mathcal{T}_0$ of $\bar{\Omega}$ with the properties that the boundary $\partial\Omega$ is composed of edges of some triangles K in $\mathcal{T}_0$ and that every triangle of $\mathcal{T}_0$ is shape regular. Each triangle K of $\mathcal{T}_0$ is regularly refined several times, giving a family of nested triangulations $\mathcal{T}_0, \mathcal{T}_1, \dots, \mathcal{T}_J \equiv \mathcal{T}$, such that each triangle of $\mathcal{T}_{k+1}$ is generated by subdividing a triangle of $\mathcal{T}_k$ into four congruent triangles. Let h_k be the mesh size for the corresponding triangulation $\mathcal{T}_k$; we then have $h_k = 2h_{k+1}$. Denote by h the mesh size of the finest grid. We associate the triangulation $\mathcal{T}_k$ with the continuous piecewise linear finite element space $\mathcal{V}_k$. It is easy to see that the family of spaces $\{\mathcal{V}_k\}$ is nested.

For $k = 0, 1, \dots, J$, define the nonsymmetric and symmetric elliptic operators $\mathcal{A}_k$, $\mathcal{A}_k^s : \mathcal{V}_k \longrightarrow \mathcal{V}_k$ by

$$(\mathcal{A}_k w, \phi) = a(w, \phi) \quad \forall\, \phi \in \mathcal{V}_k \quad \text{and} \quad (\mathcal{A}_k^s w, \phi) = a^s(w, \phi) \quad \forall\, \phi \in \mathcal{V}_k,$$

respectively. Also, define the L^2-projection operator $\mathcal{P}_k : \mathcal{V}_{k+1} \longrightarrow \mathcal{V}_k$ by

$$(\mathcal{P}_k w, \phi) = (w, \phi) \quad \forall\, \phi \in \mathcal{V}_k.$$

The smoothing operators $\mathcal{R}_k : \mathcal{V}_k \longrightarrow \mathcal{V}_k$ adopted in this paper are the exact solver on the coarsest grid and the Gauss–Seidel iteration based on the symmetric operator $\mathcal{A}_k^s$ for $k = 1, 2, \dots, J$. Now we define the MG operator $\mathcal{B}_k : \mathcal{V}_k \longrightarrow \mathcal{V}_k$ of the V(ν_1, ν_2)-cycle by induction (see [1]).

V(ν_1, ν_2)-Algorithm. *Set $\mathcal{B}_0 = \mathcal{A}_0^{-1}$. Assume that $\mathcal{B}_{k-1}$ has been defined, and define $\mathcal{B}_k\, g$ for $g \in \mathcal{V}_k$ as follows.*

1. *Set an initial guess x_0.*
2. *Define x^ℓ for $\ell = 1, \dots, \nu_1$ by $x^\ell = x^{\ell-1} + \mathcal{R}_k(g - \mathcal{A}_k x^{\ell-1})$.*
3. *Define x^{ν_1+1} by $x^{\nu_1+1} = \mathcal{B}_{k-1}\mathcal{P}_{k-1}(g - \mathcal{A}_k x^{\nu_1})$.*
4. *Set $\mathcal{B}_k g = x^{\nu_1+\nu_2+1}$, where x^ℓ is defined for $\ell = \nu_1 + 2, \dots, \nu_1 + \nu_2 + 1$ by $x^\ell = x^{\ell-1} + \mathcal{R}_k(g - \mathcal{A}_k x^{\ell-1})$.*

It was shown in [6] that if the bilinear forms $a(\cdot,\cdot)$ and $b(\cdot,\cdot)$ satisfy the continuity and coercivity bounds in (2.8) and (2.9), then the MG $V(1,0)$-cycle and, hence, the $V(\nu_1, \nu_2)$-cycles for $\nu_1 \geq 1$ and $\nu_2 \geq 0$ converge independently of the mesh size h and the number of levels J, provided that the coarsest mesh size h_0 is sufficiently small. This condition on h_0 indicates that this direct MG approach is probably expensive because it requires us to solve a relatively large coarse grid problem. Numerical experiments presented in the next section use $h_0 = 1/4$. Hence, it is clear that such a condition is not essential for our problem.

Finally, we will also consider the full MG scheme FMG(nV(ν_1, ν_2)), which proceeds from the coarsest to the finest grid, invoking n $V(\nu_1, \nu_2)$ cycles on each level along the way with $n \geq 1$ (see [4]). Thus, each coarse level serves to provide a good initial approximation to the next finer level, with the intent of producing a final approximation on the finest grid that is accurate to the level of discretization error.

5. Numerical experiments.

In this section, we present the results of numerical experiments. Performance of the solution methods and the finite element discretization accuracy are first analyzed individually, and then the overall FMG performance is assessed.

We consider the Poisson equation in (2.1) over the L-shaped domain

$$\Omega = ((-1, 1) \times (-1, 1)) \setminus ([0, 1) \times (-1, 0]) .$$

The domain Ω is first partitioned into $3(n \times n)$ square subdomains of side length $h = 1/n$. The $3(n\times n)$ subsquares are then divided into pairs of triangles by connecting the bottom right and upper left corners. We use continuous piecewise linear finite element space for the approximation of the regular part w of the solution with respect to the triangulation with the grid interval h ranging from 2^{-2} to 2^{-7} for each direction. In the performances of the MG V-cycles and the full MG V-cycles, the mesh size of the coarsest grid is chosen to be $h_0 = 2^{-2}$. To increase the accuracy of numerical quadratures, all integrations, involving singular and dual singular functions, on a given triangle $K \in \mathcal{T}_k$ are computed by the composite three points quadrature rule using small triangles of the side length 2^{-11}. These small triangles are generated by $(9 - k)$-times subdividing the triangles K of the side length $2^{-(k+2)}$. We will use the following discrete L^2-norm and H^1-seminorm for the vector $V_h \in \Re^N$:

$$\|V_h\|_h := \sqrt{\left(M_h V_h, V_h\right)} \quad \text{and} \quad |V_h|_{1,h} := \sqrt{\left(A_J^s V_h, V_h\right)},$$

where M_h is the mass matrix associating with $\mathcal{V}_h$ and A_J^s is the stiffness matrix of the Laplace operator defined in section 4.

5.1. Performance of the solution methods.

To study the performance of the solution methods, we begin with the Poisson equation in (2.1) with the homogeneous right-hand side; i.e.,

$$\begin{cases} -\Delta u &= 0 \quad \text{in } \Omega, \\ \quad u &= 0 \quad \text{on } \partial\Omega. \end{cases}$$

It is obvious that the exact solution is zero. With the initial guess of one, we report asymptotic convergence factors for the MG V-cycle defined in section 4.2 (referring to the direct MG) and the iterative Sherman–Morrison algorithm (SMA). Let X^k and Y^k denote, respectively, the kth V(ν_1,ν_2) iterates of the discrete Poisson problems in the first two steps of the SMA defined in section 4.1. The iterative SMA is defined as follows:

$$W_h^k = X^k + \alpha^k \beta^k Y^k$$

TABLE 1
Convergence factors for direct MG V(ν_1, ν_2) cycle.

	$\rho = 1.00$			$\rho = 0.50$		
Mesh size	V(1,0)	V(2,0)	V(1,1)	V(1,0)	V(2,0)	V(1,1)
$h = \frac{1}{8}$	0.3971	0.1822	0.1830	0.3988	0.1720	0.1694
$h = \frac{1}{16}$	0.4415	0.2460	0.2369	0.4564	0.2793	0.2758
$h = \frac{1}{32}$	0.4417	0.2696	0.2524	0.4504	0.2574	0.2866
$h = \frac{1}{64}$	0.4642	0.3113	0.2924	0.4360	0.2691	0.2598
$h = \frac{1}{128}$	0.4719	0.3081	0.2612	0.4777	0.3242	0.2831

TABLE 2
Convergence factors for the iterative SMA with $\rho = 1.0$.

Mesh size	V(1,0)	V(2,0)	V(1,1)
$h = \frac{1}{8}$	0.4005	0.1709	0.1691
$h = \frac{1}{16}$	0.4588	0.2722	0.2780
$h = \frac{1}{32}$	0.4563	0.2421	0.2680
$h = \frac{1}{64}$	0.4638	0.2285	0.2553
$h = \frac{1}{128}$	0.4369	0.3072	0.2474

with $\alpha^k = \frac{1}{1-V^tY^k}$ and $\beta^k = V^tX^k$. Convergence factors σ_h^k here are defined as ratios of a successive discrete H^1-seminorm of errors,

$$\sigma_h^k := \frac{|e_h^{k+1}|_{1,h}}{|e_h^k|_{1,h}} \qquad \text{with} \quad e_h^k = W_h - W_h^k,$$

where $W_h = 0$ and W_h^k are the exact solution and the kth iterate of (4.2) with $f = 0$, respectively. It is clear that the convergence factor of the iterative SMA does not depend on ρ and the mesh size h since the MG iterations apply to two discrete Poisson problems.

Tables 1 and 2 represent convergence factors measured after 20 MG V(ν_1, ν_2) cycles for the direct MG and the iterative SMA, respectively. Uniform convergence theory of the direct MG (see the discussion in section 4.2) requires the condition that h_0 is sufficiently small. Clearly, the numerical results reported here show that convergence of the direct MG is not subject to such a constraint since we used $h_0 = 1/4$. The observed convergence factors in Tables 1 and 2 for the direct MG and the iterative SMA are almost same, but the latter costs about twice as many arithmetic operations as the former for one re-entrant corner and much more for several re-entrant corners. This indicates that the direct MG is the method of choice. One V$(2,0)$ or one V$(1,1)$ cycle costs about twice as much as one V$(1,0)$ cycle. Table 1 shows that the reduction rate by two V$(1,0)$ cycles is better than the reduction rate of one V$(2,0)$ or V$(1,1)$ cycle. This suggests that the MG V$(1,0)$ is the most effective solver, which is confirmed by the subsequent tables.

5.2. Discretization errors and FMG performance. To measure the discretization error, we consider the Poisson equation in (2.1) with a known nonzero

TABLE 3
The number of iterations for the direct MG till $|W_h - W_h^k|_{1,h} < 10^{-4}$.

		$\rho = 1.00$			$\rho = 0.50$		
Mesh size	Dim.	V(1,0)	V(2,0)	V(1,1)	V(1,0)	V(2,0)	V(1,1)
$h = \frac{1}{4}$	33	1	1	1	1	1	1
$h = \frac{1}{8}$	161	10	7	6	10	7	6
$h = \frac{1}{16}$	705	11	8	7	11	7	7
$h = \frac{1}{32}$	2945	12	8	7	11	8	7
$h = \frac{1}{64}$	12033	12	8	7	11	8	7
$h = \frac{1}{128}$	48641	12	8	7	12	8	7

TABLE 4
The number of iterations till $|W_h - W_h^k|_1 < 10^{-4}$ *for* $\rho = 1.0$ *(the iterative SMA).*

Mesh Size	Dim.	V(1,0)	V(2,0)	V(1,1)
$h = \frac{1}{4}$	33	1	1	1
$h = \frac{1}{8}$	161	10	7	6
$h = \frac{1}{16}$	705	11	7	7
$h = \frac{1}{32}$	2945	11	8	7
$h = \frac{1}{64}$	12033	11	8	7
$h = \frac{1}{128}$	48641	11	8	7

solution by choosing the right-hand side function to be

$$f = \begin{cases} \sin(2\pi x)\big[2\pi^2(y^2+2y)(y^2-1) - (6y^2+6y-1)\big] - \Delta(\eta_\rho s) & \text{if } \ -1 \le y < 0, \\ \sin(2\pi x)\big[2\pi^2(-y^2+2y)(y^2-1) - (-6y^2+6y+1)\big] - \Delta(\eta_\rho s) & \text{if } \ 0 < y \le 1, \end{cases}$$

where $s = r^{\frac{2}{3}} \sin(\frac{2\theta}{3})$ is the singular function. The exact solution of the underlying problem is then

$$u = w + \eta_\rho s,$$

where w is the regular part of the solution, which is the exact solution of (2.7) given by

$$w = \begin{cases} \sin(2\pi x)(\frac{1}{2}y^2 + y)(y^2-1) & \text{if } \ -1 \le y \le 0, \\ \sin(2\pi x)(-\frac{1}{2}y^2 + y)(y^2-1) & \text{if } \ 0 \le y \le 1. \end{cases}$$

Note that the function w is in $H^2(\Omega)$ but not in $H^3(\Omega)$. Let W_h be the exact solution of (4.2), the coefficient vector of the finite element approximation. We first depict the number of iterations required in order to reduce the initial error $|W_h - W_h^0|_{1,h} = |W_h|_{1,h}$ within the given tolerance $\varepsilon = 10^{-4}$ in Tables 3 and 4.

Next, we compute the approximate solution $\hat{W}_h$ of (4.2) on each of the various levels of discretization by using 30 direct MG $V(1,0)$ cycles. This is to ensure that the errors reported in Tables 5 and 6 properly reflect discretization accuracy without contamination from algebraic iteration errors. Let w_h^I be the interpolant of w in $\mathcal{V}_h = \text{span}\{\phi_i(x)\}_{i=1}^N$; i.e.,

$$w_h^I = \sum_{i=1}^N w_j^I \phi_i(x),$$

Table 5
The discrete L^2-norm errors and the convergence rates for w.

	$\rho = 1.00$		$\rho = 0.50$	
Mesh size	L^2-norm	Rate	L^2-norm	Rate
$h = \frac{1}{4}$	2.5729e-02		2.2269e-02	
$h = \frac{1}{8}$	7.5903e-03	1.7612	6.6963e-03	1.7336
$h = \frac{1}{16}$	2.0062e-03	1.9197	1.7669e-03	1.9221
$h = \frac{1}{32}$	5.0926e-04	1.9780	4.4820e-04	1.9790
$h = \frac{1}{64}$	1.2774e-04	1.9952	1.1238e-04	1.9958
$h = \frac{1}{128}$	3.1850e-05	2.0038	2.8019e-05	2.0039

Table 6
The discrete H^1 seminorm and the convergence rates for w.

	$\rho = 1.00$		$\rho = 0.50$	
Mesh size	H^1-norm	Rate	H^1-norm	Rate
$h = \frac{1}{4}$	1.6457e-01		1.6027e-01	
$h = \frac{1}{8}$	4.7468e-02	1.7937	4.5217e-02	1.8256
$h = \frac{1}{16}$	1.2568e-02	1.9172	1.1800e-02	1.9381
$h = \frac{1}{32}$	3.1938e-03	1.9764	3.0068e-03	1.9725
$h = \frac{1}{64}$	8.0209e-04	1.9934	7.5437e-04	1.9949
$h = \frac{1}{128}$	2.0121e-04	1.9951	1.8791e-04	2.0052

where w_i^I is the nodal value of w. Let W_I be the coefficient vector of w_h^I, i.e., the vector with components w_i^I. Define the error vector by

$$E(h) = W_I - \hat{W}_h.$$

The rates of convergence for discretization errors in the discrete L^2 norm, the discrete H^1 seminorm, and the absolute value are measured by

$$\log_2 \frac{\|E(h)\|_h}{\|E(\frac{h}{2})\|_{\frac{h}{2}}}, \quad \log_2 \frac{\|E(h)\|_{1,h}}{\|E(\frac{h}{2})\|_{1,\frac{h}{2}}}, \quad \text{and} \quad \log_2 \frac{|\lambda - \lambda_h|}{|\lambda - \lambda_{\frac{h}{2}}|},$$

respectively.

Numerical results given in Tables 5 and 6 show that the discretization accuracy of the finite element approximation to w is $O(h^2)$ with respect to both the discrete L^2 norm and the discrete H^1 seminorm for $\rho = 1.0$ and 0.5. The theoretically predicted error bounds are only $O(h^{\frac{5}{3}})$ in the L^2 norm and $O(h)$ in the H^1 norm. We therefore appear to have obtained optimal convergence in the L^2 and superconvergence in the discrete H^1 seminorm for this particular case. Our finite element theory in [5] requires that the mesh size h is sufficiently small since we used Gårding's inequality in its analysis. It is clear from Tables 5 and 6 that our finite element approximation is not subject to this constraint.

To test overall accuracy for the FMG, we studied the FMG based on $5\,V(1,0)$, $4\,V(2,0)$, and $4\,V(1,1)$ cycles. Tables 7 and 8 show that the total errors produced by the FMG are comparable to those discretization errors estimated in the respective Tables 5 and 6. This indicates that the underlying problem on level h can be solved to within the level of discretization error at a cost of about $\frac{4}{3} \cdot 5V(1,0) \approx 7\,V(1,0)$ or

Table 7
The discrete L^2-norm errors and the convergence rates by FMG.

Mesh size	FMG(5V(1,0))		FMG(4V(2,0))		FMG(4V(1,1))	
	Error	Rate	Error	Rate	Error	Rate
$h = \frac{1}{8}$	7.6617e-03	1.7477	7.6010e-03	1.7591	7.6114e-03	1.7572
$h = \frac{1}{16}$	2.0498e-03	1.9022	2.0296e-03	1.9050	2.0283e-03	1.9079
$h = \frac{1}{32}$	5.2220e-04	1.9728	5.1803e-04	1.9701	5.1511e-04	1.9773
$h = \frac{1}{64}$	1.3105e-04	1.9945	1.3012e-04	1.9932	1.2912e-04	1.9962
$h = \frac{1}{128}$	3.2667e-05	2.0042	3.2446e-05	2.0037	3.2179e-05	2.0045

Table 8
The discrete H^1-seminorm errors and the convergence rates by FMG.

Mesh size	FMG(5V(1,0))		FMG(4V(2,0))		FMG(4V(1,1))	
	Error	Rate	Error	Rate	Error	Rate
$h = \frac{1}{8}$	4.7954e-02	1.7790	4.7389e-02	1.7961	4.7465e-02	1.7938
$h = \frac{1}{16}$	1.2973e-02	1.8861	1.2672e-02	1.9029	1.2669e-02	1.9056
$h = \frac{1}{32}$	3.3488e-03	1.9538	3.2528e-03	1.9619	3.2237e-03	1.9745
$h = \frac{1}{64}$	8.6492e-04	1.9530	8.2053e-04	1.9871	8.0915e-04	1.9942
$h = \frac{1}{128}$	2.3739e-04	1.8653	2.0634e-04	1.9915	2.0290e-04	1.9956

Table 9
The absolute value errors and the convergence rates for λ.

Mesh size	$\rho = 1.00$		$\rho = 0.50$	
	$\lvert\lambda - \lambda_h\rvert$	Rate	$\lvert\lambda - \lambda_h\rvert$	Rate
$h = \frac{1}{4}$	7.0471e-02		6.8240e-02	
$h = \frac{1}{8}$	2.0457e-02	1.7844	2.0362e-02	1.7447
$h = \frac{1}{16}$	5.3595e-03	1.9324	5.3451e-03	1.9296
$h = \frac{1}{32}$	1.3576e-03	1.9810	1.3449e-03	1.9907
$h = \frac{1}{64}$	3.4091e-04	1.9936	3.2827e-04	2.0345
$h = \frac{1}{128}$	8.5640e-05	1.9930	7.3006e-05	2.1688

Table 10
The absolute value errors and the convergence rates by FMG.

Mesh size	FMG(5V(1,0))		FMG(4V(2,0))		FMG(4V(1,1))	
	Error	Rate	Error	Rate	Error	Rate
$h = \frac{1}{8}$	2.0710e-02	1.7667	2.0430e-02	1.7863	2.0454e-02	1.7846
$h = \frac{1}{16}$	5.4565e-03	1.9243	5.3802e-03	1.9250	5.3889e-03	1.9243
$h = \frac{1}{32}$	1.3846e-03	1.9785	1.3672e-03	1.9764	1.3650e-03	1.9811
$h = \frac{1}{64}$	3.4781e-04	1.9931	3.4360e-04	1.9924	3.4259e-04	1.9943
$h = \frac{1}{128}$	8.7360e-05	1.9933	8.6318e-05	1.9930	8.6045e-05	1.9933

$5\,V(2,0)$ or $5\,V(1,1)$ cycles on level h. Finally, results for the stress intensity factor are contained in Tables 9 and 10.

6. Conclusion and remarks. We study the MG V-cycle applied directly to the discrete problem in (4.2) and used in the SMA. Our numerical study shows that the direct MG is much more efficient than the iterative SMA. Our numerical study confirms theories on finite element accuracy established in [5] and MG convergence discussed in section 4.2. It especially shows that our finite element method seems to be optimally accurate in the L^2 norm. The theory for MG convergence is subject to the constraint that the coarsest mesh size is sufficiently small. But this condition is not essential numerically since we used $h_0 = 1/4$ in our experiments.

Acknowledgment. We would like to thank the referee for several helpful suggestions.

REFERENCES

[1] J. H. Bramble and J. E. Pasciak, *New estimates for multigrid algorithms including the V-cycle*, Math. Comp., 60 (1993), pp. 447–471.

[2] S. C. Brenner, *Multigrid methods for the computation of singular solutions and stress intensity factor* I: *Corner singularities*, Math. Comp., 68 (1999), pp. 559–583.

[3] S. C. Brenner and L. R. Scott, *The Mathematical Theory of Finite Element Methods*, Springer-Verlag, New York, 1994.

[4] W. L. Briggs, *A Multigrid Tutorial*, SIAM, Philadelphia, 1987.

[5] Z. Cai and S. Kim, *A finite element method using singular functions for the Poisson equations with corner singularities*, SIAM J. Numer. Anal., 39 (2001), pp. 286–299.

[6] Z. Cai and C. G. Lai, *Convergence estimates of multilevel additive and multiplicative algorithms for non-symmetric and indefinite problems*, Numer. Linear Algebra Appl., 3 (1996), pp. 205–220.

[7] P. G. Ciarlet, *The Finite Element Method for Elliptic Problems*, North-Holland, New York, 1978.

[8] M. Dauge, *Elliptic Boundary Value Problems on Corner Domains*, Lecture Notes in Math. 1341, Springer-Verlag, Berlin, Heidelberg, 1988.

[9] P. Grisvard, *Elliptic Problems in Nonsmooth Domains*, Pitman, Boston, 1985.

[10] J. Mandel, S. McCormick, and R. Bank, *Variational multigrid theory*, in Multigrid Methods, S. McCormick, ed., SIAM, Philadelphia, 1987, pp. 131–178.

SIAM J. SCI. COMPUT.
Vol. 23, No. 3, pp. 683–706

A VECTOR FINITE ELEMENT TIME-DOMAIN METHOD FOR SOLVING MAXWELL'S EQUATIONS ON UNSTRUCTURED HEXAHEDRAL GRIDS[*]

GARRY RODRIGUE[†] AND DANIEL WHITE[‡]

Abstract. In this paper the vector finite element time-domain (VFETD) method is derived, analyzed, and validated. The VFETD method uses edge vector finite elements as a basis for the electric field and face vector finite elements as a basis for the magnetic flux density. The Galerkin method is used to convert Maxwell's equations to a coupled system of ordinary differential equations. The leapfrog method is used to advance the fields in time. The method is shown to be stable and to conserve energy and charge for arbitrary hexahedral grids. A numerical dispersion analysis shows the method to be second order accurate on distorted hexahedral grids. Several computational experiments are performed to determine the accuracy and efficiency of the method.

Key words. Maxwell's equations, vector finite element, unstructured grids, edge elements

AMS subject classification. 74S05

PII. S1064827598343826

1. Introduction.

This paper is concerned with the numerical solution of the time dependent Maxwell equations in charge-free regions,

$$\begin{aligned} \frac{\partial \mathbf{B}}{\partial t} &= -\nabla \times \mathbf{E} - \sigma_M \mu^{-1}\mathbf{B}, \\ \varepsilon\frac{\partial \mathbf{E}}{\partial t} &= \nabla \times \mu^{-1}\mathbf{B} - \sigma_E \mathbf{E} - \mathbf{J}, \\ \nabla \bullet \varepsilon\mathbf{E} &= 0, \\ \nabla \bullet \mathbf{B} &= 0, \end{aligned} \tag{1}$$

with initial-boundary values

$$\hat{\mathbf{n}} \times \mathbf{E} = \mathbf{E}_{bc} \quad \text{on} \quad \Gamma = \text{boundary } (\Omega), \tag{2}$$

$$\mathbf{E}(t=0) = \mathbf{E}_{ic}, \mathbf{B}(t=0) = \mathbf{B}_{ic}. \tag{3}$$

Here, $\mathbf{E} = \mathbf{E}(x,t)$ is the electric field, $\mathbf{B} = \mathbf{B}(x,t)$ is the magnetic flux density, $\mathbf{J} = \mathbf{J}(x,t)$ is the electric current, and $\hat{\mathbf{n}}$ is the outward normal vector to Γ. The volume $\Omega \subset R^3$ is a domain, not necessarily bounded, whose boundary Γ is sufficiently regular (Lipschitz-continuous). Ω may be inhomogeneous, consisting of several dielectric, magnetic, and metallic regions of arbitrary geometry. The material properties are assumed to be linear and nondispersive. The volume may also contain several independent voltage and current sources. The electric and magnetic conductivities, σ_E

[*] Received by the editors August 24, 1998; accepted for publication (in revised form) March 5, 2001; published electronically August 15, 2001. This research was supported under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory contract W-7405-Eng-48.

http://www.siam.org/journals/sisc/23-3/34382.html

[†] Department of Applied Science, University of California at Davis, CA, and Lawrence Livermore National Laboratory, P.O. Box 808, L-561, Livermore, CA 94551 (ghrodrigue@ucdavis.edu).

[‡] Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, P.O. Box 808, L-561, Livermore, CA 94551 (dwhite@llnl.gov).

and σ_M, the dielectric permittivity ε, and the magnetic permeability μ are assumed to be time independent symmetric positive definite tensors. Example electromagnetic problems within this class include the design of waveguides and antennas, scattering of electromagnetic waves from automobiles and aircraft, and the penetration and absorption of electromagnetic waves by dielectric objects. Throughout this presentation, unless the physical meaning of quantities suggests otherwise, boldface letters will stand for vector fields whereas plain faced letters will stand for scalar fields.

The most popular numerical scheme for solving the time dependent Maxwell equations on an orthogonal Cartesian grid is the finite difference time-domain (FDTD) method [1, 2, 3, 4]. The method utilizes a centered difference approximation in space and a leapfrog approximation in time to yield a conditionally stable, consistent, and second order accurate scheme. However, when one attempts to use the FDTD method on complicated geometries by approximating curved boundaries with "stair steps," poor results are obtained [4, 5]. Nevertheless the FDTD method is extremely efficient and is the benchmark to which new methods are compared.

There have been several attempts to generalize the FDTD method to unstructured hexahedral grids, most notably the modified finite volume (MFV) and discrete surface integral (DSI) methods [6, 7, 8, 9]. In these methods Maxwell's equations are cast in integral form, then the subsequent volume and/or surface integrals are approximated by standard integration rules. A leap frog time integration is used so that these methods reduce to the FDTD method when orthogonal grids are used. However, the finite volume methods are not provably stable, and weak instabilities leading to nonphysical solution growth have been observed for nonorthogonal grids [10]. The instability is caused by the nonsymmetric discretization of the curl-curl operator. Dissipative time integration schemes may be employed to counteract this nonphysical solution growth, but this results in a violation of conservation of energy [11].

There is another class of finite volume methods where Maxwell's curl equations are cast in conservative form, resulting in a PDE that resembles the Euler equation of fluid dynamics [12, 13, 14]. The classical methods of computational fluid dynamics such as Lax–Wendroff or Godunov can then be used to solve the equations. These methods can be implemented on a structured, but nonorthogonal, hexahedral grid but do not reduce to the FDTD method when implemented on orthogonal Cartesian grids. However, these methods are stable and consistent, and very good accuracy can be achieved as the grid is refined. The methods rely upon dissipative time integration for stability and consequently do not conserve energy. In addition they neglect the divergence properties of the fields so that there is no guarantee that these methods will conserve charge.

Vector finite elements, also known as edge elements, Whitney 1-forms, or $H(curl)$ elements [15, 16, 17, 18, 19], have been used to formulate finite element solutions to the weak form of Maxwell's equations. These elements enforce tangential continuity of the fields but allow for jump discontinuity in the normal component of the fields. Use of these elements also eliminates spurious, divergent solutions of Maxwell's equations that were common with nodal element formulations [20, 21, 22, 23, 24, 25]. Vector finite element methods have been successfully used in the frequency domain to analyze resonant cavities, compute waveguide modes, and perform scattering calculations [26, 27, 28]. Vector finite elements have also been proposed to solve Maxwell's equations directly in the time domain [29, 30, 31, 32, 33]. Theoretical convergence results and error estimates for time dependent vector finite element solutions of Maxwell's equations were developed in [34, 35].

1.1. Vector finite element time-domain (VFETD) method. In this paper, a Galerkin procedure is used to convert the weak form of Maxwell's equations to a semidiscrete coupled system of ordinary differential equations using vector finite elements. A leapfrog time integration scheme similar to that found in the FDTD method is then used to discretize time and update the field variables.

The VFETD method uses vector "edge" finite elements as a basis for the electric field and vector "face" finite elements as a basis for the magnetic flux density. These elements are complementary in the sense that the edge elements have tangential continuity across interfaces whereas the face elements have normal continuity across interfaces. Consequently, the Galerkin approximations preserve field continuities/discontinuities across material interfaces. The properties of these vector finite elements is discussed in detail in [15].

The VFETD method is shown to be conditionally stable. Moreover, if a stable time step is used, the method conserves energy and charge, independent of how distorted the grid is. A numerical dispersion analysis of the method is performed on several different distorted hexahedral grids, with the result that the method is second order accurate. The analysis also shows how the anisotropic part of the numerical dispersion relation depends upon the grid distortion [40, 41, 42, 43].

The VFETD method requires a sparse linear system to be solved at every time step. The computational effort required to solve the system depends upon how distorted the grid is. For Cartesian grids mass lumping can be used, in which case the VFETD method reduces to the classic FDTD method. For non-Cartesian grids conjugate gradient iterative methods are used to solve the system where it is shown, via computational experiments on unstructured hexahedral grids, that the number of iterations required to achieve a given accuracy is a constant independent of the grid cell size (grid refinement) used to discretize the problem. Hence the method is scalable. The accuracy of the method, as well as the required CPU time, are tabulated for resonant cavity, waveguide, and antenna problems.

2. Weak formulation of Maxwell's equations. In this section we convert Maxwell's equations into variational equations posed over suitable function spaces. We first consider the space

$$H(curl;\ \Omega) = \{\mathbf{u} \in \mathbf{L}_2(\Omega);\ \nabla \times \mathbf{u} \in \mathbf{L}_2(\Omega)\}.$$

A function $\mathbf{u}$ in the vector space $\mathbf{H}^1(K_1) \cup \mathbf{H}^1(K_2)$ is in $H(curl;\, \Omega = K_1 \cup K_2)$ if and only if the trace $\mathbf{u} \times \mathbf{n}$ is the same on each side of the face Γ [15, 16]. Consequently, $H(curl;\ \Omega)$ is an appropriate space for the electric field $\mathbf{E}$. Similarly, we define the function space

$$H(\mathrm{div}; \Omega) = \{\mathbf{u} \in \mathbf{L}_2(\Omega); \nabla \bullet \mathbf{u} \in L_2(\Omega)\}.$$

Then, a function $\mathbf{u}$ in the vector space $\mathbf{H}^1(K_1) \cup \mathbf{H}^1(K_2)$ is in $H(\mathrm{div};\, \Omega = K_1 \cup K_2)$ if and only if $\mathbf{u} \bullet \mathbf{n}$ is the same on each side of the face Γ [15, 16]. Hence, $H(\mathrm{div};\ \Omega)$ is an appropriate space to which the magnetic flux density $\mathbf{B}$ should belong. Both spaces, equipped with the canonical inner products, are Hilbert spaces with norms

$$\|\mathbf{u}\|_{H(curl\,:\,\Omega)} = (\|\mathbf{u}\|_2^2 + \|\nabla \times \mathbf{u}\|_2^2)^{1/2},$$
$$\|\mathbf{u}\|_{H(\mathrm{div}\,:\,\Omega)} = (\|\mathbf{u}\|_2^2 + \|\nabla \bullet \mathbf{u}\|_2^2)^{1/2}.$$

We write $\|\mathbf{u}\|_2^2 = \int_\Omega \mathbf{u}^t\mathbf{u}\, d\Omega \equiv (\mathbf{u}, \mathbf{u}) < \infty$ for the $\mathbf{L}_2(\Omega)$-norm and $(\mathbf{u}, \mathbf{v}) = \int_\Omega \mathbf{u}^t\mathbf{v}\, d\Omega$ for the $\mathbf{L}_2(\Omega)$ inner product. The subspace of $H(curl; \Omega)$ containing the vector fields

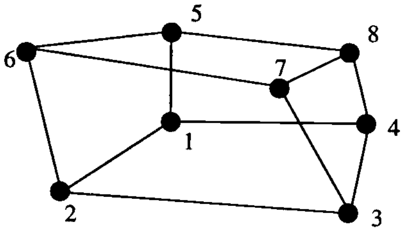


FIG. 1. *Illustration of an arbitrary hexahedron.*

$\mathbf{u}$ with vanishing tangential trace $\mathbf{n} \times \mathbf{u}$ on Γ is denoted by $H_0(curl;\Omega)$. As usual, $\mathbf{n}$ designates the exterior unit normal vector on Γ. Similarly, $H_0(\text{div};\Omega)$ denotes the subspace of $H(\text{div};\Omega)$ containing vectors $\mathbf{u}$ such that $\mathbf{u} \bullet \mathbf{n} = 0$ on Γ.

Given the above definitions, a natural way of defining the weak form of Maxwell's equations (1)–(3) is to determine functions $\mathbf{B} \in H(\text{div};\Omega)$, $\mathbf{E} \in H(curl;\Omega)$ such that

$$\frac{\partial}{\partial t}(\mu^{-1}\mathbf{B},\mathbf{B}^*) = -(\mu^{-1}\nabla \times \mathbf{E},\mathbf{B}^*) = (\mu^{-1}\sigma_M\mu^{-1}\mathbf{B},\mathbf{B}^*), \tag{4}$$

$$\frac{\partial}{\partial t}(\varepsilon\mathbf{E},\mathbf{E}^*) = (\nabla \times \mathbf{E}^*,\mu^{-1}\mathbf{B}) - (\sigma_E\mathbf{E},\mathbf{E}^*) - (J,\mathbf{E}^*) \tag{5}$$

for all $\mathbf{B}^* \in H_0(\text{div};\Omega)$, $\mathbf{E}^* \in H_0(curl;\Omega)$. The multiplication of the first equation by μ^{-1} is done to aid in the analysis of the Galerkin method. Clearly, if $\mathbf{B}$, $\mathbf{E}$ are classical solutions of Maxwell's equations (1)–(3), then they are solutions of the weak equations (4)–(5).

3. Finite element basis functions.

A general definition of finite elements on arbitrary polyhedra is given by the following.

DEFINITION. *A finite element* (K, P, A), *consists of*

1. K, *a polyhedral domain*;
2. P, *a vector space of polynomials defined on* K *having a basis* $\{\psi_1, \psi_2, \dots, \psi_N\}$ *(called shape functions)*;
3. A, *a set of linear functionals defined on* P *having a basis* $\alpha_1, \alpha_2, \dots, \alpha_N$ *(called the degrees of freedom).*

In this section we define K, P, A for the linear edge and face elements. For the elements developed in this paper, we shall approximate the domain Ω with a hexahedral mesh κ consisting of $K_i, i = 1, \dots, N$, hexahedra. Each of the hexahedra can be mapped using the standard trilinear mapping $(x, y, z) = B(\zeta, \eta, \nu)$ to a reference element, $K_0 = \{0 \leq \zeta, \eta, \nu \leq 1\}$. We require that the mapping B be one-to-one and invertible, implying a nonsingular Jacobian matrix J. Consequently, K will be a hexahedron consisting of 8 nodes labeled as in Figure 1. The 12 edges, a_i, and 6 faces, f_i, are numbered according to Table 1. The Dirichlet part of the boundary Γ_D is assumed to be the union of complete faces of elements.

3.1. Finite elements in *H(curl)*: Edge elements.

We consider finite elements (K, P, A), called "edge elements," where K is an arbitrary hexahedron and the degrees of freedom of A are defined by

$$\alpha_i(v) = \int_{a_i} (v \bullet \mathbf{t}_i)\, ds, v \in P \tag{6}$$

Table 1
Node, edge, and face numbering scheme for hexahedrons.

i	edge, a_i	face, f_i
1	1-2	1-4-8-5
2	4-3	2-3-7-6
3	5-6	1-5-6-2
4	8-7	4-8-7-3
5	1-4	1-2-3-4
6	5-8	5-6-7-8
7	2-3	
8	6-7	
9	1-5	
10	2-6	
11	4-8	
12	3-7	

with $\mathbf{t}_i$ being the unit tangent vector along edge a_i of K. The space P is defined by considering the reference element (K_0, P_0, A_0). Here

$$P_0 = \{\mathbf{u} = [u_1, u_2, u_3]^t : u_1 \in Q_{0,1,1}; u_2 \in Q_{1,0,1}; u_3 \in Q_{1,1,0}\},$$

where $Q_{l,m,n}$ denotes the vector space of polynomials in three variables (x, y, z), the maximum degree of which are, respectively, l in x, m in y, n in z. Note that *dimension* $[P_0] = 12$. The basis for P_0, as constructed by (6), is

$$\begin{aligned} &\mathbf{W}_1^{(0)} = (1 - y - z + yz, 0, 0), && \mathbf{W}_2^{(0)} = (y - yz, 0, 0), \\ &\mathbf{W}_3^{(0)} = (z - yz, 0, 0), && \mathbf{W}_4^{(0)} = (yz, 0, 0), \\ &\mathbf{W}_5^{(0)} = (0, 1 - x - z - xz, 0), && \mathbf{W}_6^{(0)} = (0, x - xz, 0), \\ &\mathbf{W}_7^{(0)} = (0, z - xz, 0), && \mathbf{W}_8^{(0)} = (0, xz, 0), \\ &\mathbf{W}_9^{(0)} = (0, 0, 1 - x - y - xy), && \mathbf{W}_{10}^{(0)} = (0, 0, x - xy), \\ &\mathbf{W}_{11}^{(0)} = (0, 0, y - xy), && \mathbf{W}_{12}^{(0)} = (0, 0, xy). \end{aligned} \tag{7}$$

The above polynomial basis defined on the reference element must be transformed to the arbitrary hexahedron K such that the degrees of freedom are preserved. It is well known that the covariant transformation preserves line integrals under a change of coordinates; hence we define P in (K, P, A) by $\mathbf{W}_i = J^{-t}\mathbf{W}_i^{(0)}$.

3.2. Finite elements in *H*(div): Face elements. We now consider a finite element (K, P, A), called "face elements," where K is an arbitrary hexahedron and the degrees of freedom of a face element are

$$\alpha_i(v) = \int_{f_i} (v \bullet \mathbf{n}_i)\, ds, \tag{8}$$

where $\mathbf{n}_i$ is the unit normal vector to the face f_i of K. The space P is defined by again considering the reference element (K_0, P_0, A_0). Here, $P_0 = \{\mathbf{u} = [u_1, u_2, u_3]^t :$

$u_1 \in Q_{1,0,0}$; $u_2 \in Q_{0,1,0}$; $u_3 \in Q_{0,0,1}\}$. The basis functions of P_0 are constructed from (8) to get

$$\begin{aligned} &\mathbf{F}_1^{(0)} = (-1+x,0,0), \quad &\mathbf{F}_2^{(0)} = (x,0,0), \quad &\mathbf{F}_3^{(0)} = (0,-1+y,0), \\ &\mathbf{F}_4^{(0)} = (0,y,0), \quad &\mathbf{F}_5^{(0)} = (0,0,-1+z), \quad &\mathbf{F}_6^{(0)} = (0,0,z). \end{aligned} \tag{9}$$

The above polynomial basis defined on the reference element must be transformed to the arbitrary hexahedron K such that the degrees of freedom are preserved. It is well known that a contravariant transformation preserves surface integrals; hence we define P in (K,P,A) by $\mathbf{F}_i = J\mathbf{F}_i^{(0)}$.

4. Galerkin approximation of Maxwell's equations. The Galerkin method constructs approximations

$$\begin{aligned} \tilde{\mathbf{E}} &= \sum_{j=1}^{N_E} e_j \mathbf{W}_j \in W = \text{span }[\mathbf{W}_1, \mathbf{W}_2, \ldots, \mathbf{W}_{N_B}] \subset H_0(curl;\Omega), \\ \tilde{\mathbf{B}} &= \sum_{j=1}^{N_B} b_j \mathbf{F}_j \in F = \text{span }[\mathbf{F}_1, \mathbf{F}_2, \ldots, \mathbf{F}_{N_E}] \subset H_0(\text{div};\Omega), \end{aligned} \tag{10}$$

such that

$$\begin{aligned} \frac{\partial}{\partial t}(\mu^{-1}\tilde{\mathbf{B}}, \mathbf{F}_i) &= -(\mu^{-1}\nabla \times \tilde{\mathbf{E}}, \mathbf{F}_i) + -(\mu^{-1}\sigma_M \mu^{-1}\tilde{\mathbf{E}}, \mathbf{F}_i), \qquad i = 1,2,\ldots,N_F, \\ \frac{\partial}{\partial t}(\varepsilon\tilde{\mathbf{E}}, \mathbf{W}_i) &= (\nabla \times \mathbf{W}_i, \mu^{-1}\tilde{\mathbf{E}}) - (\sigma_E \tilde{\mathbf{B}}, \mathbf{W}_i) - (\tilde{\mathbf{j}}, \mathbf{W}_i), \qquad i = 1,2,\ldots,N_E, \end{aligned} \tag{11}$$

where N_E and N_F are the number of internal edges and faces, respectively. This leads to systems of ordinary differential equations

$$\begin{aligned} G\frac{\partial \mathbf{b}}{\partial t} &= -K\mathbf{e} - P\mathbf{b}, \\ C\frac{\partial \mathbf{e}}{\partial t} &= K^T\mathbf{b} - S\mathbf{e} - Q\mathbf{j}, \end{aligned} \tag{12}$$

where $\mathbf{b} = [b_1, b_2, \ldots, b_{N_F}]^t$, $\mathbf{e} = [e_1, e_2, \ldots, e_{N_E}]^t$, and the matrices are given by

$$\begin{aligned} G_{ij} &= (\mu^{-1}\mathbf{F}_i, \mathbf{F}_j), \qquad K_{ij} = (\mu^{-1}\nabla \times \mathbf{W}_i, \mathbf{F}_j), \\ P_{ij} &= (\mu^{-1}\sigma_M \mu^{-1}\mathbf{F}_i, \mathbf{F}_j), \qquad C_{ij} = (\varepsilon\mathbf{W}_i, \mathbf{W}_j), \\ K_{ij}^t &= (\mu^{-1}\nabla \times \mathbf{W}_j, \mathbf{F}_i), \qquad S_{ij} = (\sigma_E \mathbf{W}_i, \mathbf{W}_j), \qquad Q_{ij} = (\mathbf{F}_i, \mathbf{W}_j). \end{aligned} \tag{13}$$

5. Leapfrog time differencing. The ordinary differential equations (12) are differenced so that the electric fields are calculated at whole time steps and the magnetic fields are calculated at the half time steps. Specifically,

$$(G + \Delta tP/2)\mathbf{b}^{n+1/2} = -\Delta tK\mathbf{e}^n + (G - \Delta tP/2)\mathbf{b}^{n-1/2}, \tag{14}$$

$$(C + \Delta tS/2)\mathbf{e}^{n+1} = \Delta tK^T\mathbf{b}^{n+1/2} + (C - \Delta tS/2)\mathbf{e}^n - \Delta tQ\mathbf{j}^{n+1/2}. \tag{15}$$

5.1. Stability and conservation. Properties of (14) will be derived by assuming no electric or magnetic conductivity, i.e., $\sigma_E = \sigma_M = \mathbf{J} = 0$. In this case the discrete equations (14) become

$$\begin{bmatrix} \mathbf{e}^{n+1} \\ \mathbf{b}^{n+1/2} \end{bmatrix} = \begin{bmatrix} (I - \Delta t^2 C^{-1}K^t G^{-1}K) & \Delta t C^{-1}K^t \\ -\Delta t G^{-1}K & I \end{bmatrix} \begin{bmatrix} \mathbf{e}^{n} \\ \mathbf{b}^{n-1/2} \end{bmatrix}. \tag{16}$$

The following theorem establishes eigenproperties of the method.

Theorem 1. *Let $T(\Delta t)$ be the amplification matrix in* (16). *Then*

(a) *the eigenvalues of $T(\Delta t)$ either have unit magnitude or are negative*

(b) *the eigenvalues of $T(\Delta t)$ have unit magnitude if and only if*

$$\Delta t \leq \frac{2}{\sqrt{\max(\psi)}}, \tag{17}$$

where ψ is an eigenvalue of $C^{-1}K^tG^{-1}K$.

Proof. (a) Suppose $T(\Delta t)$ has a complex eigenvalue $\lambda = a + ib$. Then there is a complex eigenvector $\left[\begin{smallmatrix} x \\ y \end{smallmatrix} \right]$ that solves the eigenvalue problem

$$\begin{bmatrix} (I - \Delta t^2 C^{-1}K^t G^{-1}K) & \Delta t C^{-1}K^t \\ -\Delta t G^{-1}K & I \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \lambda \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}.$$

Since the matrices C and G are symmetric and positive definite, they admit Cholesky decompositions $C = \tilde{C}^t\tilde{C}$ and $G = \tilde{G}^t\tilde{G}$, respectively. If we let $\tilde{\mathbf{x}} = \tilde{C}\mathbf{x}$ and $\tilde{\mathbf{y}} = \tilde{G}\mathbf{y}$, then the above eigenproblem is equivalent to

$$\begin{bmatrix} (I - \Delta t^2 QQ^t) & \Delta t Q \\ -\Delta t Q^t & I \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{x}} \\ \tilde{\mathbf{y}} \end{bmatrix} = \lambda \begin{bmatrix} \tilde{\mathbf{x}} \\ \tilde{\mathbf{y}} \end{bmatrix}, \tag{18}$$

where the matrix Q is given by $Q = \tilde{C}^{-t}K^t\tilde{G}^{-1}$. Note that $QQ^t = C^{-1}K^tG^{-1}K$. Since

$$\begin{bmatrix} (I - \Delta t^2 QQ^t) & \Delta t Q \\ -\Delta t Q^t & I \end{bmatrix} = \begin{bmatrix} I & \Delta t Q \\ O & I \end{bmatrix} \begin{bmatrix} I & O \\ -\Delta t Q^t & i \end{bmatrix},$$

its determinant is 1 and consequently $\lambda \neq 0$. We can now write (18) as a general eigenproblem:

$$\begin{bmatrix} I & O \\ -\Delta t Q^t & I \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{x}} \\ \tilde{\mathbf{y}} \end{bmatrix} = \lambda \begin{bmatrix} I & \Delta t Q \\ O & I \end{bmatrix}^{-1} \begin{bmatrix} \tilde{\mathbf{x}} \\ \tilde{\mathbf{y}} \end{bmatrix} = \lambda \begin{bmatrix} I & -\Delta t Q \\ O & I \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{x}} \\ \tilde{\mathbf{y}} \end{bmatrix}.$$

We get that

$$(1 - \lambda)\tilde{\mathbf{y}} = \Delta t Q^t \tilde{\mathbf{x}}$$

and

$$-\lambda \Delta t^2 QQ^t \tilde{\mathbf{x}} = (1 - \lambda)^2 \tilde{\mathbf{x}}.$$

Hence, $\tilde{\mathbf{x}}$ is an eigenvector of QQ^t with eigenvalue $-(\lambda - 1)^2/\lambda\Delta t^2$. Since QQ^t is symmetric, it has only real eigenvalues. Thus,

$$\operatorname{Im}\left(\frac{-(\lambda - 1)^2}{\lambda} \right) = \operatorname{Im}\left(2 - \frac{(\lambda^2 + 1)}{\lambda} \right) = \operatorname{Im}(\lambda + \lambda^{-1}) = b(1 - (a^2 + b^2)^{-1}) = 0.$$

It follows that either $b = 0$, in which case λ is real, or $|\lambda| = 1$. If λ is real, then using the fact that QQ^t is positive semidefinite we see its eigenvalues are nonnegative so that $\lambda = 1$ or $\lambda < 0$.

(b) Now suppose that condition (17) holds. In view of (a), we need only consider the case $\lambda < 0$. Then

$$\Delta t^2 \leq \frac{4}{\psi} \leq \frac{-4\lambda\Delta t^2}{(\lambda-1)^2} \Rightarrow (\lambda+1)^2 \leq 0 \Rightarrow \lambda = -1.$$

Now suppose all eigenvalues of $T(\Delta t)$ have unit magnitude. Let $\psi = (2+\alpha)/\Delta t^2$ be the eigenvalue of QQ^t that is largest in magnitude. Let λ be such that $\alpha = -(\lambda^2+1)/\lambda$. Specifically, λ is given by

$$\lambda = \frac{-\alpha \pm \sqrt{\alpha^2-4}}{2}.$$

Moreover, using the same arguments as in (a) to show the connection between the eigenvalues of QQ^t and $T(\Delta t)$, we see that λ is an eigenvalue of $T(\Delta t)$ and by hypothesis has unit magnitude. Then

$$\frac{\Delta t^2}{4}|\psi| = \frac{|\lambda-1|^2}{4|\lambda|} \leq 1$$

and the theorem is proved. □

Theorem 2 (conservation of magnetic charge). *If* $\tilde{\mathbf{B}}^{n+1/2} = \sum_{i=1}^{N_f} b_i^{n+1/2}\mathbf{F}_i$ *is the Galerkin approximation computed from* (16), *then*

$$\int_\Omega \nabla \bullet \tilde{\mathbf{B}}^{n+1/2} d\Omega = \int_\Omega \nabla \bullet \tilde{\mathbf{B}}^{n-1/2} d\Omega, \qquad n = 1, 2, \ldots.$$

Proof. Let $\delta\mathbf{b}^n = (\mathbf{b}^{n+1/2} - \mathbf{b}^{n-1/2})/\Delta t$. Then by (16)

$$G\delta\mathbf{b}^n = -K\mathbf{e}^n.$$

Note that the edge basis functions and the face basis functions are related by the so-called inclusion (or compatibility) condition $\nabla \times \mathbf{W}_i \in F$. In particular, the edge and face basis functions are normalized such that $\nabla \times W_i = \sum_{j=1}^{2} a_{ij}\mathbf{F}_{ij}$, where $\mathbf{F}_{ij}$ are the two face functions associated with the edge function W_i and $a_{ij} = \pm 1$ with the sign depending upon the right-hand rule. Thus,

$$\sum_{i=1}^{N_e} e_i \nabla \times \mathbf{W}_i = \sum_{i=1}^{N_e} e_i \sum_{j=1}^{2} a_{ij}\mathbf{F}_{ij}$$

and by (16) we have

$$\sum_{i=1}^{N_f} \delta b_i^n(\mathbf{F}_i, \mathbf{F}_k) = \sum_{i=1}^{N_e} e_i^n(\nabla \times \mathbf{W}_i, \mathbf{F}_k) = \sum_{i=1}^{N_e} e_i^n \sum_{j=1}^{2} a_{ij}(\mathbf{F}_{ij}, \mathbf{F}_k)$$

for $k = 1, \ldots, N_f$. The two summations in the right-hand term can be combined so that the summation is performed over all N_f face, with four electric field degrees of freedom e_j contributing to each b_i:

$$\sum_{i=1}^{N_f} \delta b_i^n(\mathbf{F}_i, \mathbf{F}_k) = \sum_{i=1}^{N_f} \left(\sum_{j=1}^{4} e_j^n a_{ij} \right) (\mathbf{F}_i, \mathbf{F}_k).$$

It follows that

$$\delta b_i^n = \sum_{j=1}^{4} e_j^n a_{ij}$$

Using Gauss' law, the divergence of the magnetic flux density in a given cell Ω_e is

$$\int_\Omega (\nabla \bullet \delta\tilde{\mathbf{B}}^n) = \oint \delta\tilde{\mathbf{B}}^n \bullet n d\Gamma = \sum_{i=1}^{6} \delta b_i^n = \sum_{i=1}^{6}\sum_{j=1}^{4} e_j^n a_{ij} = 0,$$

and the theorem follows by summing over the cells. □

THEOREM 3 (conservation of electric charge). *Let S_0 be the space of trilinear continuous Lagrangian finite elements vanishing on γ_D. If $\tilde{\mathbf{E}}^n = \sum_{i=1}^{N_e} e_i^n \mathbf{W}_i$ is the Galerkin approximation computed from* (16), *then*

$$\int_\Omega (\nabla \bullet \varepsilon\tilde{\mathbf{E}}^n)\psi \, d\Omega = \int_\Omega (\nabla \bullet \varepsilon\tilde{\mathbf{E}}^{n-1})\psi \, d\Omega, \qquad n = 1, 2, \ldots, \tag{19}$$

for all continuous piecewise linear functions $\phi \in S_0$.

Proof. First note that integration by parts yields

$$\int_\Omega (\nabla \bullet \varepsilon\overline{\mathbf{E}})\phi = -\int_\Omega \varepsilon\tilde{\mathbf{E}} \bullet \nabla\phi d\Omega + \oint_\Gamma \phi\varepsilon\tilde{\mathbf{E}} \bullet \mathbf{n} \, d\Gamma = -\int_\Omega \varepsilon\tilde{\mathbf{E}} \bullet \nabla\phi \, d\Omega.$$

Let $\delta\overline{\mathbf{E}}^n = \Delta t^{-1}(\overline{\mathbf{E}}^n - \overline{\mathbf{E}}^{n-1})$. Then by (16),

$$(\varepsilon\delta\overline{\mathbf{E}}^n, \mathbf{W}_j) = (\nabla \times \mathbf{W}_j, \mu^{-1}\tilde{\mathbf{B}}^{n-1/2}), \qquad j = 1, 2, \ldots, N_E.$$

Now, $\nabla\phi \in W$; cf. [15, 25]. Therefore

$$\int_\Omega (\nabla \bullet \varepsilon\delta\overline{\mathbf{E}}^n)\phi = \int_\Omega \varepsilon\delta\tilde{\mathbf{E}}^n \bullet \nabla\phi \, d\Omega = \int_\Omega (\nabla \times \nabla\phi) \bullet \mu^{-1}\tilde{\mathbf{B}}^{n-1/2} d\Omega = 0,$$

and the theorem is proved. □

5.2. Numerical dispersion. Equation (1) in an infinite, source free, zero conductivity region becomes the vector wave equation

$$\varepsilon\frac{\partial^2 \mathbf{E}}{\partial t^2} = -\nabla \times \mu^{-1}\nabla \times \mathbf{E}. \tag{20}$$

If μ and ε are constant scalars, then

$$\mathbf{E} = \mathbf{E}_0 e^{I(k \bullet x - \omega t)} \tag{21}$$

is a solution to (20) only if the dispersion relation $\omega^2 = c^2k^2$ holds where $k = \|\mathbf{k}\|_2$ and $c = 1/(\sqrt{\mu\varepsilon})$ is the speed of light. Now, consider the solution

$$\mathbf{e}(t) = \sum_i e_i(t)\mathbf{W}_i(\mathbf{x}), \qquad e_i(t) = \int_{a_i} \mathbf{E}(x, t) \bullet \mathbf{t} \, dl$$

of the Galerkin form of (20)

$$C\frac{\partial^2 \mathbf{e}}{\partial t^2} = -A\mathbf{e}, \qquad e_i(0) - \int_{a_i} \mathbf{E}(\mathbf{x}, 0) \bullet \mathbf{t} \, dl \tag{22}$$

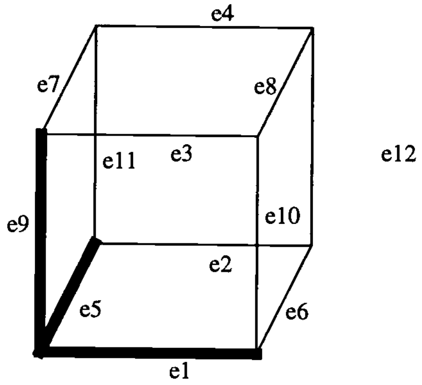


FIG. 2. *Edge numbering for numerical dispersion analysis.*

with $\mathbf{t}$ the unit tangent vector to edge a_i. The matrices C and A are defined by (13) and $A_{ij} = (\mu^{-1}\nabla \times \mathbf{W}_i, \nabla \times \mathbf{W}_j)$, respectively. We now assume the grid to be composed of identical hexahedral cells, which may be distorted. For this analysis the distortion is such that edges $\tilde{e}_1 - \tilde{e}_4$, $\tilde{e}_5 - \tilde{e}_8$, $\tilde{e}_9 - \tilde{e}_{12}$ are parallel; see Figure 2.

If we let

$$X(t) = \int_{\tilde{e}_1} \mathbf{E}(x,t) \bullet \mathbf{t}\, dl, \qquad Y(t) = \int_{\tilde{e}_5} \mathbf{E}(x,t) \bullet \mathbf{t}\, dl, \qquad Z(t) = \int_{\tilde{e}_9} \mathbf{E}(x,t) \bullet \mathbf{t}\, dl,$$

then

$$\begin{array}{lll}
\tilde{e}_1(t) = X(t), & \tilde{e}_2(t) = Xe^{I(\mathbf{k}\bullet\vec{\Delta}_{1,2}-\omega\Delta t)}, & \tilde{e}_3(t) = Xe^{I(\mathbf{k}\bullet\vec{\Delta}_{1,3}-\omega\Delta t)}, \\
\tilde{e}_4(t) = Xe^{I(\mathbf{k}\bullet\vec{\Delta}_{1,4}-\omega\Delta t)}, & \tilde{e}_5(t) = Y(t), & \tilde{e}_6(t) = Ye^{I(\vec{\mathbf{k}}\bullet\vec{\Delta}_{5,6}-\omega\Delta t)}, \\
\tilde{e}_7(t) = Xe^{I(\mathbf{k}\bullet\vec{\Delta}_{5,7}-\omega\Delta t)}, & \tilde{e}_8(t) = Xe^{I(\mathbf{k}\bullet\vec{\Delta}_{5,8}-\omega\Delta t)}, & \tilde{e}_9(t) = Z(t), \\
\tilde{e}_{10}(t) = Ze^{I(\mathbf{k}\bullet\vec{\Delta}_{9,10}-\omega\Delta t)} & \tilde{e}_{11}(t) = Ze^{I(\mathbf{k}\bullet\vec{\Delta}_{9,11}-\omega\Delta t)} & \tilde{e}_{12}(t) = Ze^{I(\mathbf{k}\bullet\vec{\Delta}_{9,12}-\omega\Delta t)},
\end{array}$$
(23)

where $\vec{\Delta}_{i,j}$ is the vector from the midpoint of edge $\tilde{e}_i$ to the midpoint of edge $\tilde{e}_j$.

Clearly,

$$(24) \qquad \left(\frac{\partial^2 \tilde{e}_i}{\partial t^2}\right)^n \approx \frac{\tilde{e}_i^{n+1} - 2\tilde{e}_i^n + \tilde{e}_i^{n-1}}{\Delta t^2} = \frac{\psi \tilde{e}_i}{\Delta t^2}, \qquad \psi = 2\cos(\omega\Delta t - 1).$$

If we assume $\|\vec{\Delta}_{i,j}\|_2 = \Delta x$ for all (i,j), then (22)–(24) yields a homogeneous system of equations:

$$(25) \qquad (\psi F + \eta G)\begin{pmatrix} \mathrm{X} \\ \mathrm{Y} \\ \mathrm{Z} \end{pmatrix} = 0, \qquad \eta = c^2 \frac{\Delta t^2}{\Delta x^2}.$$

The numerical dispersion relation is given by

$$(26) \qquad \det(\psi F + \eta G) = 0,$$

where the 3×3 matrices F, G are complicated nonlinear relationship between the wave vector $\mathbf{k}$ and the radian frequency ω. There are three roots; one is zero, which does not represent anything physical, and the other two correspond to the two distinct polarizations.

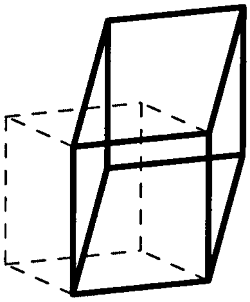


FIG. 3. *Illustration of a cube distorted in the x and z directions by an amount $\theta = 45°$.*

Example: Numerical dispersion for three-dimensional shear distortion. Let

$$\mathbf{k} = k[\cos(\phi)\sin(\Phi), \sin(\phi)\sin(\Phi), \cos(\Phi)]^t \tag{27}$$

be a wave vector as a function of the spherical angles ϕ and Φ. The exact phase velocity for (20)–(21) is given by $\omega/k = c$ and the numerical phase velocity $v = \tilde{\omega}/k$ is computed by specifying a value of $\mathbf{k}$ as a function of ϕ Φ and solving numerically for the value of $\tilde{\omega}$ that satisfies (26).

In this example, a unit cube is sheared by an amount 0 in both the x direction and the z direction; see Figure 3.

In the computational experiments we take $c = 1$ and $\Delta t = 1/3$. Figure 4 shows surfaces of the phase velocity error for shear angles of $\theta = 0°, 15°, 30°, 45°$. Each figure shows the velocity error for $k = 1\pi/5$, where the velocity error is defined as $v - c$. The shape of the velocity error surface remains the same as k is decreased, thus it is not necessary to display different surfaces. Note that the scale is different for each plot.

The maximum velocity, minimum velocity, and anisotropy ratio are tabulated in Table 2 as a function of k for each of the four grid distortions. The results demonstrate that, as the grid becomes more distorted, the numerical dispersion relation becomes more anisotropic.

It is possible to determine the rate of convergence of the numerical dispersion relation for distorted hexahedral grids by applying a least-squares fit to the above data. The logarithm of the error versus the logarithm of k is shown in Figure 5 for each of the four grids, along with a least-squares linear fit. The least-squares fit is applied to the maximum velocity error. For each grid the slope of the linear fit is approximately 2 (from 2.02 to 2.09), indicating second order convergence of the numerical dispersion relation.

6. Linear system solution methods. The VFETD method requires the solution of a large, sparse, symmetric, positive-definite mass matrix equation $Cx = y$ at every time step. In this paper, the incomplete Cholesky conjugate gradient (ICCG) method will be used to solve the mass matrix. Basically, the ICCG method is a preconditioned conjugate gradient method where the preconditioner is constructed by applying the Cholesky factorization algorithm to the mass matrix C and ignoring the nonzero fill-in [39]. This generates an incomplete Cholesky factorization $\tilde{L}\tilde{L}^t$, where $\tilde{L}$ has the same sparsity pattern as the matrix C. For the special case of a Cartesian grid, the following result holds.

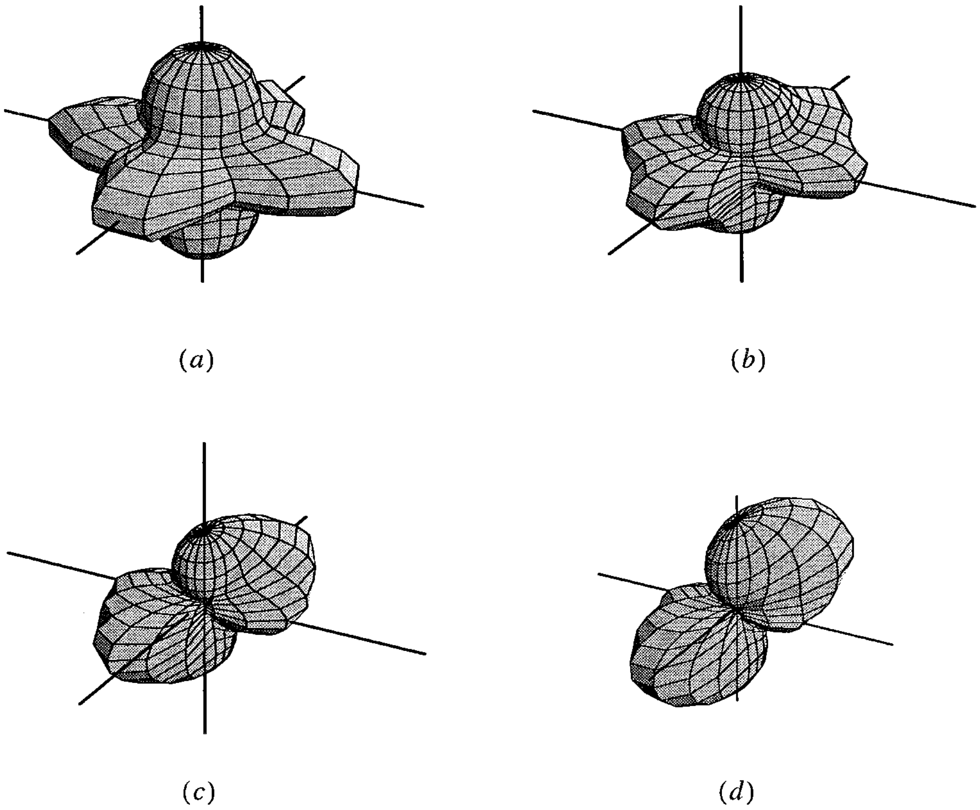


FIG. 4. *Phase velocity error for* $\theta =$ (a) 0°, (b) 15°, (c) 30°, (d) 45°. *The surface corresponds to* $k = 2\pi/5$. *The length of the axes are* (a) 0.15, (b) 0.25, (c) 0.35, (d) 0.35.

TABLE 2
Phase velocity and anisotropy ratio versus k.

	$\theta = 0°$			$\theta = 15°$		
k	$\max v$	$\min v$	ratio	$\max v$	$\min v$	ratio
$2\pi/5$	1.07538	1.03002	1.04404	1.08797	1.01709	1.06969
$2\pi/10$	1.01845	1.00736	1.01101	1.02113	1.00423	1.01682
$2\pi/15$	1.00816	1.00326	1.00488	1.00931	1.00188	1.00742
$2\pi/20$	1.00458	1.00183	1.00274	1.00522	1.00106	1.00416
	$\theta = 30°$			$\theta = 45°$		
$2\pi/5$	1.14536	1.00913	1.135	1.35058	1.00333	1.34609
$2\pi/10$	1.03401	1.00227	1.0316	1.08656	1.00083	1.08566
$2\pi/15$	1.01493	1.00101	1.0139	1.03845	1.00037	1.03807
$2\pi/20$	1.00836	1.0057	1.00779	1.02163	1.00021	1.02142

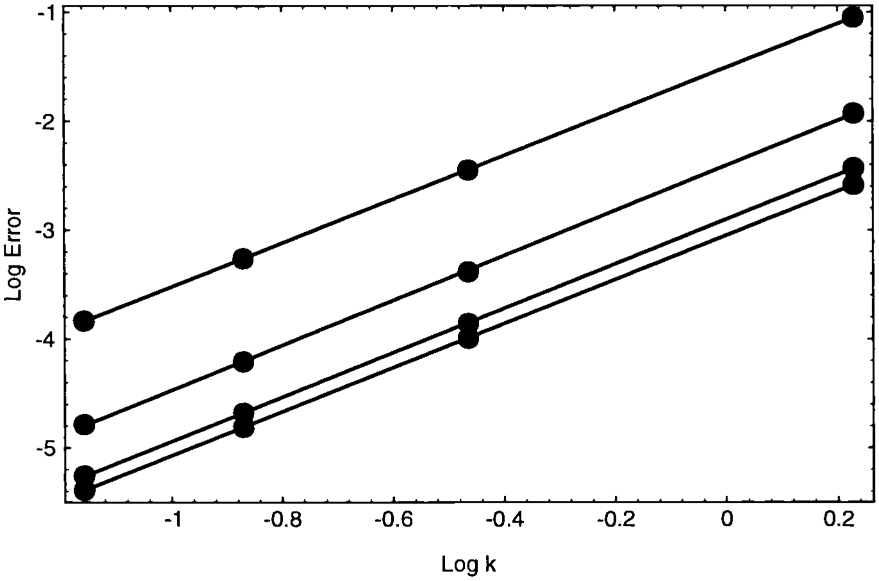


FIG. 5. *Least-squares fit of phase velocity error indicating second order accuracy for distorted hexahedral grids with shear $\theta = 0°, 15°, 30°, 45°$, respectively. The larger error corresponds to the larger shear angle.*

THEOREM 4. *Consider the mass matrix $C = [(\varepsilon \mathbf{W}_i, \mathbf{W}_j)]$ for an orthogonal hexahedral grid. Let $C = LL^t$ be the Cholesky factorization and $\tilde{L}\tilde{L}^t$ the incomplete factorization of C. Then $L = \tilde{L}$, i.e., there is no nonzero fill in the course of the Cholesky decomposition.*

Proof. The proof follows by carefully examining the inner most loop of the decomposition (see Algorithm 4.2.2 in [39])

$$C_{ij} = C_{ij} - C_{ik}C_{jk},$$

where $k < j \le i$. If $C_{ij} = 0$, i.e., there is no interaction between edges i and j, then there will be zero fill only if there is another edge k that interacts with both edges i and j. Numbering the edges sequentially precludes this possibility. This is illustrated on a two-dimensional grid in Figure 6.

It follows from the previous theorem that for the orthogonal case, the ICCG algorithm converges in one iteration. The above result does not hold for arbitrary hexahedral grids. However, as will be seen in the numerical results of the next section, the number of iterations for the ICCG algorithm to converge is quite small, indicating incomplete factorization is a very good preconditioner for the conjugate gradient algorithm.

The classical approach to dealing with the mass matrix is to "lump" it, whereby the matrix C is approximated by a diagonal matrix $\tilde{C}$ given by

$$\tilde{C}_{ii} = \sum_j \alpha_j C_{ij}, \qquad i = 1, \dots, N_e, \tag{28}$$

and the coefficients α_j are such that

$$\sum_j \alpha_j \mathbf{W}_j \bullet \mathbf{W}_i = \int_\Omega \mathbf{E} \bullet \mathbf{W}_i \, d\Omega. \tag{29}$$

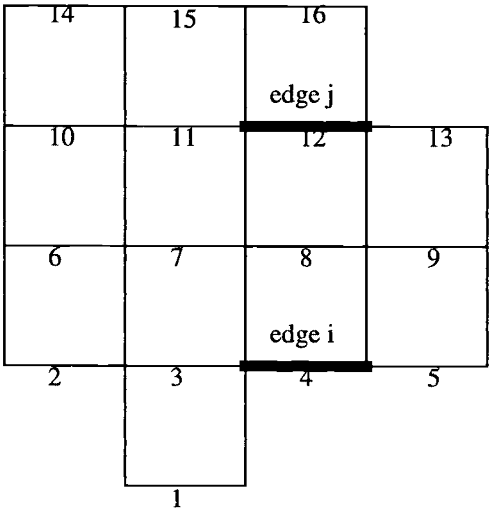


FIG. 6. *Grid numbering scheme for Cartesian grid.*

For a uniform orthogonal Cartesian grid and $\alpha_j = 1$, each diagonal term of the lumped mass matrix $\tilde{C}$ is equal to the row-sum of C and the mass lumping approximation generates the classic FDTD method. Thus, we see that the VFETD method is a generalization of the FDTD method.

7. Numerical experiments. In this section the VFETD method is used to solve several electromagnetics problems for which analytic solutions are known. In all cases the CPU time is for a Silicon Graphics 8000 workstation (64 bit, 300 MFLOPS, SPECfp92 310).

7.1. Spherical cavity. In this section a perfectly conducting spherical cavity of radius $a = 0.05855m$ is analyzed using VFETD and the computed solutions are compared to the exact analytical solution. The electric field within the cavity satisfies (20) where $\sigma_E = \sigma_M = 0$ and $\mu = \varepsilon = 1$ within the cavity. The exact solution is of the form

$$E = \sum_{np} A_{np} \mathbf{E}_{np}^{TE} \cos(\omega_{np} \mathbf{t} + \phi_{np}) + \sum_{np} B_{np} \mathbf{E}_{np}^{TM} \cos(\omega_{np} \mathbf{t} + \theta_{np}), \tag{30}$$

where the sum is over all the modes, and A, B, ϕ, and θ depend upon the initial conditions [37]. Here, ω_{np} are the resonant frequencies given by

$$\omega_{np}^{(j)} = \frac{\zeta_{np}^{(j)}}{a}, \qquad n, p = 1, 2, 3, \ldots, j = 0.1, \tag{31}$$

where $\zeta_{np}^{(j)}$ are the pth zeros of the jth derivative of the spherical Bessel function of order n. The exact resonant frequencies below 20Hz are shown in Table 3.

The spherical cavity was modeled using a sequence of hexahedral grids ranging from a coarse grid with 4 cells per radius to a fine grid with 12 cells per radius. Figure 7(a), (b) are cut-away views of the 256 hexahedral and 2048 hexahedral grids, respectively. The electromagnetic fields in the cavity were excited by a pulsed current source, the pulse having the shape of the second derivative of a Gaussian. The initial

TABLE 3
Exact value of resonant frequencies below 20*Hz.*

$\omega_{11}^{(1)}$	$\omega_{21}^{(1)}$	$\omega_{11}^{(0)}$	$\omega_{31}^{(1)}$	$\omega_{21}^{(0)}$	$\omega_{41}^{(1)}$	$\omega_{12}^{(1)}$	$\omega_{31}^{(0)}$
7.4589	10.5665	12.2132	13.518	15.6654	16.4782	16.6277	18.9953

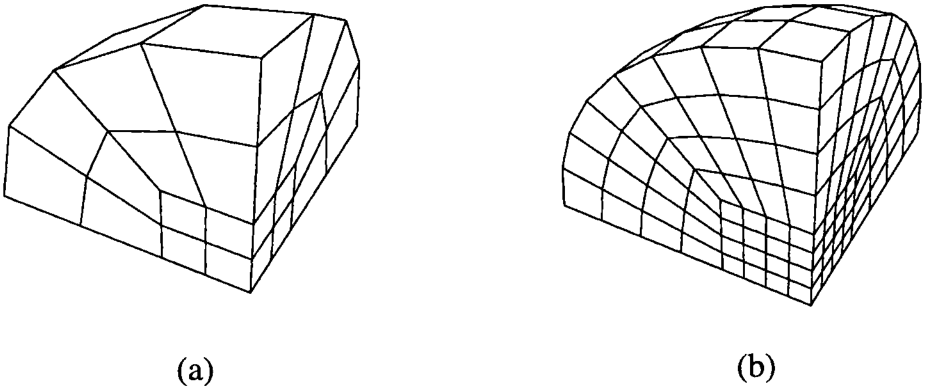


FIG. 7. *Internal view of* (a) 256, (b) 2048 *hexahedral grid of sphere.*

electric and magnetic fields within the cavity were zero. The simulation was run for $t = 6.71315s$, which corresponds to 50 periods of the lowest mode. An edge within the cavity was selected at random and the electric field along this edge was written to disk at every time step. This signal was weighted by a Hamming window and the signal was zero-extended to 32768 samples and then Fourier transformed. The magnitude of the Fourier transform is the *power spectrum* of the signal. The time step and the number of steps was different for each grid due to different stability requirements. The power spectrums for the 256 hexahedral case and the 2048 hexahedral case are shown in Figure 8(a), (b), respectively.

Naturally the power spectrum corresponding to the higher resolution grid is more accurate than the power spectrum corresponding to the lower resolution grid. The order of accuracy of the method is determined by performing a least-squares fit to the data where the error is defined to be the difference between the exact and computed values of $\omega_{31}^{(1)}$. In other words we assume that $|(\omega_{31}^{(1)})_{exact} - (\omega_{31}^{(1)})_{computed}| \propto h^m$ and we solve for the value of m that best models the results. Table 4 records the error as a function of grid size, where h is the average cell size. The logarithm of the error versus the logarithm of h/a is shown in Figure 9, along with a linear least-squares fit. The slope of the line is 2.028, so that the method is second order accurate, thus agreeing with the numerical dispersion analysis.

The CPU time for the calculations is shown in Table 5. The CPU time is for the time stepping part of the calculation only. For the above experiments the matrix fill time is approximately 1/50 of the total CPU time. The stopping criteria for the ICCG algorithm was $\|\boldsymbol{residual}\|_2/\|\boldsymbol{rhs}\|_2 \le 10^{-9}$, where $\boldsymbol{rhs}$ is the right-hand side. Note that the number of ICCG iterations does not increase as the grid is refined, indicating that the condition number of the mass matrix remains constant [38]. Therefore the computational cost per time step is proportional to the number of degrees of freedom.

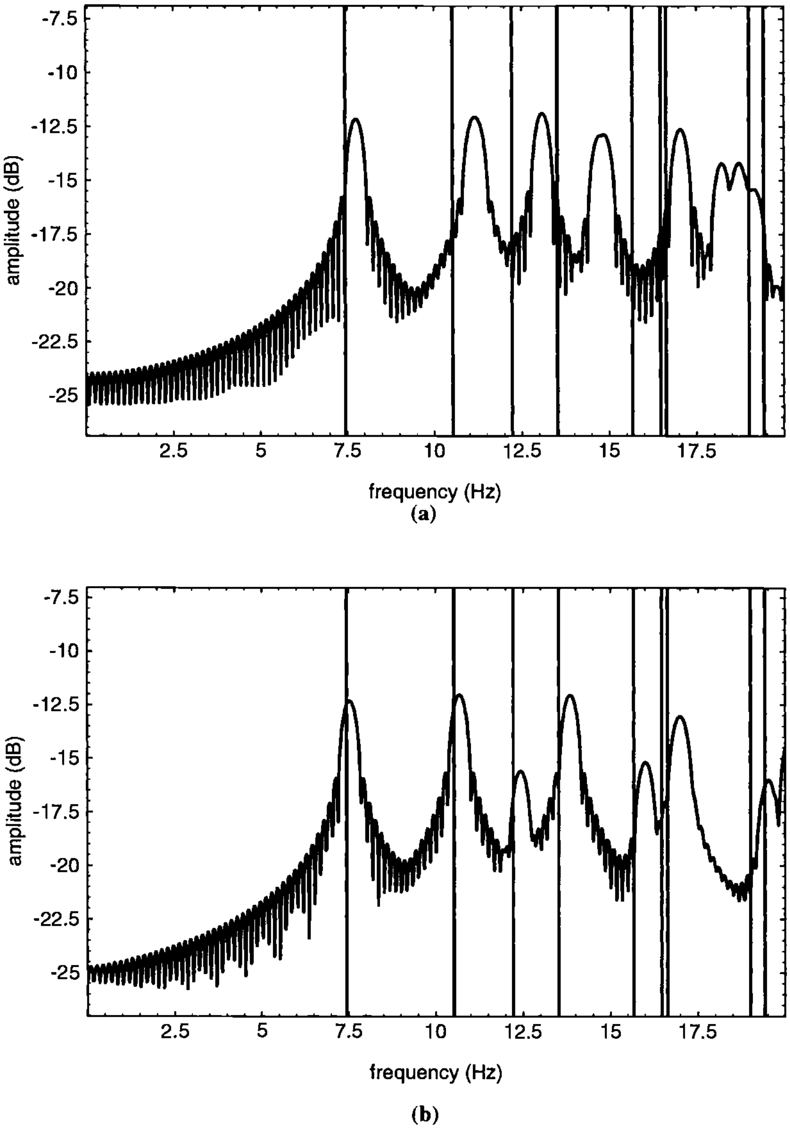


FIG. 8. *Computed power spectrum versus exact for* (a) 256, (b) 2048 *hexahedral sphere.*

TABLE 4
Relative error of $\omega_{31}^{(1)}$ *resonant frequency versus grid size for hexahedral grid.*

h/A	1/4	1/6	1/8	1/10	1/12
# nodes	321	997	2273	4341	7393
# cells	256	864	2048	4000	6912
# edges	688	2400	5792	11440	19920
error	0.09846	0.03960	0.02342	0.01589	0.009693

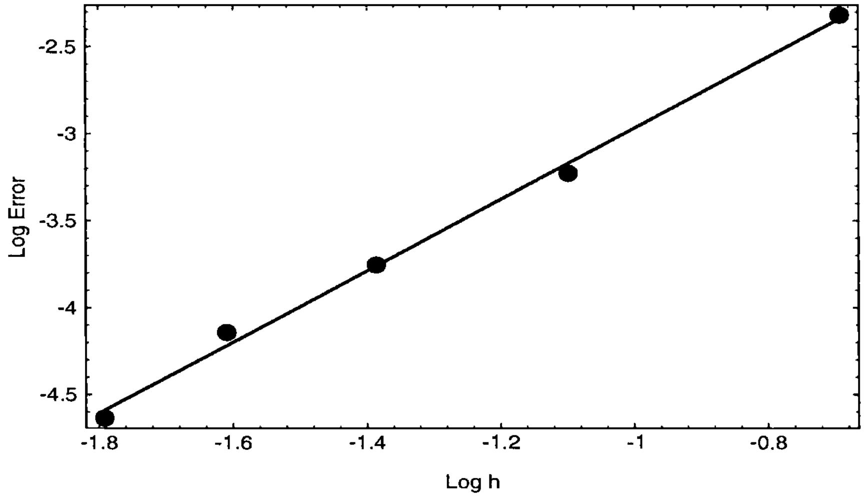


FIG. 9. *Linear fit indicating second order accuracy.*

TABLE 5
CPU time for cavity calculation versus grid size for hexahedral grid.

# edges	688	2400	5792	11440	19920
Δt	.0035	.002	.0015	.001	.001
# steps	1918	3356	4475	6713	6713
# ICCG iter.	7.8	7.8	7.8	7.8	7.8
CPU sec.	107	731	3255	11962	22490

7.2. Rectangular waveguide. In this section, the VFETD method is used to compute the electromagnetic fields in a rectangular waveguide. Let the rectangular waveguide have width $a = 0.9m$ in the x direction, height $b = 4.5m$ in the y direction, and infinite in the z direction. The fields are modeled by (1)–(3).

A wave is launched by forcing the time dependent boundary condition

$$\begin{aligned} \mathbf{E}_x &= 0, \\ \mathbf{E}_y &= \left(1 - \exp\left(-\left(\frac{t}{2T}\right)^2\right)\right) \sin(\pi x/a)\sin(\omega t) \end{aligned} \tag{32}$$

at the left end ($z = 0$) of the waveguide. Here $\omega = 5.523599$ and $T = 0.5$. The initial electric and magnetic fields in the guide are zero. The exact steady state solution is given by

$$\begin{aligned} \mathbf{E}_x &= 0, \\ \mathbf{E}_y &= A\sin(\pi x/a)\sin(\omega t - \beta_z z), \\ \mathbf{E}_z &= 0, \end{aligned} \tag{33}$$

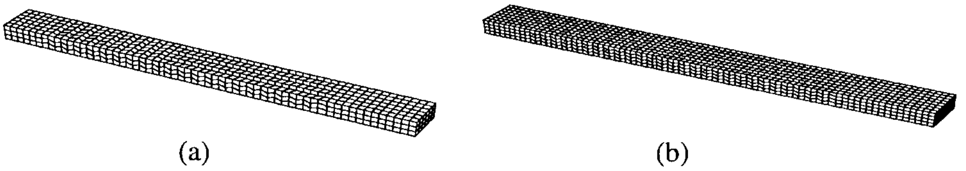


FIG. 10. *Rectangular waveguide model using* (a) 1080, (b) 2560 *chevron cells.*

$$\mathbf{B}_x = \frac{\beta_z}{\omega} A \sin(\pi x/a) \sin(\omega t - \beta_z z),$$
$$H_y = 0, \tag{34}$$
$$\mathbf{B}_z = A \frac{(\pi/a)^2}{\omega} \cos(\pi x/a) \cos(\omega t - \beta_z z),$$

where the wave number is $\beta_z = \sqrt{\omega^2 \mu\varepsilon - (\pi/a)^2}$; see [36].

The waveguide is modeled using a sequence of chevron grids with the coarsest grid having $h = a/6$ and the finest having $h = a/14$, where h is the average cell size. Several finite volume methods have been shown to be unconditionally unstable for these particular chevron grids [10]. Two of the grids are illustrated in Figure 10(a) and Figure 10(b).

The simulation was run for 20 seconds, which was enough time for the wavefront to propagate approximately 20 meters, i.e., twice the length of the finite guide.

The infinite waveguide is approximated by a finite length waveguide of length $10m$ with a radiation (or absorbing) boundary condition. The method used here to eliminate nonphysical reflections from the artificial truncation of the domain is a variant of the perfectly matched layer (PML) method. The original PML method derived in [44] is applicable only for the classic Cartesian grid FDTD method, but many variants have been proposed for unstructured grids [45, 46, 47]. The general idea is to attach to the truncated domain several layers of anisotropic conductive media, using both electric and magnetic conductivity. Grading the layers from low conductivity to high conductivity creates a broadband impedance match, thus eliminating (or nearly eliminating) front face reflections. As the outgoing wave propagates through the PML it is absorbed by the medium. The PML is not really perfect; a small amount of energy will be reflected from the boundary. But the reflection is an exponential function of layer thickness and can be made arbitrarily small. Since the VFETD method allows for arbitrary tensor material properties, the PML technique was used without modification. In this example, the wave will be attenuated by the PML at the right end of the waveguide; thus the simulation will reach a dynamic steady state condition that resembles the exact solution of an infinite waveguide.

In this simulation, a five-layer PML was used to absorb the outgoing wave. Each layer is defined by the tensor material properties μ, ε, σ_E, σ_M. In every layer μ and ε are identity matrices. The conductivity matrices are given by $\sigma_E = \sigma_M = \sigma$, where σ is a diagonal matrix with $\sigma_{xx} = \sigma_{yy} = \sigma_\perp$ and $\sigma_{zz} = 1$. The values of $\sigma_\perp$ used are tabulated in Table 6. The time step, number of steps, and ICCG iterations are shown in Table 7.

Note again that the number of ICCG iterations is constant. The same stopping criteria was the same as for the spherical cavity.

The computed electric and magnetic fields in the waveguide are compared to the

Table 6
PML parameters used for truncated waveguide.

	layer 1	layer 2	layer 3	layer 4	layer 5
$\sigma_\perp$	1.8	7.2	16.2	28.8	45

Table 7
CPU time for chevron waveguide calculations.

(h/a)	1/6	1/8	1/10	1/14	1/20	1/40
# cells	1080	2560	5000	14720	35800	271200
# edges	4425	9756	18215	47397	96469	771719
Δt	0.016666	0.0125	0.01	0.007142	.005	.0025
# steps	1272	1696	2120	2968	4240	8480
ICCG iter.	5.7	5.7	5.7	5.7	5.5	5.5

exact solution. The error measure is the standard L_2 relative error shown below, where the sum is over all the hexahedral cells (excluding PML cells) in the grid.

$$L_2\,error = \frac{\|(\mathbf{E}_{exact} - \mathbf{E}_{computed})\|_2}{\|\mathbf{E}_{exact}\|_2} \tag{35}$$

The logarithm of the error versus the logarithm of (h/a) is shown in Figure 11. In this figure the initial error is somewhat erratic until the grid spacing h/a is approximately 1/20, and after that the error decreases with second order convergence. This agrees with the analysis in [48] where it is shown that the Galerkin solution of the Helmholtz equation exhibits pollution due to numerical dispersion. In our example, the waveguide is ten wavelengths long and, for large values of h/a, the phase error builds up significantly along the length of the waveguide, degrading the L_2-norm of the error. As discussed in section 5, for a grid spacing of $(h/a) = 10$ and a grid distortion of $\theta = 30°$ the worst-case phase velocity error is approximately 3%; therefore at the termination end of the waveguide the fields are up to 100 degrees out of phase, resulting in a local error of up to 50%. The L_2 error for the $(h/a) = 10$ case is 17.38%, which is relatively poor. The convergence result in Figure 11 shows that for electrically large problems a very fine mesh is required in order to achieve a small L_2 error.

There are several other measures of error that are applicable for this specific waveguide problem. The global error as well as errors in impedance, wavelength, voltage standing wave ratio (VSWR), and reflection coefficient are shown in Table 8. In this case, the impedance in the guide is defined as $Z = E_z/H_y = \omega\mu/\beta_z =$ constant. Since the computed fields are "noisy," the computed impedance is defined to be the average impedance over the entire guide. The wavelength is computed by fitting a sine wave to the magnitude of the electric field, the period of best fit sine wave defining the wavelength of the electric field. The exact wavelength for this problem is simply $\lambda_z = 2\pi/\beta_z$. It is interesting to note that while the L_2 error is relatively large, accurate quantities such as impedance and wavelength can be derived from the computed field. Hence the L_2 error estimate may be overly pessimistic for some applications. The VSWR is defined as $VSWR = |E_{max}|/|E_{min}|$, where $|E_{max}|$ is the maximum of the time average electric field in the waveguide and $|E_{min}|$ is the minimum of the time

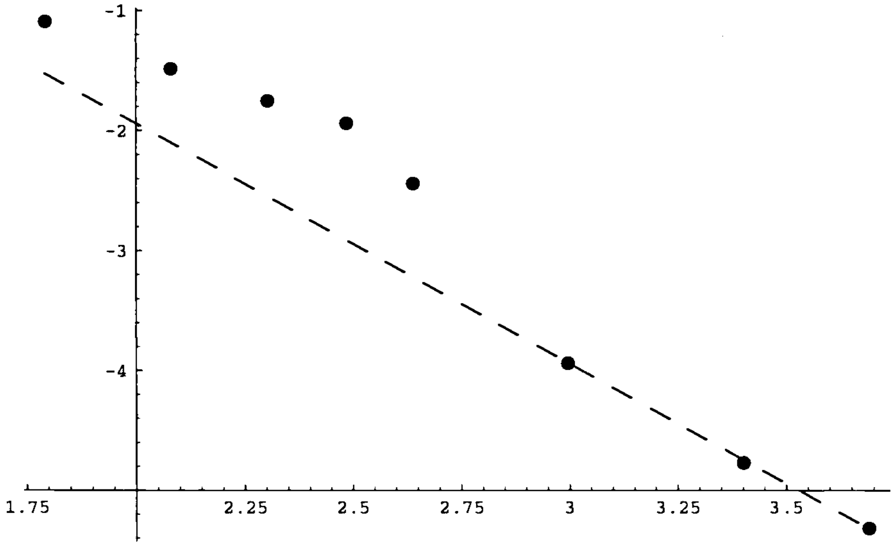


FIG. 11. *Log error versus* Log *h.*

TABLE 8
Quality of computed fields for PML terminated waveguide.

h/a	L_2	impedance	wavelength	VSWR	reflection coefficient
1/10	17.38%	2.713%	0.453%	1.057	−31dB

average electric field in the waveguide. For an infinite waveguide, or a perfectly terminated finite length waveguide, the VSWR is 1.0. For a terminated waveguide, the VSWR can be expressed as a function of the reflection coefficient of the termination, $VSWR = (1 - |\rho|)/(1 + |\rho|)$, where ρ is the reflection coefficient. The VSWR was computed by determining the maximum and minimum fields over one period, and the reflection coefficient is then computed from $|\rho| = (1 - VSWR)/(1 + VSWR)$. The reflection coefficient is a measure of the effectiveness of the PML. If the PML is perfect, the reflection coefficient would be zero. The reported reflection coefficient of −31dB is comparable to that obtained when using finite difference methods on uniform grids [44]. The reflection coefficient can in theory be reduced arbitrarily by adding more layers and/or tuning the material properties. The computed electric and magnetic fields are shown in Figure 12 for the 5000 cell waveguide, with excellent qualitative agreement with theory. Note that the chevron pattern of the underlying computational grid is not imprinted on the computed fields.

7.3. Dipole antenna. In this section, we compute the radiated electromagnetic fields due to a small current source using the VFETD method. Starting at the origin, a current oscillating at frequency ω is aligned in the z direction as illustrated in Figure 13.

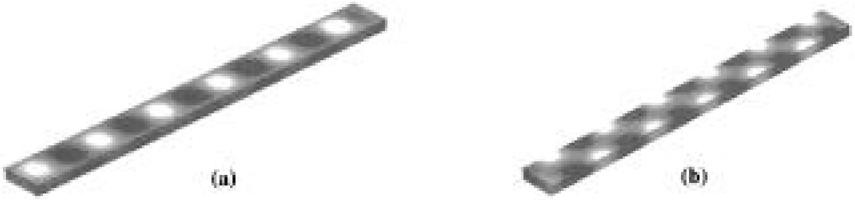


FIG. 12. *z-component of* (a) *electric field* (b) *magnetic field in PML terminated waveguide.*

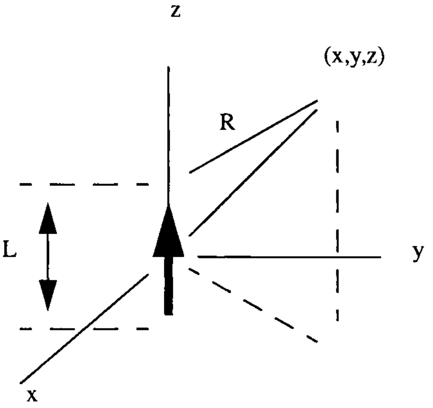


FIG. 13. *Coordinate system for dipole radiation calculation.*

The exact solution is given by

$$\mathbf{B} = \left(\mathbf{x}\frac{\partial A_z}{\partial y} - \mathbf{y}\frac{\partial A_z}{\partial x}\right)\cos(\omega t + \theta),$$
$$\mathbf{E} = \frac{1}{\omega\mu\varepsilon}\left(\mathbf{x}\frac{\partial\partial A_z}{\partial x\partial z} + \mathbf{y}\frac{\partial\partial A_z}{\partial y\partial z} - \mathbf{z}\left(\frac{\partial\partial A_z}{\partial x\partial x} + \frac{\partial\partial A_z}{\partial y\partial x}\right)\right)\sin(\omega t + \theta), \tag{36}$$

where

$$A_z = \frac{\mu I}{4\pi}\int_{-L/2}^{L/2}\frac{\exp(-j\beta R)}{R}dz, \tag{37}$$

and $\theta = \arg(A)$. The parameters for this computational experiment were $\omega = 107.3132$ and $L = \lambda/12 = 0.00487916$. The problem was modeled using a hemispherical grid consisting of 12032 hexahedral cells and 38005 edges. The grid had a spacing of $h = \lambda/24 = 0.00243972$ at the origin with the grid spacing increasing away from the origin. The current source is exactly two edge lengths long and given by

$$I(t) = \left(1 - \exp\left(-\left(\frac{t}{2T}\right)^2\right)\right)\sin(\omega t), \tag{38}$$

where $T = 0.0147$. The simulation was executed for 0.05855 seconds using a time step of $\Delta t = 10^{-4}$ seconds, which corresponds to 585 time steps.

In order to simulate free space, the same 5-layer PML used for the waveguide in the previous section was used with the exception that the conductivity tensor

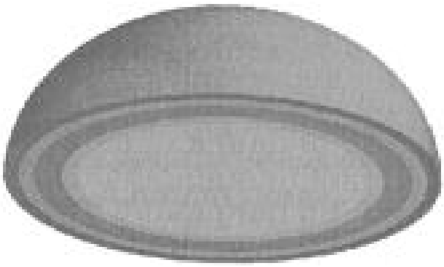


FIG. 14. *Illustration of hexahedral grid with* 5*-layer PML used for dipole calculation.*

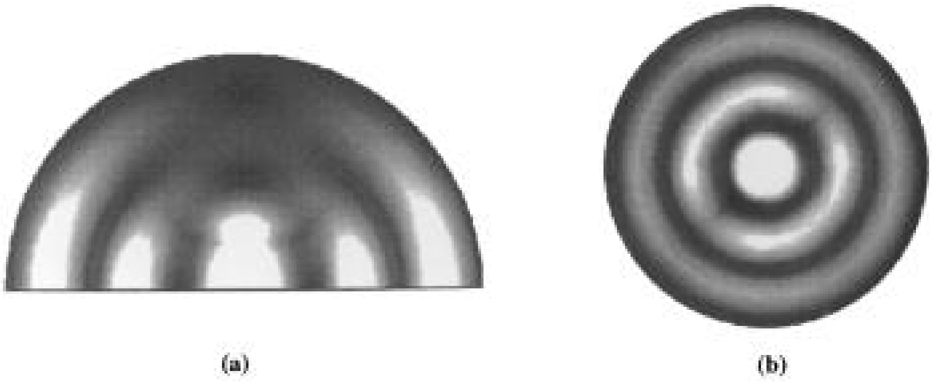


FIG. 15. *Computed* (a) *electric* (b) *magnetic field magnitude in the vicinity of* $\lambda/12$ *dipole.*

is rotated such that the axial direction corresponds to the radial direction so that the PML will absorb outgoing waves. The PML begins at radius $a = \lambda = 0.05855$ and the grid was terminated at $b = 1.5\lambda = 0.087825$; see Figure 14. The global L_2 error was computed in the same manner as for the waveguide, i.e., according to (35) where the sum is over all cells excluding PML cells. The computed electric field matched the exact electric field to within 1.6%, which is an excellent result since the electromagnetic field structure is quite complicated in the near field of the antenna. Snapshots of the computed electric and magnetic field are shown in Figure 15.

8. Conclusion. In this paper the VFETD method is derived, analyzed, and validated. It is demonstrated that the method is weakly stable, charge is conserved, and the continuity/discontinuity of the electromagnetic fields across a material interface are modeled properly. A numerical dispersion analysis indicates that the method is second order accurate, even on distorted, but regular, three-dimensional hexahedral grids.

However, like most finite element methods, the VFETD method requires that a sparse linear system be solved at every time step. The incomplete Cholesky conjugate gradient method was investigated where it is shown that the computational effort required to solve the system depends upon how distorted the grid is. However, for a uniformly refined mesh the number of iterations becomes independent of the number of mesh points; hence the method is scalable.

The VFETD method is validated by comparing computed solutions to analytical solutions for a simple resonant cavity, waveguide, and antenna. The accuracy and computer CPU time is tabulated for a variety of different grids. The computational experiments performed on these distorted hexahedral grids have rates of convergence that agree with previously published analytical approximations. It is also shown that the recently developed PML concept can be used to approximate an infinite space using a finite grid. Since the VFETD method allows for arbitrary tensor material properties, the PML concept is trivial to implement.

Acknowledgment. The authors would like to thank the Institute for Scientific Computing Research (ISCR) at the Lawrence Livermore National Laboratory (LLNL) for their support in this research.

REFERENCES

[1] K. S. Yee, *Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media*, IEEE Trans. Antenna and Propagation, 14 (1966), pp. 302–307.

[2] A. Taflove and M. E. Brodwin, *Numerical solution of steady-state electromagnetic scattering problems using the time-dependent Maxwell's equations*, IEEE Trans. Microwave Theory Tech., 23 (1975), pp. 623–630.

[3] A. Taflove, *Review of the formulation and applications of the finite difference time domain method for numerical modeling of electromagnetic wave interactions with arbitrary structures*, Wave Motion, 10 (1988), pp. 547–582.

[4] K. S. Kunz and R. J. Luebbers, *The Finite Difference Time Domain Method for Electromagnetics*, CRC Press, Boca Raton, FL, 1993.

[5] R. Holland, *The case against staircasing*, in Proceedings of the 6th annual Review of Progress in Applied Computational Electromagnetics, Monterey, CA, Applied Computation Electromagnetics Society, 1990, pp. 89–95.

[6] N. Madsen and R. Ziolkowski, *A 3 dimensional modified finite volume technique for Maxwell's equations*, Electromagnetics, 10 (1990), pp. 147–161.

[7] N. Madsen, *Divergence preserving discrete surface integral method for Maxwell's curl equations using non-orthogonal unstructured grids*, J. Comput. Phys., 119 (1995), pp. 34–45.

[8] R. Holland, V. Cable, and L. Wilson, *Finite-volume time-domain techniques for EM scattering*, IEEE Trans. Electromagnetic Compatibility, 33 (1991), pp. 281–294.

[9] K. S. Yee and J. S. Chen, *Conformal hybrid finite difference time domain and finite volume time domain*, IEEE Trans. Antennas and Propagation, 42 (1994), pp. 1450–1454.

[10] S. Brandon and P. Rambo, *Stability of the DSI electromagnetic update algorithm on a chevron grid*, in Proceedings of the 22nd IEEE International Conference on Plasma Science, Madison, WI, 1995.

[11] D. J. Riley and C. D. Turner, *VOLMAX: A solid model based transient volumetric Maxwell solver using hybrid grids*, IEEE Trans. Antennas and Propagation, 39 (1997), pp. 20–33.

[12] V. Shankar, W. Hall, and A. Mohammadian, *A time-domain differential solver for electromagnetic scattering problems*, Proc. IEEE, 77 (1989), pp. 709–721.

[13] V. Shankar, A. Mohammadian, and W. Hall, *A time-domain finite-volume treatment for the Maxwell equations*, Electromagnetics, 10 (1990), pp. 127–145.

[14] R. W. Noack and D. A. Anderson, *Time-Domain Solutions of Maxwell's Equations Using a Finite-Volume Formulation*, AIAA paper 92-0451, 1992.

[15] J. C. Nedelec, *Mixed finite elements in* $R3$, Numer. Math., 35 (1980), pp. 315–341.

[16] J. C. Nedelec, *A new family of mixed finite elements in* $R3$, Numer. Math., 50 (1986), pp. 57–81.

[17] A. Bossavit, *Whitney forms: A class of finite elements for three-dimensional computations in electromagnetism*, Proc. IEE-A, 135 (1988), pp. 493–500.

[18] A. Bossavit and I. Mayergoyz, *Edge elements for scattering problems*, IEEE Trans. Magnetics, 25 (1989), pp. 2816–2821.

[19] J. Lee, D. K. Sun, and Z. Cendes, *Tangential vector finite elements for electromagnetic field computation*, IEEE Trans. Magnetics, 27 (1991), pp. 4032–4035.

[20] A. Konrad, *Vector variational formulations of electromagnetic fields in anisotropic media*, IEEE Trans. Microwave Theory Tech., 24 (1976), pp. 533–559.

[21] A. Bossavit, *Solving Maxwell equations in a closed cavity, and the question of spurious modes*, IEEE Trans. Magnetics, 26 (1990), pp. 702–705.

[22] Z. J. Cendes, *Vector finite elements for electromagnetic field calculations*, IEEE Trans. Magnetics, 27 (1991), pp. 3958–3966.

[23] D. R. Lynch, K. D. Paulsen, and W. E. Boyse, *Synthesis of vector parasites in finite element Maxwell solutions*, IEEE Trans. Microwave Theory Tech., 41 (1998), pp. 1439–1447.

[24] B. Dillon and J. P. Webb, *A comparison of formulations for the vector finite element analysis of waveguides*, IEEE Trans. Microwave Theory Tech., 42 (1994), pp. 308–316.

[25] D. Sun, J. Magnes, X. Yuan, and Z. Cendes, *Spurious modes in finite element methods*, IEEE Antennas and Propagation, 37 (1995), pp. 12–24.

[26] J. Lee, D. Sun, and Z. Cendes, *Full-wave analysis of dielectric waveguides using tangential vector finite elements*, IEEE Trans. Microwave Theory Tech., 39 (1991), pp. 1262–1271.

[27] B. Anderson and Z. Cendes, *Solution of ferrite loaded waveguide using vector finite elements*, IEEE Trans. Magnetics, 31 (1995), pp. 1578–1581.

[28] B. Crain and A. Peterson, *Analysis of propagation on open microstrip lines using mixed-order covariant projection vector finite elements*, Int. J. Microwave and Millimeter-Wave CAD, 5 (1995), pp. 59–67.

[29] K. Mahadevan and R. Mittra, *Radar cross section computation of inhomogeneous scatterers using edge based finite element method in time and frequency domains*, Radio Science, 28 (1993), pp. 1181–1193.

[30] K. Mahadevan, R. Mittra, and P. M. Vaidya, *Use of Whitney's edge and face elements for efficient finite element time domain solution of Maxwell's equations*, J. Electromagn. Waves Appl., 8 (1994), pp. 1173–1191.

[31] J. F. Lee and Z. S. Sacks, *Whitney element time domain methods*, IEEE Trans. Magnetics, 31 (1995), pp. 1325–1329.

[32] J. F. Lee, R. Lee, and A. Cangellaris, *Time domain finite element methods*, IEEE Trans. Antennas and Propagation, 45 (1997), pp. 430–442.

[33] J. Lee, *WETD—A finite element time domain approach for solving Maxwell's equations*, IEEE Microwave Guided Wave Letters, 4 (1994), pp. 11–13.

[34] P. B. Monk, *A mixed method for approximating Maxwell's equations*, SIAM J. Numer. Anal., 28 (1991), pp. 1610–1634.

[35] P. B. Monk, *Analysis of a finite element method for Maxwell's equations*, SIAM J. Numer. Anal., 29 (1992), pp. 714–729.

[36] C. Balanis, *Advanced Engineering Electromagnetics*, John Wiley and Sons, New York, 1989.

[37] J. D. Stratton, *Electromagnetic Theory*, McGraw–Hill, New York, 1941.

[38] G. Strang and G. Fix, *An Analysis of the Finite Element Method*, Prentice–Hall, Englewood Cliffs, NJ, 1973.

[39] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, MD, 1989.

[40] P. B. Monk and A. K. Parrot, *A dispersion analysis of finite element methods for Maxwell's equations*, SIAM J. Sci. Comput., 15 (1994), pp. 916–937.

[41] G. Warren and W. Scott, *Numerical dispersion in the finite elements method using triangular edge elements*, Microwave Optical Tech. Letters, 9 (1995), pp. 315–319.

[42] D. White and G. Rodrigue, *Improved vector FEM solutions of Maxwell's equations using grid pre-conditioning*, Internat. J. Numer. Methods Engrg., 40 (1999), pp. 3815–3837.

[43] G. Warren and W. Scott, *An investigation of numerical dispersion in the vector finite element method using quadrilateral elements*, IEEE Trans. Antennas and Propagation, 42 (1994), pp. 1502–1508.

[44] J. Berenger, *A perfectly matched layer for the absorption of electromagnetic waves*, J. Comput. Phys., 114 (1994), pp. 185–200.

[45] U. Pekel and R. Mittra, *A finite element method frequency domain application of the PML concept*, Microwave and Optical Tech. Letters, 9 (1995), pp. 117–122.

[46] Z. Sacks, D. Kingsland, R. Lee, and J. Lee, *A perfectly matched anisotropic absorber for use as an absorbing boundary condition*, IEEE Trans. Antennas and Propagation, 43 (1995), pp. 1460–1463.

[47] M. Kuzuoglu and R. Mittra, *Mesh truncation by perfectly matched anisotropic absorbers in the finite element method*, Microwave and Optical Tech. Letters, 12 (1996), pp. 136–140.

[48] F. Ihlenburg and I. Babuska, *Dispersion analysis and error estimation of Galerkin finite element methods for the Helmholtz equation*, Internat. J. Numer. Methods Engrg., 38 (1995), pp. 3745–3774.

SIAM J. SCI. COMPUT.
Vol. 23, No. 3, pp. 707–740

SEMIDISCRETE CENTRAL-UPWIND SCHEMES FOR HYPERBOLIC CONSERVATION LAWS AND HAMILTON–JACOBI EQUATIONS[*]

ALEXANDER KURGANOV[†], SEBASTIAN NOELLE[‡], AND GUERGANA PETROVA[†]

Abstract. We introduce new Godunov-type semidiscrete central schemes for hyperbolic systems of conservation laws and Hamilton–Jacobi equations. The schemes are based on the use of more precise information about the local speeds of propagation and can be viewed as a generalization of the schemes from [A. Kurganov and E. Tadmor, *J. Comput. Phys.*, 160 (2000), pp. 241–282; A. Kurganov and D. Levy, *SIAM J. Sci. Comput.*, 22 (2000), pp. 1461–1488; A. Kurganov and G. Petrova, *A third-order semidiscrete genuinely multidimensional central scheme for hyperbolic conservation laws and related problems*, Numer. Math., to appear] and [A. Kurganov and E. Tadmor, *J. Comput. Phys.*, 160 (2000), pp. 720–742].

The main advantages of the proposed central schemes are the high resolution, due to the smaller amount of the numerical dissipation, and the simplicity. There are no Riemann solvers and characteristic decomposition involved, and this makes them a universal tool for a wide variety of applications.

At the same time, the developed schemes have an upwind nature, since they respect the directions of wave propagation by measuring the *one-sided* local speeds. This is why we call them *central-upwind* schemes.

The constructed schemes are applied to various problems, such as the Euler equations of gas dynamics, the Hamilton–Jacobi equations with convex and nonconvex Hamiltonians, and the incompressible Euler and Navier–Stokes equations. The incompressibility condition in the latter equations allows us to treat them both in their conservative and transport form. We apply to these problems the central-upwind schemes, developed separately for each of them, and compute the corresponding numerical solutions.

Key words. multidimensional conservation laws and Hamilton–Jacobi equations, high-resolution semidiscrete central schemes, compressible and incompressible Euler equations

AMS subject classifications. Primary, 65M06; Secondary, 35L65

PII. S1064827500373413

1. Introduction.

We consider Godunov-type schemes for the multidimensional (multi-D) systems of conservation laws

$$u_t + \nabla_{\mathbf{x}} \cdot f(u) = 0, \quad \mathbf{x} \in \mathbb{R}^d, \tag{1.1}$$

and the multi-D Hamilton–Jacobi equations

$$\varphi_t + H(\nabla_{\mathbf{x}}\varphi) = 0, \quad \mathbf{x} \in \mathbb{R}^d. \tag{1.2}$$

Godunov-type schemes for the system (1.1) are projection-evolution methods. Starting with cell averages at time level t^n, one reconstructs a piecewise polynomial interpolant of degree $r-1$ (where r is the formal order of the scheme), which is

[*] Received by the editors June 7, 2000; accepted for publication (in revised form) February 19, 2001; published electronically August 15, 2001.

http://www.siam.org/journals/sisc/23-3/37341.html

[†] Department of Mathematics, University of Michigan, Ann Arbor, MI 48109-1109 (kurganov@math.lsa.umich.edu, petrova@math.lsa.umich.edu). The first author was supported in part by a NSF Group Infrastructure grant and NSF grant DMS-0073631. The third author was supported by the Rackham Grant and Fellowship Program, and her visit to Bonn University was sponsored by DFG grant SFB256.

[‡] Institut für Geometrie und Praktische Mathematik, RWTH Aachen, Templergraben 55, 52056 Aachen, Germany (noelle@igpm.rwth-aachen.de). Part of the work of the second author was sponsored by DFG grant SFB256.

evolved to the next time level t^{n+1}, and then it is projected onto a space of piecewise constants. Depending on the projection step, we distinguish two kinds of Godunov-type schemes: central and upwind. The Godunov-type central schemes are based on exact evolution and averaging over Riemann fans. In contrast to the upwind schemes, they do not employ Riemann solvers and characteristic decomposition, which makes them *simple, efficient,* and *universal*.

In the one-dimensional (1-D) case, examples of such schemes are the first-order (staggered) Lax–Friedrichs scheme [28, 14], the second-order Nessyahu–Tadmor scheme [40], and the higher-order schemes in [39, 8, 30]. Second-order multi-D central schemes were introduced in [3, 4, 5, 6, 19, 34], and their higher-order extensions were developed in [31, 32]. We would also like to mention the central schemes for incompressible flows in [33, 22, 20, 21], and their applications to various systems, for example, [2, 13, 44, 49].

Unfortunately, these staggered central schemes may not provide a satisfactory resolution when small time steps are enforced by stability restrictions, which may occur, for example, in the application of these schemes to convection-diffusion problems. Also, they cannot be used for steady-state computations. These disadvantages are caused by the accumulation of numerical dissipation, which is of order $\mathcal{O}(\frac{(\Delta x)^{2r}}{\Delta t})$, where r is the formal order of the scheme.

The aforementioned problems have been recently resolved in [26], where new high-order Godunov-type central schemes are introduced. The proposed construction is based on the use of the CFL related *local speeds of propagation* and on integration over Riemann fans of variable sizes. In this way, a *nonstaggered* fully discrete central scheme is derived and is naturally reduced to a particularly simple semidiscrete form (for details see [26]). The same idea was used in [24] to develop a third-order semidiscrete central scheme, and in [25], where its genuinely multi-D extension was introduced.

The purpose of the first part of this paper is to present new semidiscrete central schemes for the conservation law (1.1), which we call *central-upwind* schemes. They are based on the *one-sided local speeds of propagation.* For example, in the 1-D case, these one-sided local speeds are the largest and smallest eigenvalues of the Jacobian $\frac{\partial f}{\partial u}$ (in contrast to the less precise local information, used in [26, 24, 25]—the spectral radius $\rho(\frac{\partial f}{\partial u})$).

The new schemes are Godunov-type *central* schemes, because the evolution step employs integration over Riemann fans and does not require a Riemann solver and a characteristic decomposition. They also have an *upwind* nature, since one-sided information is used to estimate the width of the Riemann fans. This more precise estimate makes our schemes less dissipative generalizations of the semidiscrete central schemes in [26, 24, 25].

The second part of this paper is devoted to the Hamilton–Jacobi equations, (1.2), which are closely related to (scalar) conservation laws. For example, in the 1-D case, the unique viscosity solution of the Hamilton–Jacobi equation, $\varphi_t + H(\varphi_x) = 0$, is the primitive of the unique entropy solution of the corresponding conservation law, $u_t + H(u)_x = 0$, where $u = \varphi_x$. However, in the multi-D case, this one-to-one correspondence no longer exists, but the gradient $\nabla_{\mathbf{x}}\varphi$ satisfies (at least formally) a system of (weakly) hyperbolic conservation laws.

This relation allows one to apply techniques, developed for conservation laws, to the derivation of the numerical methods for Hamilton–Jacobi equations. Examples of such methods can be found in [1, 10, 11, 35, 36, 42, 48]. One of the approaches [35, 36] is Godunov-type schemes. As in the case of conservation laws, they are projection-evolution methods. The differences are that here one starts with point values (not cell averages) at time t^n, builds a *continuous* piecewise polynomial reconstruction of degree r, and evolves it to the next time level t^{n+1}. The pointwise projection (not the cell averages) of this evolved solution is used as initial data for the next time step.

Godunov-type central schemes for Hamilton–Jacobi equations were first introduced in [35, 36]. Semidiscrete Godunov-type central schemes for the multi-D equations (1.2) were developed in [27], where the same idea of *local speeds of propagation* was used to separate between smooth and nonsmooth parts of the evolved solution.

In the second part of this work, we present a new second-order semidiscrete central-upwind scheme for the Hamilton–Jacobi equations (1.2). It is a less dissipative generalization of the scheme in [27], which uses more precise one-sided information of the local propagation speeds.

The paper is organized as follows. In section 2, we give a brief overview of the Godunov-type central schemes for conservation laws and Hamilton–Jacobi equations in one space dimension. We also describe the nonoscillatory piecewise quadratic reconstruction from [25], which is later used in the numerical examples. Next, in section 3, we introduce our new Godunov-type *central-upwind* schemes for the conservation laws and Hamilton–Jacobi equations, both in one and in two spatial dimensions. The results of our numerical experiments are presented in section 4. We apply the proposed scheme to a variety of test problems: the one- and two-dimensional compressible Euler equations, a 1-D Hamilton–Jacobi equation with a nonconvex Hamiltonian, the two-dimensional (2-D) eikonal equation of geometric optics. Finally, the incompressible Euler and Navier–Stokes equations are solved using two different approaches, based on either their conservative or transport form. The performed numerical experiments, especially in the case of incompressible flow simulations (see section 4.3), demonstrate the advantage of our new central-upwind approach.

2. Godunov-type central schemes—brief description. In this section, we review Godunov-type central schemes in one spatial dimension. We will consider only uniform grids and use the following notation: let $x_j := j\Delta x$, $x_{j\pm\frac{1}{2}} := (j \pm 1/2)\Delta x$, $t^n := n\Delta t$, $u_j^n := u(x_j, t^n)$, $\varphi_j^n := \varphi(x_j, t^n)$, where Δx and Δt are small spatial and time scales, respectively.

2.1. Central schemes for conservation laws. The starting point for the construction of Godunov-type schemes for conservation laws is the equivalent integral formulation of the system (1.1),

$$\begin{aligned}&\bar{u}(x, t+\Delta t)\\ &= \bar{u}(x,t) - \frac{1}{\Delta x}\left[\int_{\tau=t}^{t+\Delta t} f\left(u\left(x+\frac{\Delta x}{2},\tau\right)\right)d\tau - \int_{\tau=t}^{t+\Delta t} f\left(u\left(x-\frac{\Delta x}{2},\tau\right)\right)d\tau\right],\end{aligned} \tag{2.1}$$

where by

$$\bar{u}(x,t) := \frac{1}{\Delta x}\int_{I(x)} u(\xi,t)\,d\xi, \qquad I(x) = \left\{\xi : |\xi - x| < \frac{\Delta x}{2}\right\} \tag{2.2}$$

we denote the sliding averages of $u(\cdot,t)$ over the interval $(x - \frac{\Delta x}{2}, x + \frac{\Delta x}{2})$. At time level $t = t^n$ we consider problem (2.1) with the piecewise polynomial initial condition

$$\widetilde{u}(x,t^n) = p_j^n(x), \qquad x_{j-\frac{1}{2}} < x < x_{j+\frac{1}{2}} \quad \forall j, \tag{2.3}$$

obtained from the cell averages $\bar{u}_j^n := \bar{u}(x_j, t^n)$, computed at the previous time step. This piecewise polynomial reconstruction should be conservative, accurate of order r, and nonoscillatory.

Second-order schemes require a piecewise linear reconstruction (see the examples in [15, 16, 23, 29, 40, 43]). Third-order schemes employ a piecewise quadratic approximation, and one of the possibilities is to use the essentially nonoscillatory (ENO) reconstruction. In the 1-D case, we refer the reader to [16, 46]. The weighted ENO interpolants are proposed in [38, 18, 30, 32], and the multi-D ENO-type reconstructions can be found in [31, 32]. The ENO-type approach employs smoothness indicators. They require certain a priori information about the solution, which may be unavailable and then spurious oscillations or extra smearing of discontinuities may appear.

1-D nonoscillatory piecewise quadratic reconstructions, which do not require the use of smoothness indicators, were proposed in [37, 39, 25]. 2-D generalizations of these reconstructions were presented in [25, 41].

The reconstructed piecewise polynomial $\widetilde{u}(x, t^n)$ is then evolved exactly according to (2.1), and the solution at time $t = t^{n+1}$ is obtained in terms of its sliding averages, $\bar{u}(x, t^{n+1})$. An evaluation of these sliding averages at particular grid points provides the approximate cell averages of the solution at the next time level.

The choice of $x = x_j$ in (2.1) results in an upwind scheme. The solution then may be nonsmooth in the neighborhood of the points $\{x_{j+\frac{1}{2}}\}$, and the evaluation of the flux integrals in (2.1) requires the use of a computationally expensive (approximate) Riemann solver and characteristic decomposition.

If $x = x_{j+\frac{1}{2}}$ in (2.1), we obtain Godunov-type central schemes, namely,

$$\begin{aligned}\bar{u}_{j+\frac{1}{2}}^{n+1} = \frac{1}{\Delta x}\left[\int_{x_j}^{x_{j+\frac{1}{2}}} p_j^n(x)\,dx + \int_{x_{j+\frac{1}{2}}}^{x_{j+1}} p_{j+1}^n(x)\,dx\right] \\ -\frac{\lambda}{\Delta t}\left[\int_{t^n}^{t^{n+1}} f(u(x_{j+1},t))\,dt - \int_{t^n}^{t^{n+1}} f(u(x_j,t))\,dt\right], \quad \lambda := \frac{\Delta t}{\Delta x}.\end{aligned} \tag{2.4}$$

In contrast to the upwind framework, the solution is smooth in the neighborhood of the points $\{x_j\}$. Therefore, a discretization of the flux integrals in (2.4) can be done, using an appropriate quadrature formula. The corresponding function values can be computed either by Taylor expansion or by a Runge–Kutta method [39, 8].

2.2. Central schemes for Hamilton–Jacobi equations. In this section, we describe second-order Godunov-type central schemes for Hamilton–Jacobi equations. We follow the approach from [36] and construct a 1-D second-order staggered central scheme.

Assume that we have computed the point values of φ at time $t = t^n$. We then start with a continuous piecewise quadratic interpolant,

$$\begin{aligned}\widetilde{\varphi}(x, t^n) := \varphi_j^n + \frac{(\Delta\varphi)_{j+\frac{1}{2}}^n}{\Delta x}(x - x_j) + \frac{(\Delta\varphi)'_{j+\frac{1}{2}}}{2(\Delta x)^2}(x - x_j)(x - x_{j+1}), \\ x \in [x_j, x_{j+1}],\end{aligned} \tag{2.5}$$

where

$$(\Delta\varphi)_{j+\frac{1}{2}}^n := \varphi_{j+1}^n - \varphi_j^n. \tag{2.6}$$

Here, $(\Delta\varphi)'_{j+\frac{1}{2}}/(\Delta x)^2$ is an approximation to the second derivative $\varphi_{xx}(x_{j+\frac{1}{2}}, t^n)$. An appropriate *nonlinear limiter* employed in this approximation guarantees the nonoscillatory nature of $\widetilde{\varphi}(x, t^n)$. Examples of such limiters, developed in the context of hyperbolic conservation laws, may be found in [15, 16, 23, 40, 43]. In this paper we use a one-parameter family of the *minmod* limiters [29, 15, 43]

$$\begin{aligned}(\Delta\varphi)'_{j+\frac{1}{2}} = \text{minmod}\Bigg(&\theta\left[(\Delta\varphi)^n_{j+\frac{3}{2}} - (\Delta\varphi)^n_{j+\frac{1}{2}}\right], \frac{1}{2}\left[(\Delta\varphi)^n_{j+\frac{3}{2}} - (\Delta\varphi)^n_{j-\frac{1}{2}}\right],\\ &\theta\left[(\Delta\varphi)^n_{j+\frac{1}{2}} - (\Delta\varphi)^n_{j-\frac{1}{2}}\right]\Bigg),\end{aligned} \tag{2.7}$$

where $\theta \in [1, 2]$, and the multivariable minmod function is defined by

$$\text{minmod}(x_1, x_2, \ldots) := \begin{cases} \min_j\{x_j\} & \text{if } x_j > 0 \ \ \forall j, \\ \max_j\{x_j\} & \text{if } x_j < 0 \ \ \forall j, \\ 0 & \text{otherwise.} \end{cases} \tag{2.8}$$

Notice that larger θ's in (2.7) correspond to less dissipative, but still *nonoscillatory* limiters [29, 15, 43].

Given a reconstruction (2.5), we consider the Hamilton–Jacobi equation (1.2), subject to the initial data $\varphi(x, 0) = \widetilde{\varphi}(x, t^n)$. Under an appropriate CFL condition, due to the finite speed of propagation, the solution of this initial value problem is smooth in the neighborhood of the line segment $\{(x, t) : \ x = x_{j+\frac{1}{2}},\ t^n \leq t \leq t^{n+1}\}$. Therefore, from the Taylor expansion of the solution about the point $(x_{j+\frac{1}{2}}, t^n)$, we obtain

$$\varphi^{n+1}_{j+\frac{1}{2}} = \frac{\varphi^n_j + \varphi^n_{j+1}}{2} - \frac{(\Delta\varphi)'_{j+\frac{1}{2}}}{8} - \Delta t H\left(\frac{(\Delta\varphi)^n_{j+\frac{1}{2}}}{\Delta x}\right) + \frac{(\Delta t)^2}{2}\left[H'\left(\frac{(\Delta\varphi)^n_{j+\frac{1}{2}}}{\Delta x}\right)\right]^2 \cdot \frac{(\Delta\varphi)'_{j+\frac{1}{2}}}{(\Delta x)^2}. \tag{2.9}$$

Remarks.

1. The derived scheme (2.9) is different from the one in [36]. There, the evolution step is executed by integration of (1.2) over $[t^n, t^{n+1}]$, followed by the application of the midpoint rule to the resulting integrals. For details, see [36, 27].
2. A 2-D staggered central scheme for (1.2) can be found in [36].

3. Central-upwind semidiscrete schemes.

In this section, we develop new semidiscrete central-upwind schemes for conservation laws and Hamilton–Jacobi equations, following the approach presented in [26, 24, 25] and [27], respectively.

3.1. Semidiscrete central-upwind schemes for 1-D conservation laws.

We consider the 1-D system (1.1) of N strictly hyperbolic conservation laws. We start with a piecewise polynomial reconstruction (2.3) with possible discontinuities at the interface points $\{x_{j+\frac{1}{2}}\}$. These discontinuities propagate with right- and left-sided local speeds, which can be estimated by

$$a^+_{j+\frac{1}{2}} := \max_{\omega \in C\left(u^-_{j+\frac{1}{2}}, u^+_{j+\frac{1}{2}}\right)} \left\{\lambda_N\Big(\frac{\partial f}{\partial u}(\omega)\Big), 0\right\}$$

and

$$a^-_{j+\frac{1}{2}} := \min_{\omega \in C\left(u^-_{j+\frac{1}{2}}, u^+_{j+\frac{1}{2}}\right)} \left\{\lambda_1\Big(\frac{\partial f}{\partial u}(\omega)\Big), 0\right\},$$

respectively. Here, $\lambda_1 < \cdots < \lambda_N$ are the N eigenvalues of the Jacobian $\frac{\partial f}{\partial u}$, and $C(u^-_{j+\frac{1}{2}}, u^+_{j+\frac{1}{2}})$ is the curve in the phase space that connects

$$u^+_{j+\frac{1}{2}} := p_{j+1}(x_{j+\frac{1}{2}}) \quad \text{and} \quad u^-_{j+\frac{1}{2}} := p_j(x_{j+\frac{1}{2}}). \tag{3.1}$$

For example, in the genuinely nonlinear or linearly degenerate case, we have

$$\begin{aligned} a^+_{j+\frac{1}{2}} &= \max\left\{\lambda_N\Big(\frac{\partial f}{\partial u}\Big(u^-_{j+\frac{1}{2}}\Big)\Big), \lambda_N\Big(\frac{\partial f}{\partial u}\Big(u^+_{j+\frac{1}{2}}\Big)\Big), 0\right\}, \\ a^-_{j+\frac{1}{2}} &= \min\left\{\lambda_1\Big(\frac{\partial f}{\partial u}\Big(u^-_{j+\frac{1}{2}}\Big)\Big), \lambda_1\Big(\frac{\partial f}{\partial u}\Big(u^+_{j+\frac{1}{2}}\Big)\Big), 0\right\}. \end{aligned} \tag{3.2}$$

In fact, these one-sided local speeds are related to the CFL number. Note that in the schemes from [26, 24] only the spectral radius of $\frac{\partial f}{\partial u}$ is used, and for its computation one actually needs to know both λ_1 and λ_N.

Further, we utilize these one-sided local speeds of propagation in the following way. We consider the nonequal rectangular domains

$$[x^n_{j-\frac{1}{2},r}, x^n_{j+\frac{1}{2},l}] \times [t^n, t^{n+1}] \quad \text{and} \quad [x^n_{j+\frac{1}{2},l}, x^n_{j+\frac{1}{2},r}] \times [t^n, t^{n+1}], \tag{3.3}$$

with $x^n_{j+\frac{1}{2},l} := x_{j+\frac{1}{2}} + \Delta t a^-_{j+\frac{1}{2}}$ and $x^n_{j+\frac{1}{2},r} := x_{j+\frac{1}{2}} + \Delta t a^+_{j+\frac{1}{2}}$, where the solution of (1.1) with the initial data $\widetilde{u}(x, t^n)$ is smooth and nonsmooth, respectively.

The cell averages

$$\bar{w}^{n+1}_j = \frac{1}{x^n_{j+\frac{1}{2},l} - x^n_{j-\frac{1}{2},r}}\left[\int_{x^n_{j-\frac{1}{2},r}}^{x^n_{j+\frac{1}{2},l}} p^n_j(x)\,dx - \int_{t^n}^{t^{n+1}} \Big(f(u(x^n_{j+\frac{1}{2},l}, t)) - f(u(x^n_{j-\frac{1}{2},r}, t))\Big)\,dt\right], \tag{3.4}$$

and

$$\begin{aligned} \bar{w}^{n+1}_{j+\frac{1}{2}} = \frac{1}{x^n_{j+\frac{1}{2},r} - x^n_{j+\frac{1}{2},l}}\Bigg[&\int_{x^n_{j+\frac{1}{2},l}}^{x_{j+\frac{1}{2}}} p^n_j(x)\,dx + \int_{x_{j+\frac{1}{2}}}^{x^n_{j+\frac{1}{2},r}} p^n_{j+1}(x)\,dx \\ &- \int_{t^n}^{t^{n+1}} \Big(f(u(x^n_{j+\frac{1}{2},r}, t)) - f(u(x^n_{j+\frac{1}{2},l}, t))\Big)\,dt\Bigg] \end{aligned} \tag{3.5}$$

are obtained by integrating (1.1) over the corresponding domains in (3.3); see Figure 3.1.

Given the polynomials $\{p^n_j\}$, the spatial integrals in (3.4) and (3.5) can be computed explicitly. To discretize the flux integrals there, one may use an appropriate quadrature formula, since the solution is smooth along the line segments $(x^n_{j+\frac{1}{2},l}, t)$, $t^n \le t < t^{n+1}$ and $(x^n_{j+\frac{1}{2},r}, t)$, $t^n \le t < t^{n+1}$.

Next, from the cell averages, $\bar{w}^{n+1}_{j+\frac{1}{2}}$, $\bar{w}^{n+1}_j$, given by (3.4)–(3.5), we reconstruct a nonoscillatory, conservative, third-order, piecewise polynomial interpolant, denoted by

$$\widetilde{w}^{n+1}(x) = \sum_j \Big(\widetilde{w}^{n+1}_j(x)\chi_{\left[x^n_{j-\frac{1}{2},r},\, x^n_{j+\frac{1}{2},l}\right]} + \widetilde{w}^{n+1}_{j+\frac{1}{2}}(x)\chi_{\left[x^n_{j+\frac{1}{2},l},\, x^n_{j+\frac{1}{2},r}\right]}\Big). \tag{3.6}$$

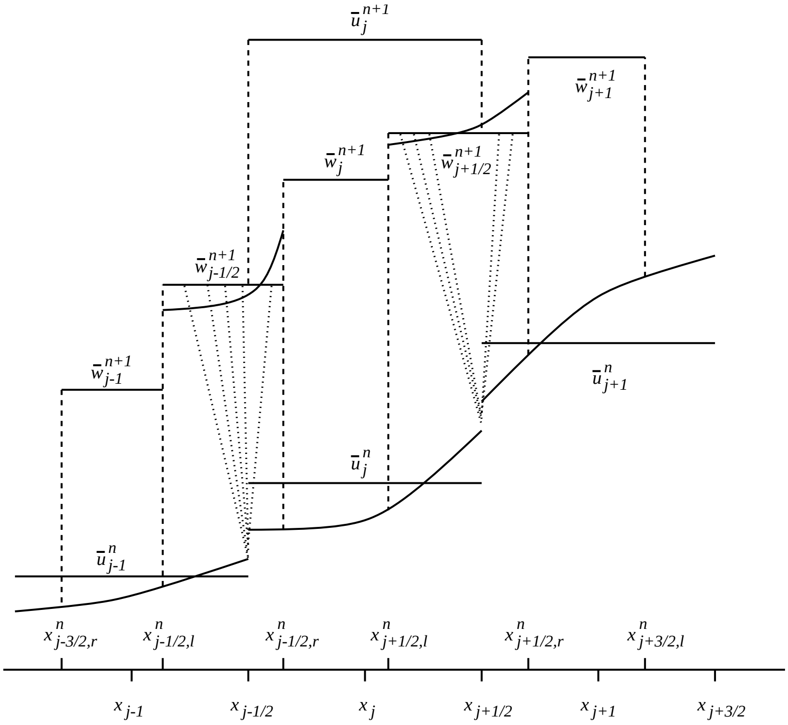


FIG. 3.1. *Central-upwind differencing.*

Here, the χ's are the characteristic functions, and $\{\widetilde{w}_{j+\frac{1}{2}}^{n+1}(x), \widetilde{w}_j^{n+1}(x)\}$ are the quadratic pieces, associated with the corresponding intervals. In fact, we do not need any high-order reconstruction $\widetilde{w}_j^{n+1}(x)$ since it will be averaged out (consult Figure 3.1).

Remark. Notice that even for a nonuniform grid, a particular piecewise quadratic reconstruction can be written explicitly. Since these formulae are rather messy and irrelevant for the semidiscrete scheme, we omit them.

The construction of our scheme is then completed by projecting $\widetilde{w}^{n+1}$ back onto the original grid, namely, we compute the cell averages

$$\bar{u}_j^{n+1} = \frac{1}{\Delta x}\int_{x_{j-\frac{1}{2}}}^{x_{j+\frac{1}{2}}} \widetilde{w}^{n+1}(x)\,dx \tag{3.7}$$

at the next time level. This leads to a fully discrete Godunov-type central-upwind scheme, which can be derived explicitly. Its derivation is similar to the derivation of the central schemes from [26, 24]. We omit here the details of these rather messy computations and continue within a much simpler semidiscrete framework.

The time derivative of $\bar{u}_j(t)$ is expressed with the help of (3.7) as

$$\frac{d}{dt}\bar{u}_j(t) = \lim_{\Delta t\to 0}\frac{\bar{u}_j^{n+1}-\bar{u}_j^n}{\Delta t} = \lim_{\Delta t\to 0}\frac{1}{\Delta t}\left[\frac{1}{\Delta x}\int_{x_{j-\frac{1}{2}}}^{x_{j+\frac{1}{2}}} \widetilde{w}^{n+1}(x)\,dx - \bar{u}_j^n\right]. \tag{3.8}$$

Now, let us suppose that the slopes of $\widetilde{w}_{j\pm\frac{1}{2}}^{n+1}$ are uniformly bounded, independently of

Δt. Since the width of the Riemann fans is bounded by $(a^+_{j+\frac{1}{2}} - a^-_{j+\frac{1}{2}})\Delta t$, we obtain

$$\widetilde{w}^{n+1}_{j\pm\frac{1}{2}}(x) = \bar{w}^{n+1}_{j\pm\frac{1}{2}} + \mathcal{O}(\Delta t) \qquad \forall x \in [x^n_{j\pm\frac{1}{2},l},\ x^n_{j\pm\frac{1}{2},r}]. \tag{3.9}$$

The conservation property of the reconstruction gives

$$\frac{1}{(x^n_{j+\frac{1}{2},l} - x^n_{j-\frac{1}{2},r})} \int_{x^n_{j-\frac{1}{2},r}}^{x^n_{j+\frac{1}{2},l}} \widetilde{w}^{n+1}_j(x)\,dx = \bar{w}^{n+1}_j. \tag{3.10}$$

From (3.8)–(3.10) and the definition of $x^n_{j-\frac{1}{2},r}$ and $x^n_{j+\frac{1}{2},l}$, we derive

$$\frac{d}{dt}\bar{u}_j(t) = \frac{a^+_{j-\frac{1}{2}}}{\Delta x} \lim_{\Delta t\to 0} \bar{w}^{n+1}_{j-\frac{1}{2}} + \lim_{\Delta t\to 0} \frac{1}{\Delta t}\left(\frac{x^n_{j+\frac{1}{2},l} - x^n_{j-\frac{1}{2},r}}{\Delta x}\bar{w}^{n+1}_j - \bar{u}^n_j \right) - \frac{a^-_{j+\frac{1}{2}}}{\Delta x} \lim_{\Delta t\to 0} \bar{w}^{n+1}_{j+\frac{1}{2}}. \tag{3.11}$$

The three limits in (3.11) are computed separately. Using (3.4) and (3.5), we obtain

$$\lim_{\Delta t\to 0} \frac{1}{\Delta t}\left(\frac{x^n_{j+\frac{1}{2},l} - x^n_{j-\frac{1}{2},r}}{\Delta x}\bar{w}^{n+1}_j - \bar{u}^n_j \right) = \frac{a^-_{j+\frac{1}{2}}u^-_{j+\frac{1}{2}} - a^+_{j-\frac{1}{2}}u^+_{j-\frac{1}{2}}}{\Delta x} - \frac{f(u^-_{j+\frac{1}{2}}) - f(u^+_{j-\frac{1}{2}})}{\Delta x}, \tag{3.12}$$

and

$$\lim_{\Delta t\to 0} \bar{w}^{n+1}_{j+\frac{1}{2}} = \frac{a^+_{j+\frac{1}{2}}u^+_{j+\frac{1}{2}} - a^-_{j+\frac{1}{2}}u^-_{j+\frac{1}{2}}}{a^+_{j+\frac{1}{2}} - a^-_{j+\frac{1}{2}}} - \frac{f(u^+_{j+\frac{1}{2}}) - f(u^-_{j+\frac{1}{2}})}{a^+_{j+\frac{1}{2}} - a^-_{j+\frac{1}{2}}}, \tag{3.13}$$

where, similarly to (3.1), $u^\pm_{j+\frac{1}{2}}$ stand for the corresponding right and left values of the piecewise polynomial interpolant $\{p_j\}$, reconstructed at time t.

Finally, a substitution of (3.12) and (3.13) in (3.11) results in our new semidiscrete central-upwind scheme, which can be written in the following conservative form:

$$\frac{d}{dt}\bar{u}_j(t) = -\frac{H_{j+\frac{1}{2}}(t) - H_{j-\frac{1}{2}}(t)}{\Delta x}. \tag{3.14}$$

Here, the numerical fluxes $H_{j+\frac{1}{2}}$ are given by

$$H_{j+\frac{1}{2}}(t) := \frac{a^+_{j+\frac{1}{2}}f(u^-_{j+\frac{1}{2}}) - a^-_{j+\frac{1}{2}}f(u^+_{j+\frac{1}{2}})}{a^+_{j+\frac{1}{2}} - a^-_{j+\frac{1}{2}}} + \frac{a^+_{j+\frac{1}{2}}a^-_{j+\frac{1}{2}}}{a^+_{j+\frac{1}{2}} - a^-_{j+\frac{1}{2}}}\left[u^+_{j+\frac{1}{2}} - u^-_{j+\frac{1}{2}}\right]. \tag{3.15}$$

Remarks.

1. The new semidiscrete scheme (3.14)–(3.15) is a Godunov-type central scheme, since it is based on integration over Riemann fans. It does not require characteristic decompositions and Riemann solvers, and therefore it preserves the main advantage of the central schemes—simplicity.
2. As with the semidiscrete schemes, proposed in [26], the numerical viscosity of (3.14)–(3.15) is independent of $\mathcal{O}(1/\Delta t)$, and thus it can be used for steady-state computations. Moreover, due to a more accurate estimate of the widths of the Riemann fans, the numerical dissipation in (3.14)–(3.15) is even smaller than the numerical viscosity of the schemes from [26, 24]. Notice that if

one takes $a^+_{j+\frac{1}{2}} = -a^-_{j+\frac{1}{2}} = a_{j+\frac{1}{2}} := \max_{\omega\in C(u^{n-}_{j+\frac{1}{2}}, u^{n+}_{j+\frac{1}{2}})} \rho(\frac{\partial f}{\partial u}(\omega))$, then the numerical flux (3.15) reduces to

$$H_{j+\frac{1}{2}}(t) := \frac{f(u^+_{j+\frac{1}{2}}) + f(u^-_{j+\frac{1}{2}})}{2} - \frac{a_{j+\frac{1}{2}}}{2}\left[u^+_{j+\frac{1}{2}} - u^-_{j+\frac{1}{2}}\right],$$

which is the numerical flux of the schemes from [26, 24].

3. We would like to point out that the first-order version of our scheme is exactly the semidiscrete version of the scheme in [17, 12]. Moreover, if the flux f is monotone, it reduces to the standard upwind scheme. That is why we call our new schemes *central-upwind*. For example, if $f'(u) \geq 0$, then $a^-_{j+\frac{1}{2}} = 0$ $\forall j$, and the first-order scheme simplifies to

$$\dot{u}_j(t) = -\frac{f(u^n_j) - f(u^n_{j-1})}{\Delta x}.$$

4. A fully discrete, 2-D, third-order accurate scheme using the Harten–Lax–van Leer approximate Riemann solver [17, 12] was implemented and tested in [41].
5. It can be proved that a scalar second-order version of (3.14)–(3.15), together with the minmod reconstruction,

$$\widetilde{u}^n_j(x) = \bar{u}^n_j + s^n_j(x - x_j),$$
$$s^n_j = \text{minmod}\left(\theta\frac{\bar{u}^n_j - \bar{u}^n_{j-1}}{\Delta x}, \frac{\bar{u}^n_{j+1} - \bar{u}^n_{j-1}}{2\Delta x}, \theta\frac{\bar{u}^n_{j+1} - \bar{u}^n_j}{\Delta x}\right), \tag{3.16}$$

is a TVD scheme (for $1 \leq \theta \leq 2$), that is, $\|u(\cdot,t)\|_{BV} \leq \|u(\cdot,0)\|_{BV}$. The proof is analogous to the proof of Theorem 4.1 in [26], and we leave the details to the reader.
6. The semidiscrete scheme (3.14)–(3.15) is a system of time-dependent ODEs, which can be solved by any stable ODE solver which retains the spatial accuracy of the semidiscrete scheme. In the numerical examples below, we have used the TVD Runge–Kutta method, proposed in [47, 45].
7. The scheme (3.14)–(3.15) can be easily generalized and applied to convection-diffusion equations in a straightforward manner. For details, we refer the reader to [26, 24].

3.2. Semidiscrete central-upwind schemes for multi-D conservation laws. The semidiscrete central-upwind schemes, presented in section 3.1, can be generalized to the multi-D case. Without loss of generality, we consider the 2-D system

$$u_t + f(u)_x + g(u)_y = 0. \tag{3.17}$$

Given the grid points $x_j := j\Delta x$, $y_k := k\Delta y$ and the intermediate points $x_{j\pm\frac{1}{2}} := x_j \pm \frac{\Delta x}{2}$, $y_{k\pm\frac{1}{2}} := y_k \pm \frac{\Delta y}{2}$, we start at time $t = t^n$ with a conservative piecewise polynomial reconstruction of an appropriate order:

$$\widetilde{u}^n(x,y) := \sum_{j,k} p^n_{j,k}(x,y)\chi_{j,k},$$

where $\chi_{j,k}$ is the characteristic function of the cell $[x_{j-\frac{1}{2}}, x_{j+\frac{1}{2}}] \times [y_{k-\frac{1}{2}}, y_{k+\frac{1}{2}}]$. In the numerical examples in this paper, we have used the third-order piecewise quadratic reconstruction, described in [25].

We use the notation

$$u_{j,k} := p^n_{j,k}(x_j, y_k), \quad u^{\mathrm{N}}_{j,k} := p^n_{j,k}(x_j, y_{k+\frac{1}{2}}), \quad u^{\mathrm{S}}_{j,k} := p^n_{j,k}(x_j, y_{k-\frac{1}{2}}),$$

$$u^{\mathrm{E}}_{j,k} := p^n_{j,k}(x_{j+\frac{1}{2}}, y_k), \quad u^{\mathrm{W}}_{j,k} := p^n_{j,k}(x_{j-\frac{1}{2}}, y_k), \quad u^{\mathrm{NE}}_{j,k} := p^n_{j,k}(x_{j+\frac{1}{2}}, y_{k+\frac{1}{2}}), \tag{3.18}$$

$$u^{\mathrm{NW}}_{j,k} := p^n_{j,k}(x_{j-\frac{1}{2}}, y_{k+\frac{1}{2}}), \quad u^{\mathrm{SE}}_{j,k} := p^n_{j,k}(x_{j+\frac{1}{2}}, y_{k-\frac{1}{2}}), \quad u^{\mathrm{SW}}_{j,k} := p^n_{j,k}(x_{j-\frac{1}{2}}, y_{k-\frac{1}{2}})$$

for the corresponding point values and

$$\bar{u}_{j,k} := \frac{1}{\Delta x \Delta y} \int_{x_{j-\frac{1}{2}}}^{x_{j+\frac{1}{2}}} \int_{y_{k-\frac{1}{2}}}^{y_{k+\frac{1}{2}}} p^n_{j,k}(x,y)\, dx\, dy$$

for the cell averages.

The piecewise polynomial interpolant $\widetilde{u}^n$ may have discontinuities along the lines $x = x_{j\pm\frac{1}{2}}$ and $y = y_{k\pm\frac{1}{2}}$, which propagate with different right- and left-sided local speeds. To estimate them is a nontrivial problem, but in practice one may use

$$\begin{aligned}
a^+_{j+\frac{1}{2},k} &:= \max\left\{\lambda_N\Big(\frac{\partial f}{\partial u}(u^{\mathrm{W}}_{j+1,k})\Big), \lambda_N\Big(\frac{\partial f}{\partial u}(u^{\mathrm{E}}_{j,k})\Big), 0\right\}, \\
b^+_{j,k+\frac{1}{2}} &:= \max\left\{\lambda_N\Big(\frac{\partial g}{\partial u}(u^{\mathrm{S}}_{j,k+1})\Big), \lambda_N\Big(\frac{\partial g}{\partial u}(u^{\mathrm{N}}_{j,k})\Big), 0\right\}, \\
a^-_{j+\frac{1}{2},k} &:= \min\left\{\lambda_1\Big(\frac{\partial f}{\partial u}(u^{\mathrm{W}}_{j+1,k})\Big), \lambda_1\Big(\frac{\partial f}{\partial u}(u^{\mathrm{E}}_{j,k})\Big), 0\right\}, \\
b^-_{j,k+\frac{1}{2}} &:= \min\left\{\lambda_1\Big(\frac{\partial g}{\partial u}(u^{\mathrm{S}}_{j,k+1})\Big), \lambda_1\Big(\frac{\partial g}{\partial u}(u^{\mathrm{N}}_{j,k})\Big), 0\right\},
\end{aligned} \tag{3.19}$$

respectively. As in [25], we consider the nonuniform domains, outlined in Figure 3.2 and defined by

$$D_{j,k+\frac{1}{2}} := [x_{j-\frac{1}{2}} + A^+_{j-\frac{1}{2},k+\frac{1}{2}}\Delta t,\ x_{j+\frac{1}{2}} + A^-_{j+\frac{1}{2},k+\frac{1}{2}}\Delta t] \times [y_{k+\frac{1}{2}} + b^-_{j,k+\frac{1}{2}}\Delta t,\ y_{k+\frac{1}{2}} + b^+_{j,k+\frac{1}{2}}\Delta t],$$

$$D_{j+\frac{1}{2},k} := [x_{j+\frac{1}{2}} + a^-_{j+\frac{1}{2},k}\Delta t,\ x_{j+\frac{1}{2}} + a^+_{j+\frac{1}{2},k}\Delta t] \times [y_{k-\frac{1}{2}} + B^+_{j+\frac{1}{2},k-\frac{1}{2}}\Delta t,\ y_{k+\frac{1}{2}} + B^-_{j+\frac{1}{2},k+\frac{1}{2}}\Delta t],$$

$$\begin{aligned}
D_{j+\frac{1}{2},k+\frac{1}{2}} := {}& [x_{j+\frac{1}{2}} + A^-_{j+\frac{1}{2},k+\frac{1}{2}}\Delta t,\ x_{j+\frac{1}{2}} + A^+_{j+\frac{1}{2},k+\frac{1}{2}}\Delta t] \\
& \times [y_{k+\frac{1}{2}} + B^-_{j+\frac{1}{2},k+\frac{1}{2}}\Delta t,\ y_{k+\frac{1}{2}} + B^+_{j+\frac{1}{2},k+\frac{1}{2}}\Delta t],
\end{aligned}$$

$$D_{j,k} := [x_{j-\frac{1}{2}},\ x_{j+\frac{1}{2}}] \times [y_{k-\frac{1}{2}},\ y_{k+\frac{1}{2}}] \ \Big\backslash \ \bigcup_{\pm} [D_{j,k\pm\frac{1}{2}} \cup D_{j\pm\frac{1}{2},k} \cup D_{j\pm\frac{1}{2},k\pm\frac{1}{2}}],$$

where

$$A^+_{j+\frac{1}{2},k+\frac{1}{2}} := \max\left\{a^+_{j+\frac{1}{2},k}, a^+_{j+\frac{1}{2},k+1}\right\}, \qquad B^+_{j+\frac{1}{2},k+\frac{1}{2}} := \max\left\{b^+_{j,k+\frac{1}{2}}, b^+_{j+1,k+\frac{1}{2}}\right\},$$

$$A^-_{j+\frac{1}{2},k+\frac{1}{2}} := \min\left\{a^-_{j+\frac{1}{2},k}, a^-_{j+\frac{1}{2},k+1}\right\}, \qquad B^-_{j+\frac{1}{2},k+\frac{1}{2}} := \min\left\{b^-_{j,k+\frac{1}{2}}, b^-_{j+1,k+\frac{1}{2}}\right\}.$$

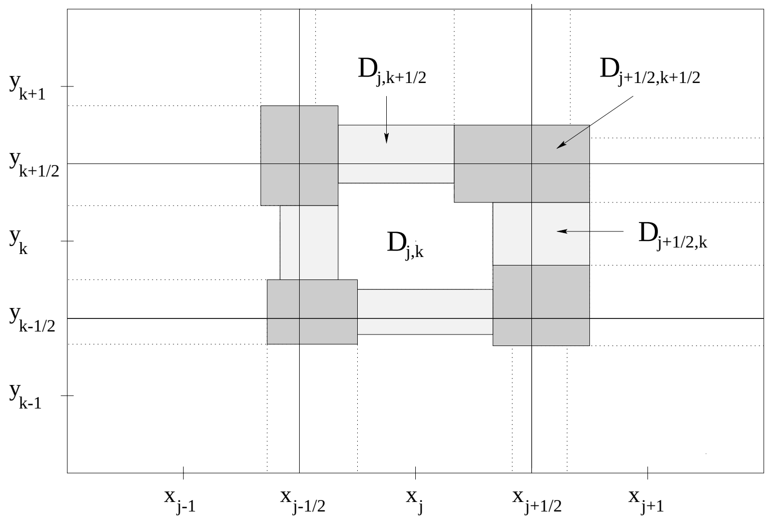


FIG. 3.2. *2-D central-upwind differencing.*

Similarly to the 1-D case, under an appropriate CFL condition, the solution of system (3.17) with initial data $\widetilde{u}(x,y)$ is smooth in the domain $D_{j,k}$ and may be nonsmooth in the other domains. Notice that, in general, $D_{j,k}$ is a nonrectangular domain inside the (j,k)-cell; see Figure 3.2.

An integration of the system (3.17) over the domains

$$D_{j,k} \times [t^n, t^{n+1}], \quad D_{j\pm\frac{1}{2},k} \times [t^n, t^{n+1}], \quad D_{j,k\pm\frac{1}{2}} \times [t^n, t^{n+1}], \quad D_{j\pm\frac{1}{2},k\pm\frac{1}{2}} \times [t^n, t^{n+1}]$$

results in new cell averages $\{\bar{w}^{n+1}_{j,k+\frac{1}{2}}\}$, $\{\bar{w}^{n+1}_{j+\frac{1}{2},k}\}$, $\{\bar{w}^{n+1}_{j+\frac{1}{2},k+\frac{1}{2}}\}$, and $\{\bar{w}^{n+1}_{j,k}\}$. They are used for an intermediate piecewise polynomial reconstruction,

$$\widetilde{w}^{n+1}(x,y) := \sum_{j,k} \left[\widetilde{w}^{n+1}_{j,k}\widetilde{\chi}_{j,k} + \widetilde{w}^{n+1}_{j+\frac{1}{2},k}\widetilde{\chi}_{j+\frac{1}{2},k} + \widetilde{w}^{n+1}_{j,k+\frac{1}{2}}\widetilde{\chi}_{j,k+\frac{1}{2}} + \widetilde{w}^{n+1}_{j+\frac{1}{2},k+\frac{1}{2}}\widetilde{\chi}_{j+\frac{1}{2},k+\frac{1}{2}} \right]. \tag{3.20}$$

Here, similarly to (3.6), $\{\widetilde{w}^{n+1}_{j,k}(x,y), \widetilde{w}^{n+1}_{j+\frac{1}{2},k}(x,y), \widetilde{w}^{n+1}_{j,k+\frac{1}{2}}(x,y), \widetilde{w}^{n+1}_{j+\frac{1}{2},k+\frac{1}{2}}(x,y)\}$ are the quadratic pieces, and the $\widetilde{\chi}$'s stand for the characteristic functions of the corresponding domains D.

The construction of our 2-D fully discrete central-upwind scheme is then completed by projecting the interpolant (3.20) back onto the original cells,

$$\bar{u}^{n+1}_{j,k} = \frac{1}{\Delta x \Delta y} \int_{x_{j-\frac{1}{2}}}^{x_{j+\frac{1}{2}}} \int_{y_{k-\frac{1}{2}}}^{y_{k+\frac{1}{2}}} \widetilde{w}^{n+1}(x,y)\,dxdy. \tag{3.21}$$

The derivation of the explicit form of the fully discrete higher-order scheme is omitted, since it is rather complicated and is of no practical use. Note, however, that Wendroff [50] has recently proposed a 2-D version of the Harten–Lax–van Leer Riemann solver, which is closely related to the first-order fully discrete version of our scheme.

As in [25, section 3.3], we continue within the semidiscrete framework (as $\Delta t \to 0$), where all the computations are much simpler.

We use the following notation for the intersections of the cell $[x_{j-\frac{1}{2}}, x_{j+\frac{1}{2}}] \times [y_{k-\frac{1}{2}}, y_{k+\frac{1}{2}}]$ with the domains D – $C_{j\pm\frac{1}{2},k\pm\frac{1}{2}}$ for the four corners, $S_{j\pm\frac{1}{2},k}$, $S_{j,k\pm\frac{1}{2}}$ for the four side domains, and $D_{j,k}$ for the center. The sizes of these domains are $|C| \sim (\Delta t)^2$ and $|S| \sim \Delta t$. Since we assume that the spatial derivatives of $\widetilde{w}^{n+1}$ are bounded independently of Δt, the relation between $\widetilde{w}^{n+1}$ and $\bar{w}^{n+1}$ is given by

$$\int\!\!\int_{C_{j\pm\frac{1}{2},k\pm\frac{1}{2}}} \widetilde{w}^{n+1}_{j\pm\frac{1}{2},k\pm\frac{1}{2}}\, dxdy = \mathcal{O}((\Delta t)^2), \tag{3.22}$$

$$\int\!\!\int_{S_{j\pm\frac{1}{2},k}} \widetilde{w}^{n+1}_{j\pm\frac{1}{2},k}\, dxdy = |S_{j\pm\frac{1}{2},k}|\, \bar{w}^{n+1}_{j\pm\frac{1}{2},k} + \mathcal{O}(\Delta t^2), \tag{3.23}$$

$$\int\!\!\int_{S_{j,k\pm\frac{1}{2}}} \widetilde{w}^{n+1}_{j,k\pm\frac{1}{2}}\, dxdy = |S_{j,k\pm\frac{1}{2}}|\, \bar{w}^{n+1}_{j,k\pm\frac{1}{2}} + \mathcal{O}(\Delta t^2). \tag{3.24}$$

Also, the conservation property of the reconstruction $\widetilde{w}^{n+1}$ yields

$$\int\!\!\int_{D_{j,k}} \widetilde{w}^{n+1}_{j,k}(x,y)\, dxdy = |D_{j,k}|\bar{w}^{n+1}_{j,k}. \tag{3.25}$$

We now use (3.21) together with (3.22)–(3.25) and obtain

$$\begin{aligned}
\frac{d}{dt}\bar{u}_{j,k}(t) &= \lim_{\Delta t\to 0} \frac{\bar{u}^{n+1}_{j,k} - \bar{u}^{n}_{j,k}}{\Delta t} \\
&= \lim_{\Delta t\to 0} \left(\sum_{\pm} \frac{|S_{j,k\pm\frac{1}{2}}|}{\Delta t\Delta x\Delta y} \bar{w}^{n+1}_{j,k\pm\frac{1}{2}} + \sum_{\pm} \frac{|S_{j\pm\frac{1}{2},k}|}{\Delta t\Delta x\Delta y} \bar{w}^{n+1}_{j\pm\frac{1}{2},k} \right) \\
&\quad + \lim_{\Delta t\to 0} \frac{1}{\Delta t} \left[\frac{|D_{j,k}|}{\Delta x\Delta y} \bar{w}^{n+1}_{j,k} - \bar{u}^{n}_{j,k} \right].
\end{aligned} \tag{3.26}$$

For the first sum on the right-hand side (RHS), we apply Simpson's quadrature formula to the integrals over $D_{j,k\pm\frac{1}{2}}$ in the computation of $\bar{w}^{n+1}_{j,k\pm\frac{1}{2}}$. Since $|S_{j,k\pm\frac{1}{2}}| = \mp b^{\mp}_{j,k\pm\frac{1}{2}}\Delta t\Delta x + \mathcal{O}((\Delta t)^2)$, we arrive at (consult [25] for details)

$$\begin{aligned}
\lim_{\Delta t\to 0} \frac{|S_{j,k\pm\frac{1}{2}}|}{\Delta t\Delta x\Delta y} \bar{w}^{n+1}_{j,k\pm\frac{1}{2}} \approx &-\frac{b^{+}_{j,k\pm\frac{1}{2}} b^{-}_{j,k\pm\frac{1}{2}}}{6\Big(b^{+}_{j,k\pm\frac{1}{2}} - b^{-}_{j,k\pm\frac{1}{2}}\Big)\Delta y} \left[u^{\mathrm{SW(NW)}}_{j,k\pm 1} + 4u^{\mathrm{S(N)}}_{j,k\pm 1} + u^{\mathrm{SE(NE)}}_{j,k\pm 1} \right] \\
&+ \frac{\Big(b^{\mp}_{j,k\pm\frac{1}{2}}\Big)^2}{6\Big(b^{+}_{j,k\pm\frac{1}{2}} - b^{-}_{j,k\pm\frac{1}{2}}\Big)\Delta y} \left[u^{\mathrm{NW(SW)}}_{j,k} + 4u^{\mathrm{N(S)}}_{j,k} + u^{\mathrm{NE(SE)}}_{j,k} \right] \\
&+ \frac{b^{\mp}_{j,k\pm\frac{1}{2}}}{6\Big(b^{+}_{j,k\pm\frac{1}{2}} - b^{-}_{j,k\pm\frac{1}{2}}\Big)\Delta y} \Bigg[g(u^{\mathrm{SW(NW)}}_{j,k\pm 1}) - g(u^{\mathrm{NW(SW)}}_{j,k}) \\
&+ 4\Big(g(u^{\mathrm{S(N)}}_{j,k\pm 1}) - g(u^{\mathrm{N(S)}}_{j,k}) \Big) + g(u^{\mathrm{SE(NE)}}_{j,k\pm 1}) - g(u^{\mathrm{NE(SE)}}_{j,k}) \Bigg].
\end{aligned} \tag{3.27}$$

The second sum on the RHS of (3.26) is treated similarly, and we obtain

$$\begin{aligned}\lim_{\Delta t\to 0}\frac{|S_{j\pm\frac{1}{2},k}|}{\Delta t\Delta x\Delta y}\bar{w}^{n+1}_{j\pm\frac{1}{2},k} \approx & -\frac{a^+_{j\pm\frac{1}{2},k}a^-_{j\pm\frac{1}{2},k}}{6\Big(a^+_{j\pm\frac{1}{2},k}-a^-_{j\pm\frac{1}{2},k}\Big)\Delta x}\left[u^{\mathrm{NW(NE)}}_{j\pm1,k}+4u^{\mathrm{W(E)}}_{j\pm1,k}+u^{\mathrm{SW(SE)}}_{j\pm1,k}\right]\\ & +\frac{\Big(a^{\mp}_{j\pm\frac{1}{2},k}\Big)^2}{6\Big(a^+_{j\pm\frac{1}{2},k}-a^-_{j\pm\frac{1}{2},k}\Big)\Delta x}\left[u^{\mathrm{NE(NW)}}_{j,k}+4u^{\mathrm{E(W)}}_{j,k}+u^{\mathrm{SE(SW)}}_{j,k}\right]\\ & +\frac{a^{\mp}_{j\pm\frac{1}{2},k}}{6\Big(a^+_{j\pm\frac{1}{2},k}-a^-_{j\pm\frac{1}{2},k}\Big)\Delta x}\Bigg[f(u^{\mathrm{NW(NE)}}_{j\pm1,k})-f(u^{\mathrm{NE(NW)}}_{j,k})\\ & +4\Big(f(u^{\mathrm{W(E)}}_{j\pm1,k})-f(u^{\mathrm{E(W)}}_{j,k})\Big)f(u^{\mathrm{SW(SE)}}_{j\pm1,k})-f(u^{\mathrm{SE(SW)}}_{j,k})\Bigg].\end{aligned} \tag{3.28}$$

Finally, we consider the last term on the RHS of (3.26). Since the domain $D_{j,k}$ becomes *rectangular* as $\Delta t\to 0$, up to small corners of a negligible size $\mathcal{O}((\Delta t)^2)$, the integration of (3.17) over $D_{j,k}\times[t^n,t^n+\Delta t]$ and the application of Simpson's rule result in

$$\begin{aligned}&\lim_{\Delta t\to 0}\frac{1}{\Delta t}\left[\frac{|D_{j,k}|}{\Delta x\Delta y}\bar{w}^{n+1}_{j,k}-\bar{u}^n_{j,k}\right]\\ &\approx\frac{a^-_{j+\frac{1}{2},k}}{6\Delta x}\left[u^{\mathrm{NE}}_{j,k}+4u^{\mathrm{E}}_{j,k}+u^{\mathrm{SE}}_{j,k}\right]-\frac{a^+_{j-\frac{1}{2},k}}{6\Delta x}\left[u^{\mathrm{NW}}_{j,k}+4u^{\mathrm{W}}_{j,k}+u^{\mathrm{SW}}_{j,k}\right]\\ &+\frac{b^-_{j,k+\frac{1}{2}}}{6\Delta y}\left[u^{\mathrm{NW}}_{j,k}+4u^{\mathrm{N}}_{j,k}+u^{\mathrm{NE}}_{j,k}\right]-\frac{b^+_{j,k-\frac{1}{2}}}{6\Delta y}\left[u^{\mathrm{SW}}_{j,k}+4u^{\mathrm{S}}_{j,k}+u^{\mathrm{SE}}_{j,k}\right]\\ &-\frac{1}{6\Delta x}\left[f(u^{\mathrm{NE}}_{j,k})-f(u^{\mathrm{NW}}_{j,k})+4\Big(f(u^{\mathrm{E}}_{j,k})-f(u^{\mathrm{W}}_{j,k})\Big)+f(u^{\mathrm{SE}}_{j,k})-f(u^{\mathrm{SW}}_{j,k})\right]\\ &-\frac{1}{6\Delta y}\left[g(u^{\mathrm{NW}}_{j,k})-g(u^{\mathrm{SW}}_{j,k})+4\Big(g(u^{\mathrm{N}}_{j,k})-g(u^{\mathrm{S}}_{j,k})\Big)+g(u^{\mathrm{NE}}_{j,k})-g(u^{\mathrm{SE}}_{j,k})\right].\end{aligned} \tag{3.29}$$

Our 2-D semidiscrete central-upwind scheme is obtained by plugging (3.27)–(3.29) into (3.26). It can be written in the following conservative form:

$$\frac{d}{dt}\bar{u}_{j,k}(t)=-\frac{H^x_{j+\frac{1}{2},k}(t)-H^x_{j-\frac{1}{2},k}(t)}{\Delta x}-\frac{H^y_{j,k+\frac{1}{2}}(t)-H^y_{j,k-\frac{1}{2}}(t)}{\Delta y}, \tag{3.30}$$

where the numerical fluxes are

$$\begin{aligned}H^x_{j+\frac{1}{2},k}:=&\frac{a^+_{j+\frac{1}{2},k}}{6\Big(a^+_{j+\frac{1}{2},k}-a^-_{j+\frac{1}{2},k}\Big)}\left[f(u^{\mathrm{NE}}_{j,k})+4f(u^{\mathrm{E}}_{j,k})+f(u^{\mathrm{SE}}_{j,k})\right]\\ &-\frac{a^-_{j+\frac{1}{2},k}}{6\Big(a^+_{j+\frac{1}{2},k}-a^-_{j+\frac{1}{2},k}\Big)}\left[f(u^{\mathrm{NW}}_{j+1,k})+4f(u^{\mathrm{W}}_{j+1,k})+f(u^{\mathrm{SW}}_{j+1,k})\right]\\ &+\frac{a^+_{j+\frac{1}{2},k}a^-_{j+\frac{1}{2},k}}{6\Big(a^+_{j+\frac{1}{2},k}-a^-_{j+\frac{1}{2},k}\Big)}\left[u^{\mathrm{NW}}_{j+1,k}-u^{\mathrm{NE}}_{j,k}+4(u^{\mathrm{W}}_{j+1,k}-u^{\mathrm{E}}_{j,k})+u^{\mathrm{SW}}_{j+1,k}-u^{\mathrm{SE}}_{j,k}\right]\end{aligned} \tag{3.31}$$

and

$$\begin{aligned}H^y_{j,k+\frac{1}{2}} &:= \frac{b^+_{j,k+\frac{1}{2}}}{6\Big(b^+_{j,k+\frac{1}{2}}-b^-_{j,k+\frac{1}{2}}\Big)}\Big[g(u^{\rm NW}_{j,k})+4g(u^{\rm N}_{j,k})+g(u^{\rm NE}_{j,k})\Big]\\ &\quad-\frac{b^-_{j,k+\frac{1}{2}}}{6\Big(b^+_{j,k+\frac{1}{2}}-b^-_{j,k+\frac{1}{2}}\Big)}\Big[g(u^{\rm SW}_{j,k+1})+4g(u^{\rm S}_{j,k+1})+g(u^{\rm SE}_{j,k+1})\Big]\\ &\quad+\frac{b^+_{j,k+\frac{1}{2}}b^-_{j,k+\frac{1}{2}}}{6\Big(b^+_{j,k+\frac{1}{2}}-b^-_{j,k+\frac{1}{2}}\Big)}\Big[u^{\rm SW}_{j,k+1}-u^{\rm NW}_{j,k}+4(u^{\rm S}_{j,k+1}-u^{\rm N}_{j,k})+u^{\rm SE}_{j,k+1}-u^{\rm NE}_{j,k}\Big].\end{aligned} \tag{3.32}$$

Here, the one-sided local speeds $a^\pm_{j+\frac{1}{2},k}, b^\pm_{j,k+\frac{1}{2}}$ are defined in (3.19), and the values of the u's are computed in (3.18), using the piecewise quadratic reconstruction $\{p_{j,k}\}$ at time t. In our numerical examples, we have implemented the reconstruction introduced in [25].

Remarks.

1. Our 2-D semidiscrete central-upwind scheme (3.30)–(3.32) is a Godunov-type *central* scheme; therefore it can be applied *componentwise* and does not require Riemann solvers. As in [25], this scheme is constructed as a genuinely multi-D scheme. Moreover, if one sets

$$\begin{aligned}a^+_{j+\frac{1}{2},k} &:= -a^-_{j+\frac{1}{2},k} := \max\Big\{a^+_{j+\frac{1}{2},k}, -a^-_{j+\frac{1}{2},k}\Big\},\\ b^+_{j,k+\frac{1}{2}} &:= -b^-_{j,k+\frac{1}{2}} := \max\Big\{b^+_{j,k+\frac{1}{2}}, -b^-_{j,k+\frac{1}{2}}\Big\},\end{aligned}$$

the scheme (3.30)–(3.32) reduces to the one in [25].

2. As in the 1-D case, our 2-D scheme (3.30)–(3.32) has an upwind nature. To illustrate this, let us consider the simplest linear scalar advection equation, $u_t + au_x + bu_y = 0$, with positive a and b. In this setting, the first-order version of the scheme (3.30)–(3.32) becomes a standard first-order upwind scheme

$$\frac{d}{dt}u_{j,k}(t) = -a\frac{u_{j,k}-u_{j-1,k}}{\Delta x} - b\frac{u_{j,k}-u_{j,k-1}}{\Delta y}.$$

3. A second-order version of the 2-D scheme (3.30)–(3.32) can be obtained if one uses a second-order piecewise polynomial reconstruction (say, the minmod reconstruction) and a lower-order midpoint quadrature instead of the fourth-order Simpson's rule. This results in the scheme

$$\frac{d}{dt}u_{j,k}(t) = -\frac{H^x_{j+\frac{1}{2},k}(t)-H^x_{j-\frac{1}{2},k}(t)}{\Delta x} - \frac{H^y_{j,k+\frac{1}{2}}(t)-H^y_{j,k-\frac{1}{2}}(t)}{\Delta y}, \tag{3.33}$$

with the corresponding numerical fluxes

$$H^x_{j+\frac{1}{2},k} := \frac{a^+_{j+\frac{1}{2},k}f(u^{\rm E}_{j,k})-a^-_{j+\frac{1}{2},k}f(u^{\rm W}_{j+1,k})}{a^+_{j+\frac{1}{2},k}-a^-_{j+\frac{1}{2},k}} + \frac{a^+_{j+\frac{1}{2},k}a^-_{j+\frac{1}{2},k}}{a^+_{j+\frac{1}{2},k}-a^-_{j+\frac{1}{2},k}}\Big[u^{\rm W}_{j+1,k}-u^{\rm E}_{j,k}\Big] \tag{3.34}$$

and

$$H^y_{j,k+\frac{1}{2}} := \frac{b^+_{j,k+\frac{1}{2}} g(u^{\rm N}_{j,k}) - b^-_{j,k+\frac{1}{2}} g(u^{\rm S}_{j,k+1})}{b^+_{j,k+\frac{1}{2}} - b^-_{j,k+\frac{1}{2}}} + \frac{b^+_{j,k+\frac{1}{2}} b^-_{j,k+\frac{1}{2}}}{b^+_{j,k+\frac{1}{2}} - b^-_{j,k+\frac{1}{2}}} \Big[u^{\rm S}_{j,k+1} - u^{\rm N}_{j,k}\Big]. \tag{3.35}$$

The same scheme can also be derived by applying the 1-D numerical flux (3.15) in both x- and y-directions (this is the so-called dimension-by-dimension approach, used in [26]).

4. The scheme (3.30)–(3.32) can be generalized and applied to convection-diffusion equations (for details see [26, 25, 24]). Also, it can be rather easily extended to the multi-D case, $d \geq 3$.

3.2.1. Maximum principle for the second-order central-upwind scheme. We consider the 2-D second-order central-upwind scheme (3.33)–(3.35), together with the minmod reconstruction (3.16). We solve the time-dependent ODE system (3.33), using a TVD Runge–Kutta method. Under an appropriate CFL condition, the resulting fully discrete scheme, applied to a scalar conservation law, satisfies the maximum principle; see the following theorem.

THEOREM 3.1 (maximum principle). *Consider the scalar conservation laws* (3.17)*. Then the second-order scheme* (3.33)*–*(3.35)*, with the minmod reconstruction* (3.16)*, coupled with a TVD Runge–Kutta method* [47, 45] *satisfies the maximum principle*

$$\min_{j,k}\{u^n_{j,k}\} \leq \min_{j,k}\{u^{n+1}_{j,k}\} \leq \max_{j,k}\{u^{n+1}_{j,k}\} \leq \max_{j,k}\{u^n_{j,k}\} \tag{3.36}$$

under the CFL condition

$$\max\left(\frac{\Delta t^n}{\Delta x} \max_u |f'(u)|, \frac{\Delta t^n}{\Delta y} \max_u |g'(u)|\right) \leq \frac{1}{8}, \tag{3.37}$$

where Δt^n is the variable time step of the Runge–Kutta method.

We omit the proof since it is similar to the proof of Theorem 5.1 in [26].

3.3. Semidiscrete central-upwind scheme for Hamilton–Jacobi equations. In this section, we propose a new Godunov-type central-upwind scheme for the 1-D and 2-D Hamilton–Jacobi equations (1.2). We follow the approach in [27], but this time we utilize more precise information about the *one-sided* local speeds of propagation.

We begin with the 1-D case and start at time level $t = t^n$ with the continuous piecewise polynomial interpolant $\widetilde{\varphi}(x, t^n)$ and estimate the maximal one-sided local speeds, a^+_j and a^-_j. For example, in the convex case, they are equal to

$$a^+_j := \max\{H'(\varphi^+_x), H'(\varphi^-_x), 0\}, \qquad a^-_j := \min\{H'(\varphi^+_x), H'(\varphi^-_x), 0\}, \tag{3.38}$$

where we use the notation $\varphi^\pm_x := \widetilde{\varphi}_x(x_j \pm 0, t^n)$. To construct the second-order scheme one should use the continuous piecewise quadratic polynomial (2.5), and in this case,

$$\varphi^\pm_x = \frac{(\Delta\varphi)^n_{j\pm\frac{1}{2}}}{\Delta x} \mp \frac{(\Delta\varphi)'_{j\pm\frac{1}{2}}}{2\Delta x}. \tag{3.39}$$

Note that under an appropriate CFL-condition, the solution of the Hamilton–Jacobi equation (1.2) with the piecewise polynomial initial data $\widetilde{\varphi}(x, t^n)$ is smooth

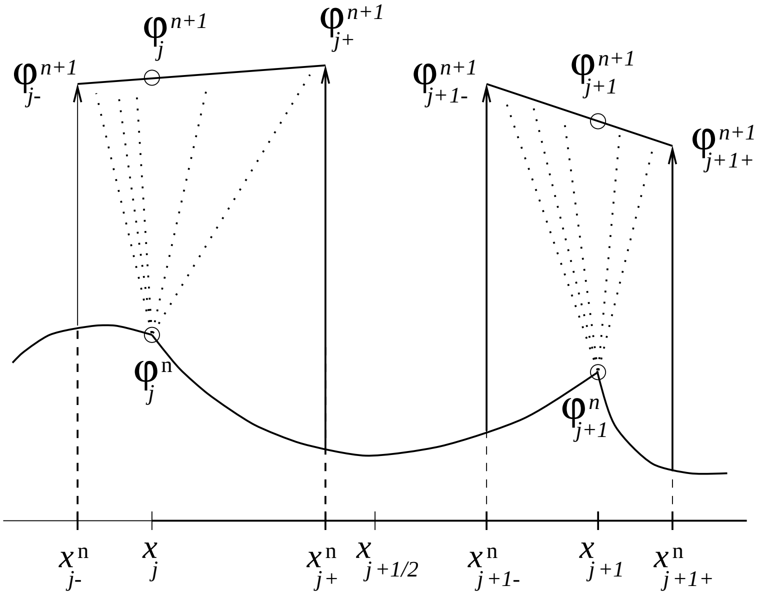


FIG. 3.3. *Central-upwind differencing*—1-D.

along the line segments $(x^n_{j\pm}, t)$, $t^n \le t < t^{n+1}$, $x^n_{j\pm} := x_j + a^{\pm}_j \Delta t$; see Figure 3.3. Therefore, one can use the Taylor expansion to compute the intermediate point values at the next time level:

$$\varphi^{n+1}_{j\pm} = \widetilde{\varphi}(x_{j\pm}, t^n) - \Delta t \cdot H(\widetilde{\varphi}_x(x^n_{j\pm}, t^n)) + \mathcal{O}(\Delta t)^2. \tag{3.40}$$

We complete the construction of our fully discrete central-upwind scheme by projecting the intermediate point values back onto the original grid. Since the distance between x^n_{j+} and x^n_{j-} is proportional to Δt, it suffices to compute the weighted averages of φ^{n+1}_{j+} and φ^{n+1}_{j-}, that is,

$$\varphi^{n+1}_j = \frac{a^+_j}{a^+_j - a^-_j} \varphi^{n+1}_{j-} - \frac{a^-_j}{a^+_j - a^-_j} \varphi^{n+1}_{j+} + \mathcal{O}(\Delta t)^2. \tag{3.41}$$

Finally, we substitute (3.40) in (3.41), and arrive at the fully discrete scheme

$$\begin{aligned} \varphi^{n+1}_j = {} & \frac{a^+_j}{a^+_j - a^-_j} \Big(\widetilde{\varphi}(x_{j-}, t^n) - \Delta t H(\widetilde{\varphi}_x(x^n_{j-}, t^n)) \Big) \\ & - \frac{a^-_j}{a^+_j - a^-_j} \Big(\widetilde{\varphi}(x_{j+}, t^n) - \Delta t H(\widetilde{\varphi}_x(x^n_{j+}, t^n)) \Big), \end{aligned} \tag{3.42}$$

which is high-order in space (depending on the order of the piecewise polynomial reconstruction) and only first-order in time.

A semidiscrete version of the scheme (3.42), coupled with a high-order ODE solver, will allow us to achieve high accuracy both in space and time. To derive such a scheme, we first use the Taylor expansions, $\widetilde{\varphi}(x_{j\pm}, t^n) = \widetilde{\varphi}(x_j, t^n) + \Delta t a_j^{\pm}\widetilde{\varphi}_x(x_j \pm 0, t^n) + \mathcal{O}(\Delta t)^2$, and rewrite the fully discrete scheme (3.42) as

$$\varphi_j^{n+1} = \varphi_j^n - \Delta t \frac{a_j^+ a_j^-}{a_j^+ - a_j^-}\Big[\widetilde{\varphi}_x(x_j + 0, t^n) - \widetilde{\varphi}_x(x_j - 0, t^n)\Big] + \frac{\Delta t}{a_j^+ - a_j^-}\Big[a_j^- H(\widetilde{\varphi}_x(x_{j+}^n, t^n)) - a_j^+ H(\widetilde{\varphi}_x(x_{j-}^n, t^n))\Big] + \mathcal{O}(\Delta t)^2. \tag{3.43}$$

We now let $\Delta t \to 0$, and end up with the following semidiscrete central-upwind scheme:

$$\frac{d}{dt}\varphi_j(t) = \frac{1}{a_j^+ - a_j^-}\Big[a_j^- H(\varphi_x^+) - a_j^+ H(\varphi_x^-)\Big] - \frac{a_j^+ a_j^-}{a_j^+ - a_j^-}\Big(\varphi_x^+ - \varphi_x^-\Big). \tag{3.44}$$

Here, $a_j^{\pm}$ are given by (3.38), and $\varphi_x^{\pm}$ are the right and the left derivatives at the point $x = x_j$ of the reconstruction $\widetilde{\varphi}(\cdot, t)$ at time t.

We continue with the construction of a multi-D extension of the scheme (3.44). Without loss of generality, we consider the 2-D Hamilton–Jacobi equation,

$$\varphi_t + H(\varphi_x, \varphi_y) = 0. \tag{3.45}$$

Assume that at time $t = t^n$ the discrete approximation to the point values of its solution, $\{\varphi_{j,k}^n \approx \varphi(x_j, y_k, t^n)\}$, has already been computed. We then construct the 2-D continuous piecewise polynomial interpolant $\widetilde{\varphi}(x, y, t^n)$. Such a reconstruction is defined over the four triangles (NW, NE, SW, and SE) around each grid-point (x_j, y_k) (see Figure 3.4). We refer the reader to [27] for an example of a nonoscillatory second-order piecewise quadratic interpolant.

We now continue with the construction of our semidiscrete central-upwind scheme. As in the 1-D case, we use the maximal values of the one-sided local speeds of propagation in the x- and y-directions. These values at the grid-point (x_j, y_k) are given by

$$\begin{aligned} a_{j,k}^+ &:= \max_{C_{j,k}}\Big\{H_u(\widetilde{\varphi}_x(x,y), \widetilde{\varphi}_y(x,y))\Big\}_+, \quad & a_{j,k}^- &:= \min_{C_{j,k}}\Big\{H_u(\widetilde{\varphi}_x(x,y), \widetilde{\varphi}_y(x,y))\Big\}_-, \\ b_{j,k}^+ &:= \max_{C_{j,k}}\Big\{H_v(\widetilde{\varphi}_x(x,y), \widetilde{\varphi}_y(x,y))\Big\}_+, \quad & b_{j,k}^- &:= \min_{C_{j,k}}\Big\{H_v(\widetilde{\varphi}_x(x,y), \widetilde{\varphi}_y(x,y))\Big\}_-, \end{aligned} \tag{3.46}$$

where $C_{j,k} := [x_{j-\frac{1}{2}}, x_{j+\frac{1}{2}}] \times [y_{k-\frac{1}{2}}, y_{k+\frac{1}{2}}]$ and $(\,\cdot\,)_+ := \max(\cdot, 0)$, $(\,\cdot\,)_- := \min(\cdot, 0)$.

To compute the solution at the next time level $t = t^{n+1}$, we use the intermediate values $\{\varphi_{j\pm,k\pm}^{n+1}\}$, obtained by the Taylor expansion about the points $(x_{j\pm}^n := x_j + a_{j,k}^{\pm}\Delta t,\ y_{k\pm}^n := y_k + b_{j,k}^{\pm}\Delta t)$,

$$\varphi_{j\pm,k\pm}^{n+1} = \widetilde{\varphi}(x_{j\pm}^n, y_{k\pm}^n, t^n) - \Delta t \cdot H(\widetilde{\varphi}_x(x_{j\pm}^n, y_{k\pm}^n, t^n), \widetilde{\varphi}_y(x_{j\pm}^n, y_{k\pm}^n, t^n)) + \mathcal{O}(\Delta t)^2. \tag{3.47}$$

Expansion (3.47) is valid, since due to the finite speed of propagation, the solution of (3.45) with the initial data $\widetilde{\varphi}(x, y, t^n)$ is smooth around $(x_{j\pm}^n, y_{k\pm}^n)$; see Figure 3.4.

Next, the computed intermediate values (3.47) are projected back onto the original grid. This can be done using the weighted average of the values $\varphi_{j\pm,k\pm}^{n+1}$, since the

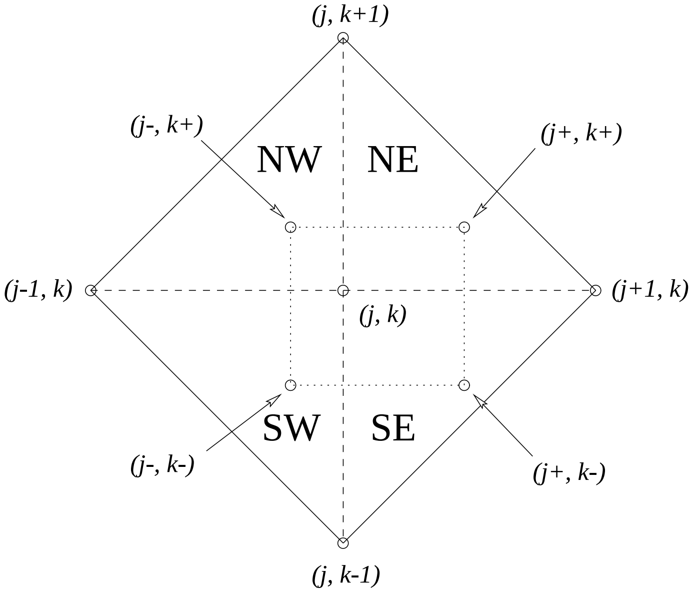


FIG. 3.4. *Central-upwind differencing*—2-D.

distance between the points $(x_{j\pm}, y_{k\pm})$ is proportional to $\mathcal{O}(\Delta t)$. The resulting fully discrete central-upwind scheme is

$$\begin{aligned}
&\varphi_{j,k}^{n+1}\\
&= \frac{a_{j,k}^- b_{j,k}^-}{(a_{j,k}^+ - a_{j,k}^-)(b_{j,k}^+ - b_{j,k}^-)}\Big(\widetilde{\varphi}(x_{j+}^n, y_{k+}^n, t^n) - \Delta t \cdot H(\widetilde{\varphi}_x(x_{j+}^n, y_{k+}^n, t^n), \widetilde{\varphi}_y(x_{j+}^n, y_{k+}^n, t^n))\Big)\\
&- \frac{a_{j,k}^- b_{j,k}^+}{(a_{j,k}^+ - a_{j,k}^-)(b_{j,k}^+ - b_{j,k}^-)}\Big(\widetilde{\varphi}(x_{j+}^n, y_{k-}^n, t^n) - \Delta t \cdot H(\widetilde{\varphi}_x(x_{j+}^n, y_{k-}^n, t^n), \widetilde{\varphi}_y(x_{j+}^n, y_{k-}^n, t^n))\Big)\\
&- \frac{a_{j,k}^+ b_{j,k}^-}{(a_{j,k}^+ - a_{j,k}^-)(b_{j,k}^+ - b_{j,k}^-)}\Big(\widetilde{\varphi}(x_{j-}^n, y_{k+}^n, t^n) - \Delta t \cdot H(\widetilde{\varphi}_x(x_{j-}^n, y_{k+}^n, t^n), \widetilde{\varphi}_y(x_{j-}^n, y_{k+}^n, t^n))\Big)\\
&+ \frac{a_{j,k}^+ b_{j,k}^+}{(a_{j,k}^+ - a_{j,k}^-)(b_{j,k}^+ - b_{j,k}^-)}\Big(\widetilde{\varphi}(x_{j-}^n, y_{k-}^n, t^n) - \Delta t \cdot H(\widetilde{\varphi}_x(x_{j-}^n, y_{k-}^n, t^n), \widetilde{\varphi}_y(x_{j-}^n, y_{k-}^n, t^n))\Big).
\end{aligned}$$

(3.48)

As in the 1-D case, the scheme (3.48) is only first-order in time. This disadvantage can be eliminated by passing to the semidiscrete limit in (3.48) as $\Delta t \to 0$. To this end, we first compute the values of $\widetilde{\varphi}(x_\pm^n, y_\pm^n, t^n)$ by the Taylor expansions,

$$\begin{aligned}
\widetilde{\varphi}(x_{j\pm}, y_{k+}, t^n) &= \widetilde{\varphi}(x_j, y_k, t^n) + \Delta t a_{j,k}^\pm \widetilde{\varphi}_x(x_j \pm 0, y_k, t^n)\\
&\quad + \Delta t b_{j,k}^+ \widetilde{\varphi}_y(x_j, y_k + 0, t^n) + \mathcal{O}(\Delta t)^2,\\
\widetilde{\varphi}(x_{j\pm}, y_{k-}, t^n) &= \widetilde{\varphi}(x_j, y_k, t^n) + \Delta t a_{j,k}^\pm \widetilde{\varphi}_x(x_j \pm 0, y_k, t^n)\\
&\quad + \Delta t b_{j,k}^- \widetilde{\varphi}_y(x_j, y_k - 0, t^n) + \mathcal{O}(\Delta t)^2.
\end{aligned}$$

We then plug these values in (3.48), and after passing to the limit as $\Delta t \to 0$, we obtain the 2-*D semidiscrete central-upwind scheme*,

$$
\begin{aligned}
&\frac{d}{dt}\varphi_{j,k}(t)\\
&= -\ \frac{a^-_{j,k}b^-_{j,k}H(\varphi^+_x,\varphi^+_y) - a^-_{j,k}b^+_{j,k}H(\varphi^+_x,\varphi^-_y) - a^+_{j,k}b^-_{j,k}H(\varphi^-_x,\varphi^+_y) + a^+_{j,k}b^+_{j,k}H(\varphi^-_x,\varphi^-_y)}{(a^+_{j,k}-a^-_{j,k})(b^+_{j,k}-b^-_{j,k})}\\
&\quad - \frac{a^+_{j,k}a^-_{j,k}}{a^+_{j,k}-a^-_{j,k}}\Big(\varphi^+_x-\varphi^-_x\Big) - \frac{b^+_{j,k}b^-_{j,k}}{b^+_{j,k}-b^-_{j,k}}\Big(\varphi^+_y-\varphi^-_y\Big).
\end{aligned}
\tag{3.49}
$$

Here, $\varphi^\pm_x := \widetilde{\varphi}_x(x_j \pm 0, y_k, t)$ and $\varphi^\pm_y := \widetilde{\varphi}_y(x_j, y_k \pm 0, t)$ are the right and the left derivatives in the x- and y-direction, respectively. The one-sided local speeds in (3.49) are given by (3.46). In practice, these speeds can be estimated in a simpler way. For instance, in the numerical examples, we have used

$$
\begin{aligned}
a^+_{j,k} &:= \max_{\pm}\Big\{H_u(\varphi^\pm_x,\varphi^\pm_y)\Big\}_+, \quad a^-_{j,k} := \min_{\pm}\Big\{H_u(\varphi^\pm_x,\varphi^\pm_y)\Big\}_-,\\
b^+_{j,k} &:= \max_{\pm}\Big\{H_v(\varphi^\pm_x,\varphi^\pm_y)\Big\}_+, \quad b^-_{j,k} := \min_{\pm}\Big\{H_v(\varphi^\pm_x,\varphi^\pm_y)\Big\}_-.
\end{aligned}
\tag{3.50}
$$

Finally, to obtain the same second-order accuracy in time, our semidiscrete central-upwind scheme (3.49)–(3.50) should be complemented with at least a second-order ODE solver for time discretization.

4. Numerical examples.

In this section, we implement our scheme for conservation laws and Hamilton–Jacobi equations and perform several numerical experiments. We test the accuracy of the scheme on problems with smooth solutions and solve various equations which admit nonsmooth solutions. Among them are the Euler equations of gas dynamics, the incompressible Euler equations, and others. The numerical results show that our scheme gives sharper resolution and reduces some of the side effects of the schemes from [27, 25].

The high-order semidiscrete methods, presented in this paper, require a time discretization of the corresponding order. In the numerical examples, shown below, we have used the third-order TVD Runge–Kutta method, proposed in [45, 47], and the second-order modified Euler method. Our choice is based on the stability properties of these methods, each time step of which can be viewed as a convex combination of small forward Euler steps.

In all the numerical experiments below, the CFL number is equal to 0.475, and the value of θ in the generalized minmod limiter is 2.

4.1. 1-D problems.

Example 1. Burgers' equation. We consider the initial boundary value problem (IBVP) for the inviscid Burgers' equation

$$
u_t + \left(\frac{u^2}{2}\right)_x = 0, \qquad u(x,0) = 0.5 + \sin x, \quad x \in [0, 2\pi], \tag{4.1}
$$

with periodic boundary conditions. It is known that the unique entropy solution of (4.1) develops a shock discontinuity at time $t = 1$. The solution at the preshock time $T = 0.5$ is smooth, and this allows us to test the accuracy of the 1-D third-order central-upwind scheme (3.14)–(3.15). We couple it with the basic piecewise quadratic reconstruction (for details, see [37, 39, 25]), and compute the solution using N grid points, $N = 40, 80, \dots, 1280$.

The L^∞- and L^1-errors are shown in Table 4.1, and they clearly demonstrate a third-order experimental convergence rate.

Table 4.1
Accuracy test for the Burgers' equation (4.1), $T = 0.5$.

N	L^∞-error	rate	L^1-error	rate
40	1.456e-03	–	1.241e-03	–
80	2.177e-04	2.74	1.683e-04	2.88
160	2.893e-05	2.91	2.187e-05	2.94
320	3.689e-06	2.97	2.794e-06	2.97
640	4.559e-07	3.02	3.484e-07	3.00
1280	5.720e-08	2.99	4.376e-08	2.99

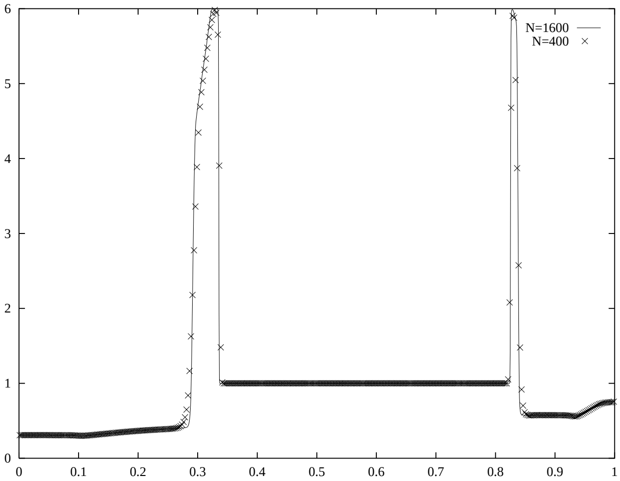


Fig. 4.1. *Problem* (4.2)–(4.3), *density at* $T = 0.01$.

Example 2. 1-D Euler equations of gas dynamics. We solve the 1-D Euler system

$$\frac{\partial}{\partial t}\begin{bmatrix}\rho\\ m\\ E\end{bmatrix}+\frac{\partial}{\partial x}\begin{bmatrix}m\\ \rho u^2+p\\ u(E+p)\end{bmatrix}=0, \qquad p=(\gamma-1)\cdot\Big(E-\frac{\rho}{2}u^2\Big), \tag{4.2}$$

with the initial data

$$\vec{u}(x,0)=\begin{cases}\vec{u}_L=(1,0,2500)^T, & 0\le x<0.1,\\ \vec{u}_M=(1,0,0.025)^T, & 0.1\le x<0.9,\\ \vec{u}_R=(1,0,250)^T, & 0.9\le x<1,\end{cases} \tag{4.3}$$

and solid wall boundary conditions, applied to both ends. This problem, proposed in [51], models the interaction of blast waves. Here, ρ, u, $m = \rho u$, p, and E are the density, velocity, momentum, pressure, and the total energy, respectively; $\gamma = 1.4$.

The solution is computed with our scheme (3.14)–(3.15) and the 1-D reconstruction in [25]. We use $N = 400$ grid points and plot the density, the velocity, and the pressure together with a reference solution, obtained by the same method with $N = 1600$.

Figures 4.1, 4.2, and 4.3 show the density, the velocity, and the pressure at time $T = .01$. Note, that for $N = 400$, the second density spike has a height of ~ 5.9, which is closer to the actual value of the solution. This result is better than the heights of

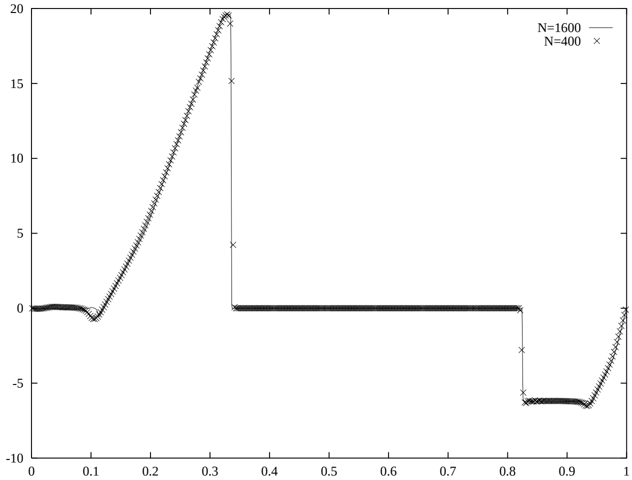


FIG. 4.2. *Problem* (4.2)–(4.3)*, velocity at* $T = 0.01$.

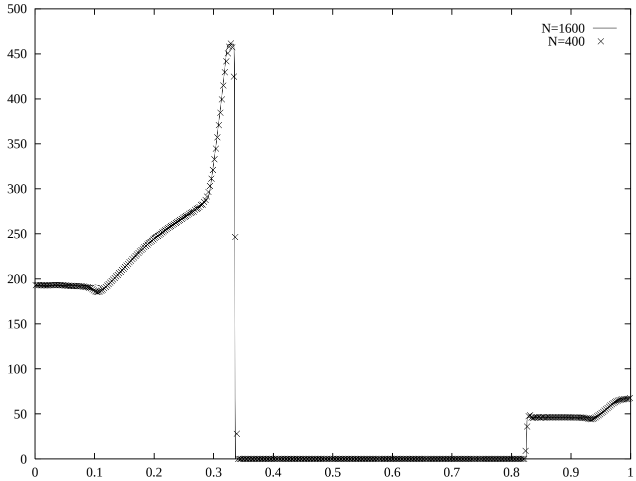


FIG. 4.3. *Problem* (4.2)–(4.3)*, pressure at* $T = 0.01$.

~ 5.75, ~ 5.2, and ~ 3.7, obtained by the third-order central schemes in [25, 39], and the second-order Nessyahu–Tadmor scheme in [40], respectively. This illustrates the higher resolution and smaller numerical dissipation of the central-upwind scheme.

The computations at times $T = 0.03$ and $T = 0.038$ are comparable to the results from [25], and are shown in Figures 4.4–4.9.

Example 3. 1-D Hamilton–Jacobi equation. In this example, we apply the second-order central-upwind scheme (2.7), (3.38)–(3.39), (3.44) to the Riemann problem for a 1-D Hamilton–Jacobi equation with a nonconvex Hamiltonian,

$$\varphi_t + \frac{1}{4}(\varphi_x^2 - 1)(\varphi_x^2 - 4) = 0, \qquad \varphi(x,0) = -2|x|. \tag{4.4}$$

The numerical solution, computed for different numbers of grid points is plotted in Figure 4.10. One can observe a very fast convergence of the approximate solutions

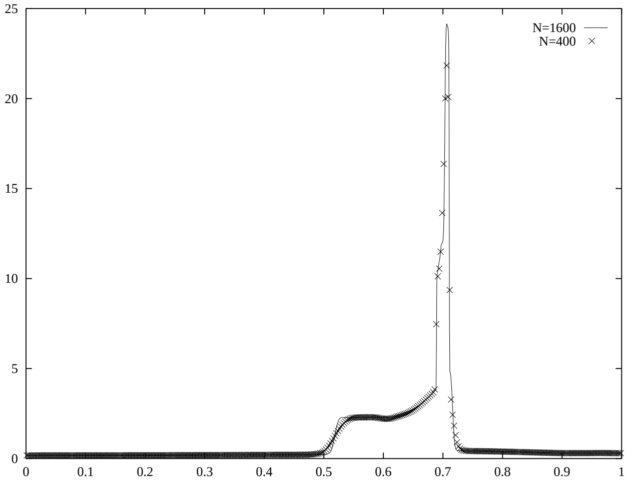


FIG. 4.4. *Problem* (4.2)–(4.3), *density at* $T = 0.03$.

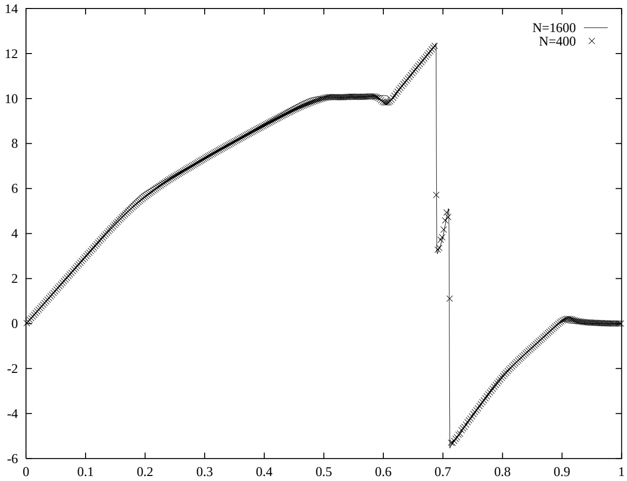


FIG. 4.5. *Problem* (4.2)–(4.3), *velocity at* $T = 0.03$.

toward the exact (viscosity) solution of (4.4) as the mesh is refined. The exact solution is obtained by solving the Riemann problem for the corresponding conservation law.

4.2. 2-D problems.

Example 4. 2-D Euler equations of gas dynamics. We solve the 2-D compressible Euler equations

$$\frac{\partial}{\partial t}\begin{bmatrix}\rho\\ \rho u\\ \rho v\\ E\end{bmatrix}+\frac{\partial}{\partial x}\begin{bmatrix}\rho u\\ \rho u^2+p\\ \rho uv\\ u(E+p)\end{bmatrix}+\frac{\partial}{\partial y}\begin{bmatrix}\rho v\\ \rho uv\\ \rho v^2+p\\ v(E+p)\end{bmatrix}=0,\ \ p=(\gamma-1)\cdot\Big[E-\frac{\rho}{2}(u^2+v^2)\Big]$$
(4.5)

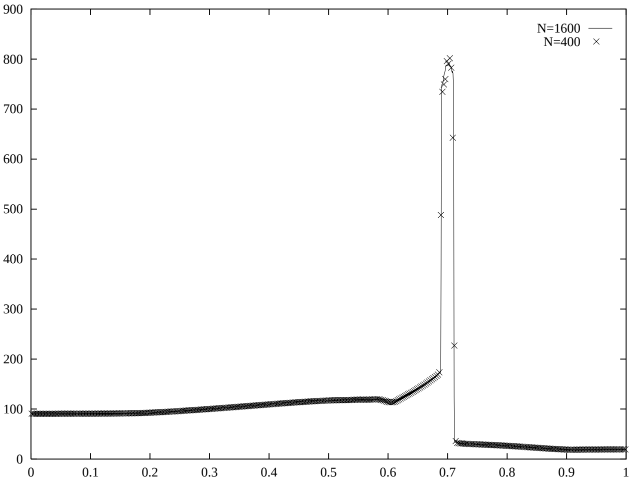


FIG. 4.6. *Problem* (4.2)–(4.3)*, pressure at* $T = 0.03$.

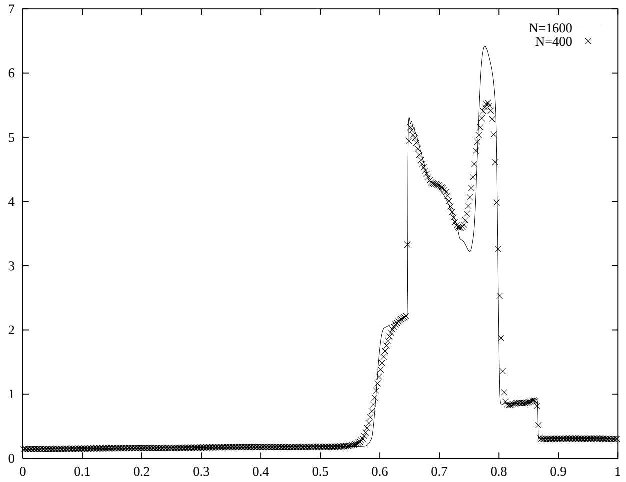


FIG. 4.7. *Problem* (4.2)–(4.3)*, density at* $T = 0.038$.

for an ideal gas, $\gamma = 1.4$, in the domain $[0, 2] \times [0, 0.5] \cup [0, 1] \times [0.5, 1]$, with the initial data corresponding to a vertical left-moving Mach 1.65 shock, positioned at $x = 1.375$. The initial shock propagates and then diffracts around a solid corner. Here ρ, u, v, p , and E are the density, the x- and y-velocities, the pressure, and the total energy, respectively.

We compute the solution at time $T = 0.5$, using the scheme (3.30)–(3.32), coupled with the reconstruction in [25]. The contour plots of the density for 128×64, 256×128, and 512×256 grid points are given in Figures 4.11, 4.12, and 4.13, respectively.

Note that as in [25], the results are obtained without using characteristic decomposition, dimensional splitting, or evolution of nonconservative quantities.

Example 5. 2-D Hamilton–Jacobi equation. We consider the initial value problem for the 2-D eikonal equation of geometric optics, which is a Hamilton–Jacobi

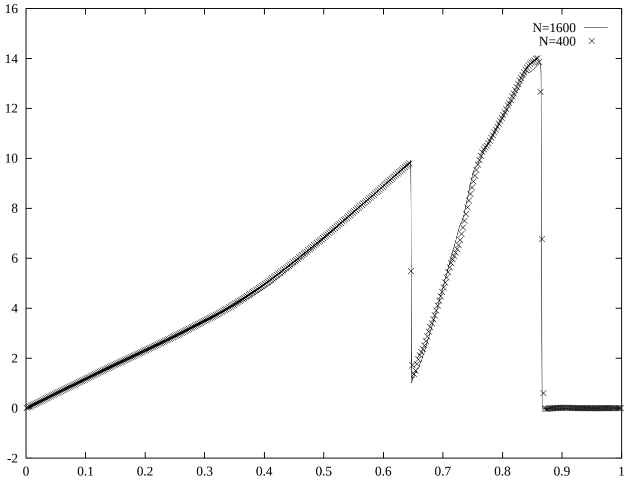


FIG. 4.8. *Problem* (4.2)–(4.3)*, velocity at* $T = 0.038$*.*

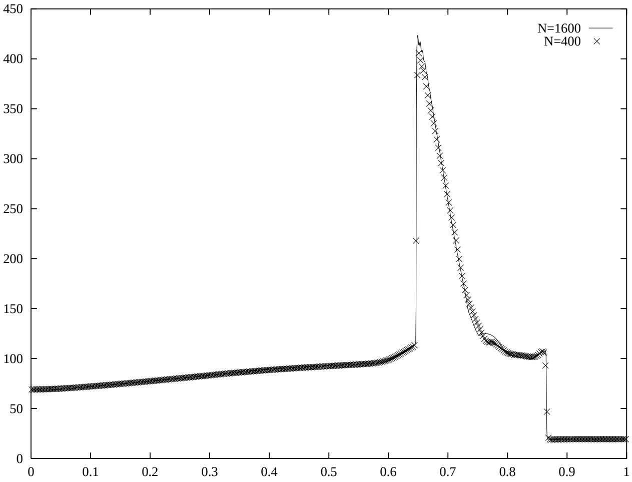


FIG. 4.9. *Problem* (4.2)–(4.3)*, pressure at* $T = 0.038$*.*

equation with a convex Hamiltonian,

$$\varphi_t+\sqrt{\varphi_x^2+\varphi_y^2+1}=0, \quad \varphi(x,y,0)=\frac{1}{4}(\cos(2\pi x)-1)(\cos(2\pi y)-1)-1, \ (x,y)\in[0,1]^2. \tag{4.6}$$

The numerical solution of (4.6) at time $T = 0.6$ (after formation of the singularity) has been computed by the 2-D second-order central-upwind scheme (3.49)–(3.50). The nonoscillatory nature of the computed solution and nearly perfect resolution of the singularity can be clearly seen in Figures 4.14–4.15.

4.3. 2-D incompressible Euler and Navier–Stokes equations. Here, we consider the incompressible Euler ($\nu = 0$) and Navier–Stokes ($\nu > 0$) equations

$$\vec{u}_t + (\vec{u}\cdot\nabla)\vec{u} + \nabla p = \nu\Delta\vec{u}, \tag{4.7}$$

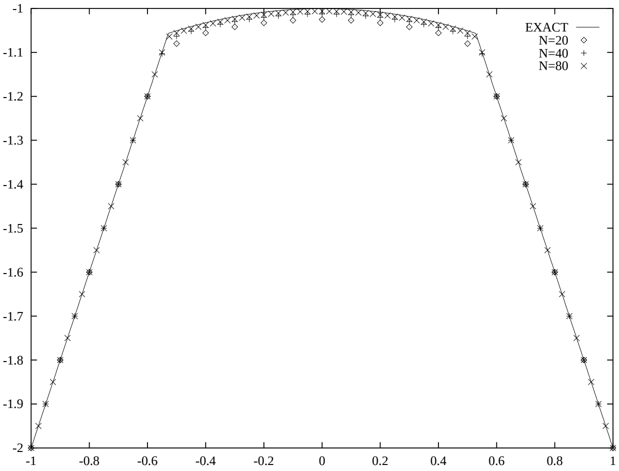


FIG. 4.10. *Problem* (4.4), *solution at* $T = 1$.

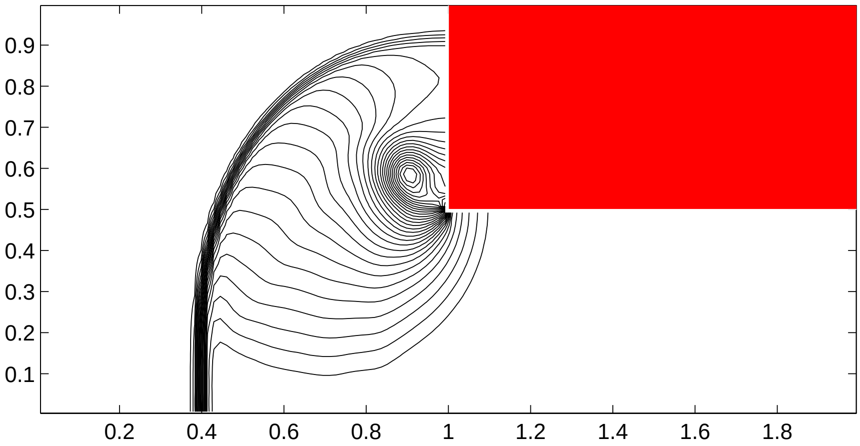


FIG. 4.11. *Equation* (4.5), *density;* $T = 0.5$, 128×64 *grid,* 30 *contours.*

where p denotes the pressure and $\vec{u} = (u, v)$ is the divergence-free velocity field, satisfying $u_x + v_y = 0$. In the 2-D case, equation (4.7) admits an equivalent vorticity formulation, which can be written either in the conservative form,

$$\omega_t + (u\omega)_x + (v\omega)_y = \nu\Delta\omega, \tag{4.8}$$

or in the transport form,

$$\omega_t + u\omega_x + v\omega_y = \nu\Delta\omega, \tag{4.9}$$

where ω is the vorticity, $\omega := v_x - u_y$. Equation (4.8) can be viewed as a 2-D conservation law

$$\omega_t + f(\omega)_x + g(\omega)_y = \nu\Delta\omega, \tag{4.10}$$

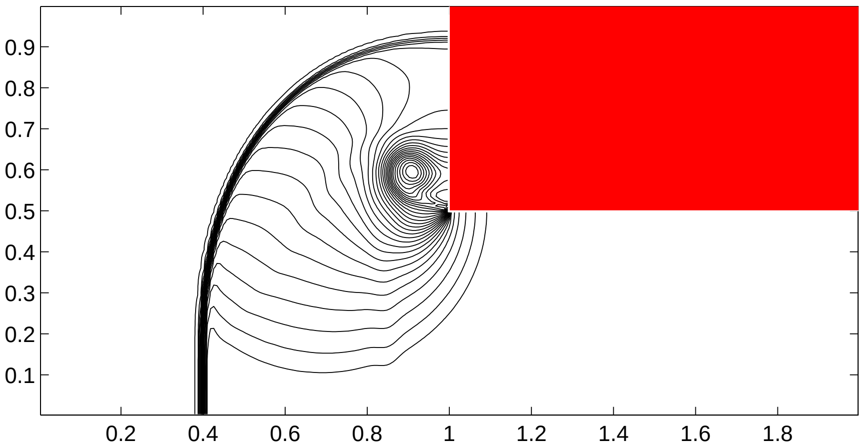


FIG. 4.12. *Equation* (4.5)*, density;* $T = 0.5$, 256×128 *grid,* 30 *contours.*

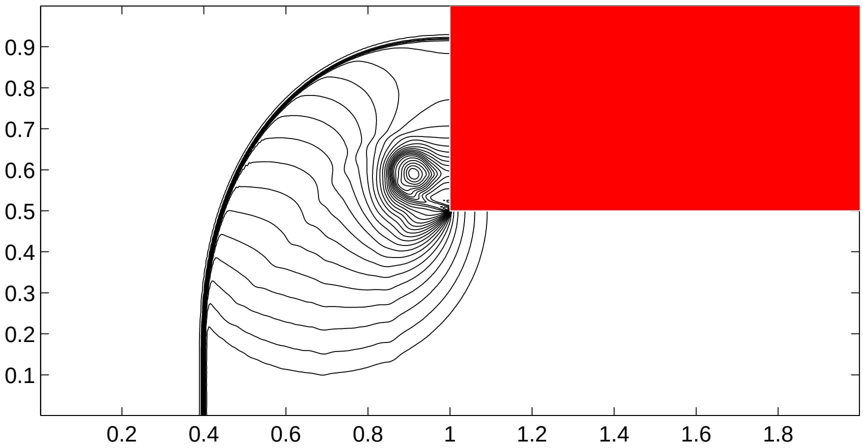


FIG. 4.13. *Equation* (4.5)*, density;* $T = 0.5$, 512×256 *grid,* 30 *contours.*

with a global flux $(f, g) := (u\omega, v\omega)$, while the transport equation (4.9) can be considered as a 2-D (viscous) Hamilton–Jacobi equation

$$\omega_t + H(\omega_x, \omega_y) = \nu\Delta\omega, \tag{4.11}$$

with a global Hamiltonian $H(\omega_x, \omega_y) = u\omega_x + v\omega_y$.

We propose two alternative Godunov-type semidiscrete central-upwind schemes for these two different formulations of the vorticity equation, (4.8) and (4.9). First, we consider (4.8) as a conservation law (4.10) and apply our 2-D third-order central-upwind scheme (3.30)–(3.32) to it. The resulting scheme has the conservative form

$$\frac{d}{dt}\bar{\omega}_{j,k}(t) = -\frac{H^x_{j+\frac{1}{2},k}(t) - H^x_{j-\frac{1}{2},k}(t)}{\Delta x} - \frac{H^y_{j,k+\frac{1}{2}}(t) - H^y_{j,k-\frac{1}{2}}(t)}{\Delta y} + \nu Q_{j,k}(t). \tag{4.12}$$

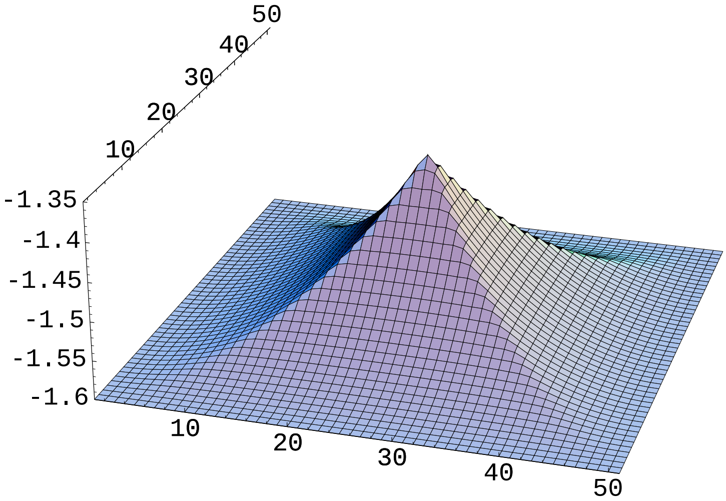


FIG. 4.14. *Problem* (4.6)*;* $T = 0.6$*,* 50×50 *grid.*

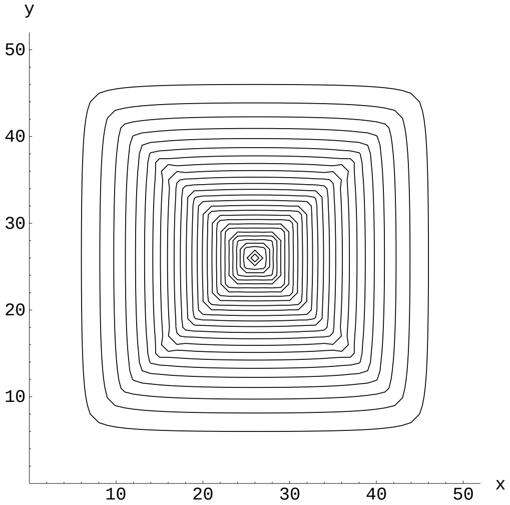


FIG. 4.15. *Contour plot.*

The following choice of the one-sided local speeds,

$$a^{\pm}_{j+\frac{1}{2},k} := \left(u_{j+\frac{1}{2},k}\right)_{\pm}, \qquad b^{\pm}_{j,k+\frac{1}{2}} := \left(v_{j,k+\frac{1}{2}}\right)_{\pm}, \tag{4.13}$$

yields the simplified numerical convection fluxes

$$\begin{aligned} H^x_{j+\frac{1}{2},k} := {} & \frac{a^+_{j+\frac{1}{2},k}}{6\left(a^+_{j+\frac{1}{2},k} - a^-_{j+\frac{1}{2},k}\right)} \left[u_{j+\frac{1}{2},k+\frac{1}{2}}\omega^{\mathrm{NE}}_{j,k} + 4u_{j+\frac{1}{2},k}\omega^{\mathrm{E}}_{j,k} + u_{j+\frac{1}{2},k-\frac{1}{2}}\omega^{\mathrm{SE}}_{j,k}\right] \\ & - \frac{a^-_{j+\frac{1}{2},k}}{6\left(a^+_{j+\frac{1}{2},k} - a^-_{j+\frac{1}{2},k}\right)} \left[u_{j+\frac{1}{2},k+\frac{1}{2}}\omega^{\mathrm{NW}}_{j+1,k} + 4u_{j+\frac{1}{2},k}\omega^{\mathrm{W}}_{j+1,k} + u_{j+\frac{1}{2},k-\frac{1}{2}}\omega^{\mathrm{SW}}_{j+1,k}\right], \end{aligned} \tag{4.14}$$

and

$$H^y_{j,k+\frac{1}{2}} := \frac{b^+_{j,k+\frac{1}{2}}}{6\Big(b^+_{j,k+\frac{1}{2}} - b^-_{j,k+\frac{1}{2}}\Big)}\Big[v_{j-\frac{1}{2},k+\frac{1}{2}}\omega^{\mathrm{NW}}_{j,k} + 4v_{j,k+\frac{1}{2}}\omega^{\mathrm{N}}_{j,k} + v_{j+\frac{1}{2},k+\frac{1}{2}}\omega^{\mathrm{NE}}_{j,k}\Big]$$

$$- \frac{b^-_{j,k+\frac{1}{2}}}{6\Big(b^+_{j,k+\frac{1}{2}} - b^-_{j,k+\frac{1}{2}}\Big)}\Big[v_{j-\frac{1}{2},k+\frac{1}{2}}\omega^{\mathrm{SW}}_{j,k+1} + 4v_{j,k+\frac{1}{2}}\omega^{\mathrm{S}}_{j,k+1} + v_{j+\frac{1}{2},k+\frac{1}{2}}\omega^{\mathrm{SE}}_{j,k+1}\Big]. \tag{4.15}$$

The diffusion term in (4.12) is obtained by the fourth-order central differencing,

$$Q_{j,k} = \frac{-\bar\omega_{j+2,k} + 16\bar\omega_{j+1,k} - 30\bar\omega_{j,k} + 16\bar\omega_{j-1,k} - \bar\omega_{j-2,k}}{12(\Delta x)^2}$$

$$+ \frac{-\bar\omega_{j,k+2} + 16\bar\omega_{j,k+1} - 30\bar\omega_{j,k} + 16\bar\omega_{j,k-1} - \bar\omega_{j,k-2}}{12(\Delta y)^2}. \tag{4.16}$$

The intermediate values of the velocities, required to compute the convection fluxes (4.14) and (4.15), are approximated by the fourth-order averaging, namely,

$$u_{j+\frac{1}{2},k} = \frac{-u_{j+2,k} + 9u_{j+1,k} + 9u_{j,k} - u_{j-1,k}}{16},$$

$$v_{j,k+\frac{1}{2}} = \frac{-v_{j,k+2} + 9v_{j,k+1} + 9v_{j,k} - v_{j,k-1}}{16}. \tag{4.17}$$

The velocities at the grid points, $\{u_{j,k}, v_{j,k}\}$, are recovered from the computed vorticities $\{\omega_{j,k}\}$ at every time step. This is done with the help of the streamfunction ψ, such that $u = \psi_y$, $v = -\psi_x$, and $\Delta\psi = -\omega$. We solve this Poisson equation by the nine-point Laplacian approximation. Then, having the values of $\{\psi_{j,k}\}$, we compute the velocities

$$u_{j,k} = \frac{-\psi_{j,k+2} + 8\psi_{j,k+1} - 8\psi_{j,k-1} + \psi_{j,k-2}}{12\Delta y},$$

$$v_{j,k} = \frac{\psi_{j+2,k} - 8\psi_{j+1,k} + 8\psi_{j-1,k} - \psi_{j-2,k}}{12\Delta x}. \tag{4.18}$$

This completes the construction of the "conservative" central-upwind scheme for the incompressible Euler and Navier–Stokes equations.

We now apply this scheme, coupled with the reconstruction in [25], to the Navier–Stokes equation (4.8) with $\nu = 0.05$, augmented with the smooth periodic initial data

$$u(x,y,0) = -\cos x \sin y, \qquad v(x,y,0) = \sin x \cos y. \tag{4.19}$$

This test problem, proposed in [9], admits the exact classical solution, given by

$$u(x,y,t) = -e^{-2\nu t}\cos x \sin y, \qquad v(x,y,t) = e^{-2\nu t}\sin x \cos y.$$

In this numerical experiment, we check the accuracy of our scheme. The numerical solution is computed at time $T = 2$, and the errors for the vorticity, measured in the L^∞-, L^1-, and L^2-norms, are presented in Table 4.2. These results indicate the expected third-order accuracy of the proposed scheme (4.12)–(4.18).

Next, we apply the scheme (4.12)–(4.18) together with the reconstruction in [25] to the Euler equation, (4.8) with $\nu = 0$, subject to the $(2\pi, 2\pi)$-periodic initial data

$$u(x,y,0) = \begin{cases} \tanh(\frac{1}{\rho}(y - \pi/2)), & y \le \pi, \\ \tanh(\frac{1}{\rho}(3\pi/2 - y)), & y > \pi, \end{cases} \qquad v(x,y,0) = \delta \cdot \sin x. \tag{4.20}$$

Table 4.2

Accuracy test for the third-order "conservative" scheme for the Navier–Stokes equation (4.8), (4.19), $\nu = 0.05$; *errors at* $T = 2$.

$Nx \times Ny$	L^{∞}-error	rate	L^{1}-error	rate	L^{2}-error	rate
32×32	2.103e-03	–	2.761e-02	–	5.623e-03	–
64×64	2.788e-04	2.92	3.652e-03	2.92	7.404e-04	2.92
128×128	3.548e-05	2.97	4.636e-04	2.98	9.385e-05	2.98
256×256	4.444e-06	3.00	5.811e-05	3.00	1.176e-05	3.00

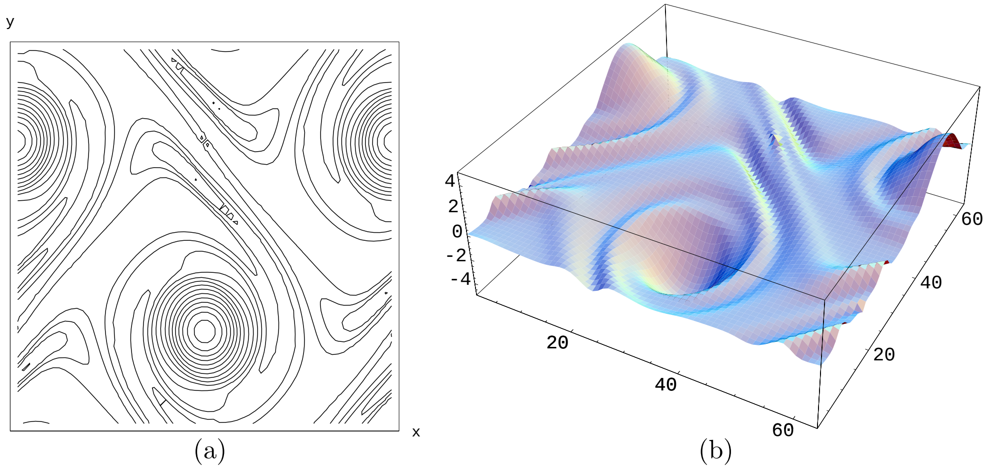


Fig. 4.16. *Vorticity—"conservative" scheme,* 64×64.

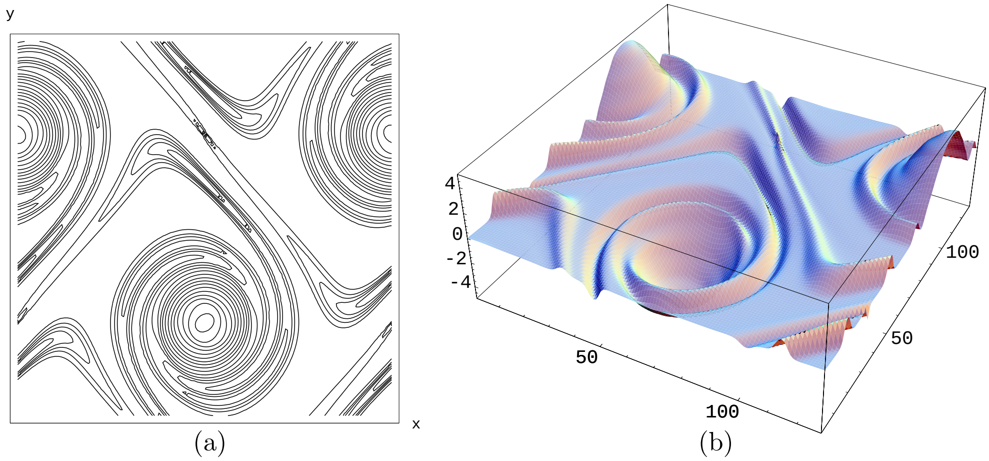


Fig. 4.17. *Vorticity—"conservative" scheme,* 128×128.

This is the double shear-layer model problem, proposed in [7]. We take $\rho = \pi/15$ and $\delta = 0.05$. Figures 4.16(a) and 4.17(a) are the contour plots (30 contours) of the vorticity at time $T = 10$ with 64×64 and 128×128 grid points, respectively. The three-dimensional plots of the same results are shown in Figures 4.16(b) and 4.17(b). The performed numerical experiments demonstrate that our scheme provides a very high resolution. These results are comparable with the results obtained by the third-

TABLE 4.3
Accuracy test for the second-order "transport" scheme for the Navier–Stokes equation (4.9), (4.19), $\nu = 0.05$; *errors at* $T = 2$.

$Nx \times Ny$	L^∞-error	rate	L^1-error	rate	L^2-error	rate
32×32	5.559e-03	–	7.304e-02	–	1.492e-02	–
64×64	1.672e-03	1.73	2.265e-02	1.69	4.574e-03	1.71
128×128	4.531e-04	1.88	6.263e-03	1.85	1.250e-03	1.87
256×256	1.176e-04	1.95	1.644e-03	1.93	3.276e-04	1.93

order semidiscrete central scheme in [25].

The second alternative is to solve the vorticity equation in its transport form, (4.9), which can be viewed as a Hamilton–Jacobi equation (4.11).

We choose the one-sided local speeds to be

$$a^{\pm}_{j,k} := (u_{j,k})_{\pm}, \qquad b^{\pm}_{j,k} := (v_{j,k})_{\pm}, \tag{4.21}$$

and in this setting, our 2-D second-order central-upwind scheme (3.49)–(3.50) has the following simple form:

$$\begin{aligned} \frac{d}{dt}\omega_{j,k}(t) = &-\frac{a^-_{j,k}b^-_{j,k}(u_{j,k}\omega_x^+ + v_{j,k}\omega_y^+) - a^-_{j,k}b^+_{j,k}(u_{j,k}\omega_x^+ + v_{j,k}\omega_y^-)}{(a^+_{j,k} - a^-_{j,k})(b^+_{j,k} - b^-_{j,k})} \\ &+\frac{a^+_{j,k}b^-_{j,k}(u_{j,k}\omega_x^- + v_{j,k}\omega_y^+) - a^+_{j,k}b^+_{j,k}(u_{j,k}\omega_x^- + v_{j,k}\omega_y^-)}{(a^+_{j,k} - a^-_{j,k})(b^+_{j,k} - b^-_{j,k})} + \nu L_{j,k}. \end{aligned} \tag{4.22}$$

Here, $w_x^{\pm}$, $w_y^{\pm}$ are the right and the left derivatives in the x- and y-directions of the continuous piecewise polynomial reconstruction of $\{\omega_{j,k}\}$. The term $L_{j,k}$ stands for the standard central difference approximation of the linear viscous term, that is,

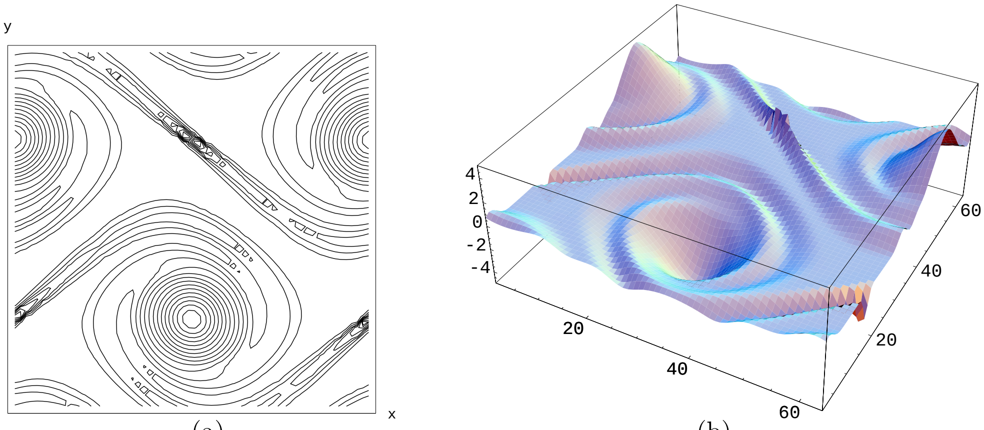
$$L_{j,k} = \frac{\omega_{j+1,k} - 2\omega_{j,k} + \omega_{j-1,k}}{(\Delta x)^2} + \frac{\omega_{j,k+1} - 2\omega_{j,k} + \omega_{j,k-1}}{(\Delta y)^2}. \tag{4.23}$$

As in the "conservative" scheme, we recover the velocities $\{u_{j,k}, v_{j,k}\}$ from the known values of the vorticity $\{\omega_{j,k}\}$, using the streamfunction approach. At each time step we solve the five-points Laplacian $\Delta\psi_{j,k} = -\omega_{j,k}$ and compute the velocities as follows:

$$u_{j,k} = \frac{\psi_{j,k+1} - \psi_{j,k-1}}{2\Delta y}, \qquad v_{j,k} = -\frac{\psi_{j+1,k} - \psi_{j-1,k}}{2\Delta x}. \tag{4.24}$$

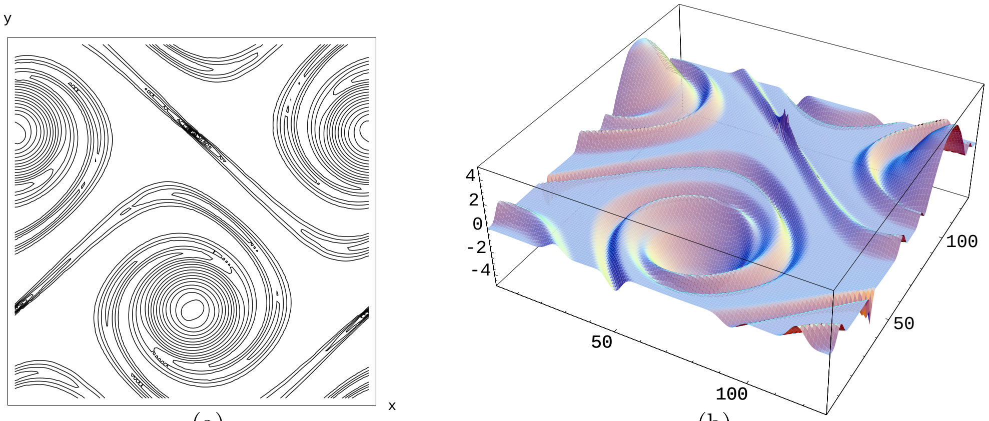
We now apply this second-order "transport" scheme to the IBVP for the Navier–Stokes equation (4.9), (4.19) with $\nu = 0.05$. The numerical solution of this test problem is computed at time $T = 2$, and the L^∞-, L^1-, and L^2-errors for the vorticity are presented in Table 4.3. These results indicate the second-order convergence rate measured in all these norms. We would also like to point out that the absolute values of the errors here are about 10 times smaller than the corresponding errors obtained by the semidiscrete central scheme in [27, Table 6.1].

Finally, we test our scheme (4.21)–(4.24) on the double shear-layer problem (4.9), (4.20). The contour plots (30 contours) of the vorticity are shown in Figures 4.18(a), 4.19(a), and 4.20(a), where the computations are performed at time $T = 10$ with 64×64, 128×128, and 256×256 grid points. The corresponding three-dimensional plots are presented in Figures 4.18(b), 4.19(b), and 4.20(b).



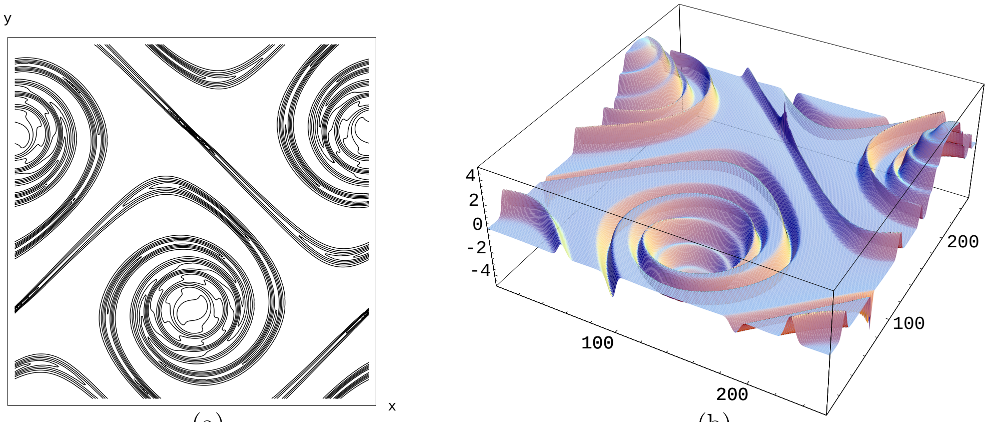
(a) (b)

FIG. 4.18. *Vorticity—"transport" scheme,* 64 × 64.



(a) (b)

FIG. 4.19. *Vorticity—"transport" scheme,* 128 × 128.



(a) (b)

FIG. 4.20. *Vorticity—"transport" scheme,* 256 × 256.

We would like to point out that our second-order "transport" scheme (4.21)–(4.24) has a very high resolution. The numerical experiments show that it is far superior to the second-order semidiscrete central scheme in [27, Figures 6.7–6.10]. This improvement is attributed to the smaller numerical viscosity present in the central-upwind scheme. Moreover, the resolution of (4.21)–(4.24) is almost as good as the resolution of our third-order "conservative" scheme (Figures 4.16 and 4.17).

As in [27, Figures 6.9–6.10], our numerical solution has spurious spikes, but of smaller heights. Also, the consequent mesh refinements (Figures 4.18–4.20) clearly demonstrate that the supports of these spikes diminish as the mesh size decreases.

Concluding remark. We have already mentioned the benefits of using the new central-upwind schemes in comparison to the central schemes in [26, 27, 24, 25]. Namely, they are *less dissipative*, and at the same time they retain the major advantages of central schemes—*simplicity* and *efficiency*.

In particular, the effect of the reduced numerical dissipation can be clearly seen when solving the incompressible Euler equation in its transport form. Moreover, in all of the numerical examples presented above, the achieved resolution is slightly better than the resolution obtained in [27, 24, 25].

The only drawback is the fact that the new schemes require the computation of both left and right local speeds, which increases the computational costs. However, the increase is not substantial, because as in any central scheme, our central-upwind schemes are Riemann-solver-free and do not require any computationally expensive characteristic decomposition.

REFERENCES

[1] R. Abgrall, *Numerical discretization of first-order Hamilton-Jacobi equation on triangular meshes*, Comm. Pure Appl. Math., 49 (1996), pp. 1339–1373.

[2] A.M. Anile, V. Romano, and G. Russo, *Extended hydrodynamical model of carrier transport in semiconductors*, SIAM J. Appl. Math., 61 (2000), pp. 74–101.

[3] P. Arminjon and M.-C. Viallon, *Généralisation du schéma de Nessyahu-Tadmor pour une équation hyperbolique à deux dimensions d'espace*, C. R. Acad. Sci. Paris Sér. I Math., 320 (1995), pp. 85–88.

[4] P. Arminjon, D. Stanescu, and M.-C. Viallon, *A two-dimensional finite volume extension of the Lax-Friedrichs and Nessyahu-Tadmor schemes for compressible flows*, in Proceedings of the 6th International Symposium on Computational Fluid Dynamics, Vol. 4, Lake Tahoe, CA, M. Hafez and K. Oshima, eds., 1995, pp. 7–14.

[5] P. Arminjon, M.-C. Viallon, and A. Madrane, *A finite volume extension of the Lax-Friedrichs and Nessyahu-Tadmor schemes for conservation laws on unstructured grids*, Int. J. Comput. Fluid Dyn., 9 (1997), pp. 1–22.

[6] P. Arminjon and M.-C. Viallon, *Convergence of a finite volume extension of the Nessyahu–Tadmor scheme on unstructured grids for a two-dimensional linear hyperbolic equation*, SIAM J. Numer. Anal., 36 (1999), pp. 738–771.

[7] J.B. Bell, P. Colella, and H.M. Glaz, *A second-order projection method for the incompressible Navier-Stokes equations*, J. Comput. Phys., 85 (1989), pp. 257–283.

[8] F. Bianco, G. Puppo, and G. Russo, *High-order central schemes for hyperbolic systems of conservation laws*, SIAM J. Sci. Comput., 21 (1999), pp. 294–322.

[9] A. Chorin, *Numerical solution of the Navier-Stokes equations*, Math. Comp., 22 (1968), pp. 745–762.

[10] L. Corrias, M. Falcone, and R. Natalini, *Numerical schemes for conservation laws via Hamilton-Jacobi equations*, Math. Comp., 64 (1995), pp. 555–580.

[11] M.G. Crandall and P.-L. Lions, *Two approximations of solutions of Hamilton-Jacobi equations*, Math. Comp., 43 (1984), pp. 1–19.

[12] B. Einfeldt, *On Godunov-type methods for gas dynamics,* SIAM J. Numer. Anal., 25 (1988), pp. 294–318.

[13] B. Engquist and O. Runborg, *Multi-phase computations in geometrical optics*, J. Comput. Appl. Math., 74 (1996), pp. 175–192.

[14] K.O. Friedrichs, *Symmetric hyperbolic linear differential equations*, Comm. Pure Appl. Math., 7 (1954), pp. 345–392.

[15] A. Harten, *High resolution schemes for hyperbolic conservation laws*, J. Comput. Phys., 49 (1983), pp. 357–393.

[16] A. Harten, B. Engquist, S. Osher, and S.R. Chakravarthy, *Uniformly high order accurate essentially non-oscillatory schemes* III, J. Comput. Phys., 71 (1987), pp. 231–303.

[17] A. Harten, P.D. Lax, and B. van Leer, *On upstream differencing and Godunov-type schemes for hyperbolic conservation laws*, SIAM Rev., 25 (1983), pp. 35–61.

[18] G.-S. Jiang and C.-W. Shu, *Efficient implementation of weighted ENO schemes*, J. Comput. Phys., 126 (1996), pp. 202–228.

[19] G.-S. Jiang and E. Tadmor, *Nonoscillatory central schemes for multidimensional hyperbolic conservation laws*, SIAM J. Sci. Comput., 19 (1998), pp. 1892–1917.

[20] R. Kupferman, *Simulation of viscoelastic fluids: Couette-Taylor flow*, J. Comput. Phys., 147 (1998), pp. 22–59.

[21] R. Kupferman, *A numerical study of the axisymmetric Couette–Taylor problem using a fast high-resolution second-order central scheme*, SIAM J. Sci. Comput., 20 (1998), pp. 858–877.

[22] R. Kupferman and E. Tadmor, *A fast high-resolution second-order central scheme for incompressible flows*, Proc. Natl. Acad. Sci. USA, 94 (1997), pp. 4848–4852.

[23] A. Kurganov, *Conservation Laws: Stability of Numerical Approximations and Nonlinear Regularization*, Ph.D. Thesis, Tel-Aviv University, Israel, 1997.

[24] A. Kurganov and D. Levy, *A third-order semidiscrete central scheme for conservation laws and convection-diffusion equations*, SIAM J. Sci. Comput., 22 (2000), pp. 1461–1488.

[25] A. Kurganov and G. Petrova, *A third-order semi-discrete genuinely multidimensional central scheme for hyperbolic conservation laws and related problems*, Numer. Math., to appear.

[26] A. Kurganov and E. Tadmor, *New high-resolution central schemes for nonlinear conservation laws and convection-diffusion equations*, J. Comput. Phys., 160 (2000), pp. 241–282.

[27] A. Kurganov and E. Tadmor, *New high-resolution semi-discrete central schemes for Hamilton-Jacobi equations*, J. Comput. Phys., 160 (2000), pp. 720–742.

[28] P.D. Lax, *Weak solutions of nonlinear hyperbolic equations and their numerical computation*, Comm. Pure Appl. Math., 7 (1954), pp. 159–193.

[29] B. van Leer, *Towards the ultimate conservative difference scheme,* V. *A second order sequel to Godunov's method*, J. Comput. Phys., 32 (1979), pp. 101–136.

[30] D. Levy, G. Puppo, and G. Russo, *Central WENO schemes for hyperbolic systems of conservation laws,* M2AN Math. Model. Numer. Anal., 33 (1999), pp. 547–571.

[31] D. Levy, G. Puppo, and G. Russo, *A third order central WENO scheme for* 2D *conservation laws*, Appl. Numer. Math., 33 (2000), pp. 407–414.

[32] D. Levy, G. Puppo, and G. Russo, *Compact central WENO schemes for multidimensional conservation laws*, SIAM J. Sci. Comput., 22 (2000), pp. 656–672.

[33] D. Levy and E. Tadmor, *Non-oscillatory central schemes for the incompressible* 2-D *Euler equations*, Math. Res. Lett., 4 (1997), pp. 1–20.

[34] K.-A. Lie and S. Noelle, *Remarks on High-Resolution Non-Oscillatory Central Schemes for Multi-Dimensional Systems of Conservation Laws. Part* I: *An Improved Quadrature Rule for the Flux-Computation*, SFB256 preprint 679, Bonn University, Germany.

[35] C.-T. Lin and E. Tadmor, *L^1-stability and error estimates for approximate Hamilton-Jacobi solutions*, Numer. Math., 87 (2001), pp. 701–735.

[36] C.-T. Lin and E. Tadmor, *High-resolution nonoscillatory central schemes for Hamilton–Jacobi equations*, SIAM J. Sci. Comput., 21 (2000), pp. 2163–2186.

[37] X.-D. Liu and S. Osher, *Nonoscillatory high order accurate self-similar maximum principle satisfying shock capturing schemes.* I, SIAM J. Numer. Anal., 33 (1996), pp. 760–779.

[38] X.-D. Liu, S. Osher, and T. Chan, *Weighted essentially non-oscillatory schemes*, J. Comput. Phys., 115 (1994), pp. 200–212.

[39] X.-D. Liu and E. Tadmor, *Third order nonoscillatory central scheme for hyperbolic conservation laws*, Numer. Math., 79 (1998), pp. 397–425.

[40] H. Nessyahu and E. Tadmor, *Non-oscillatory central differencing for hyperbolic conservation laws*, J. Comput. Phys., 87 (1990), pp. 408–463.

[41] S. Noelle, *A comparison of third and second order accurate finite volume schemes for the two-dimensional compressible Euler equations*, in Proceedings of the Seventh International Conference on Hyperbolic Problems, Zürich, 1998, Internat. Ser. Numer. Math.

129, Birkhäuser, Basel, pp. 757–766.

[42] S. Osher and C.-W. Shu, *High-order essentially nonoscillatory schemes for Hamilton-Jacobi equations*, SIAM J. Numer. Anal., 28 (1991), pp. 907–922.

[43] S. Osher and E. Tadmor, *On the convergence of difference approximations to scalar conservation laws*, Math. Comp., 50 (1988), pp. 19–51.

[44] V. Romano and G. Russo, *Numerical solution for hydrodynamical models of semiconductors*, Math. Models Methods Appl. Sci., 10 (2000), pp. 1099–1120.

[45] C.-W. Shu, *Total-variation-diminishing time discretizations*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 1073–1084.

[46] C.-W. Shu, *Numerical experiments on the accuracy of ENO and modified ENO schemes*, J. Sci. Comput., 5 (1990), pp. 127–149.

[47] C.-W. Shu and S. Osher, *Efficient implementation of essentially non-oscillatory shock-capturing schemes*, J. Comput. Phys., 77 (1988), pp. 439–471.

[48] P.E. Souganidis, *Approximation schemes for viscosity solutions of Hamilton-Jacobi equations*, J. Differential Equations., 59 (1985), pp. 1–43.

[49] Y.C. Tai, S. Noelle, N. Gray, and K. Hutter, *Shock Capturing and Front Tracking Methods for Granular Avalanches*, SFB256 Preprint 678, Bonn University, Germany; J. Comput. Phys., to appear.

[50] B. Wendroff, *Approximate Riemann solvers, Godunov schemes, and contact discontinuities*, in *Godunov Methods: Theory and Applications*, E. F. Toro, ed., Kluwer Academic/Plenum Publishers, New York, to appear.

[51] P. Woodward and P. Colella, *The numerical solution of two-dimensional fluid flow with strong shocks*, J. Comput. Phys., 54 (1988), pp. 115–173.

SIAM J. Sci. Comput.
Vol. 23, No. 3, pp. 741–760

A NEW FAST-MULTIPOLE ACCELERATED POISSON SOLVER IN TWO DIMENSIONS*

FRANK ETHRIDGE† AND LESLIE GREENGARD‡

Abstract. We present an adaptive fast multipole method for solving the Poisson equation in two dimensions. The algorithm is direct, assumes that the source distribution is discretized using an adaptive quad-tree, and allows for Dirichlet, Neumann, periodic, and free-space conditions to be imposed on the boundary of a square. The amount of work *per grid point* is comparable to that of classical fast solvers, even for highly nonuniform grids.

Key words. fast multipole method, Poisson equation, adaptive refinement, fast Poisson solver

AMS subject classifications. 31A10, 35J05, 65R10, 78A30

PII. S1064827500369967

1. Introduction.

A variety of problems in scientific computing involve the solution of the Poisson equation

$$\Delta\psi = f, \tag{1}$$

subject to appropriate radiation or boundary conditions. In simple geometries (circular or rectangular domains) with regular grids, there are well-known fast direct solvers [6, 7] which typically rely on the fast Fourier transform (FFT) and are well suited to the task. When either restriction is relaxed, however, these methods no longer apply. Since practical problems tend to involve complex geometries, highly inhomogeneous source distributions f, or both, there has been a lot of effort directed at developing alternative approaches. Most currently available solvers rely on iterative techniques using multigrid, domain decomposition, or some other preconditioning strategy [5, 9, 21]. Unfortunately, while such multilevel strategies can achieve nearly optimal efficiency in theory, they require an appropriate hierarchy of coarse grids which is not provided in practice. Although there has been significant progress in this direction [1, 2, 10, 11, 20, 23], the available solvers compare unfavorably with the fast direct solvers in terms of work per grid point.

In this paper, we describe an integral equation method for solving the Poisson equation in two dimensions which is direct, high order accurate, insensitive to the degree of adaptive mesh refinement, and accelerated by the fast multipole method (FMM) [16, 17, 26]. It is competitive with standard fast solvers in terms of *work per grid point.* This is a rather stringent test, since we compare the time for a classical, FFT-based solver using N mesh points with our adaptive, FMM-based solver using the same number of points, ignoring the fact that the latter solver uses grids which are highly inhomogeneous. We allow for the imposition of various combinations of free-space, periodic, Dirichlet, and Neumann conditions on the boundary of a square.

Earlier work on FMM-based integral equation schemes in two dimensions includes [15, 22, 29]. The paper [22] describes a fast Poisson solver for complex geometries,

*Received by the editors April 3, 2000; accepted for publication (in revised form) December 8, 2000; published electronically August 15, 2001.
http://www.siam.org/journals/sisc/23-3/36996.html

†Department of Computer Science, Yale University, New Haven, CT 06520 (ethridge@cs.yale.edu).

‡Courant Institute of Mathematical Sciences, New York University, New York, NY 10012 (greengard@cims.nyu.edu). The work of this author was supported by the Applied Mathematical Sciences Program of the U.S. Department of Energy under contract DEFGO288ER25053.

where the boundary can be arbitrarily shaped and multiply-connected, but where the right-hand side is specified on a uniform underlying mesh. The other two papers discuss the inversion of (1) in free space by evaluation of the analytic solution

$$\psi(\mathbf{x}) = \frac{1}{2\pi} \int_D f(\mathbf{y}) \log(|\mathbf{x} - \mathbf{y}|) \, d\mathbf{y}. \tag{2}$$

The algorithms of both [15] and [29] are highly adaptive, with the former relying on a quad-tree and the latter relying on an unstructured triangulation. Neither, however, goes beyond the free-space problem.

The present approach is similar to that outlined in [15] but differs in several respects.

1. In the method of [15], one solves local Poisson problems with spectral methods on each leaf node of a quad-tree data structure and then patches the solutions together using the FMM in a domain decomposition approach. Here, we apply the FMM directly to the volume integral, using high order quadratures.
2. We incorporate a new version of the FMM described in [19], which is based on diagonal forms for translation operators (see section 3.8).
3. We incorporate the method of images to solve a variety of boundary value problems on a square (with adaptive refinement).
4. For fourth and sixth order accurate discretizations, we use locally uniform meshes, compatible with adaptive mesh refinement (AMR) data structures [3]. For eighth order accuracy, we follow [13, 15, 24] and rely on local spectral meshes.

The paper is organized as follows. In section 2, we outline the relevant potential theory, with particular emphasis on the method of images. In section 3, we describe the fast multipole algorithm itself, and in section 4, we present several numerical examples. Finally, it should be noted that our algorithm shares a number of features with the recently developed scheme of [12] for solving the pseudodifferential equation

$$(-\Delta)^{1/2} \psi = \omega \tag{3}$$

in the plane via the integral representation

$$\psi(\mathbf{x}) = \int_{\mathbf{R}^2} \frac{\omega(\mathbf{y})}{|\mathbf{x} - \mathbf{y}|} \, d\mathbf{y}.$$

2. Potential theory.

2. Potential theory. To complete a description of a well-posed problem, we must obviously add to the Poisson equation (1) a specification of boundary conditions on the unit square D. We allow the free-space conditions defined by (2), periodic boundary conditions, Dirichlet conditions, and Neumann conditions. We can also handle mixed conditions, but assume that the transition from one type to another (Dirichlet–Neumann, etc.) occurs only at corners. The solution to all these problems can be constructed analytically using the method of images.

For periodic boundary conditions, one simply imagines the entire plane to be tiled with copies of the source distribution contained in the unit cell D. (How to compute the influence of each of these images efficiently is discussed in the next section.) For other boundary conditions, the construction is a bit more subtle. Let us consider, for example, the boundary value problem

$$\begin{aligned} \Delta\psi &= f && \text{in } D,\\ \psi &= g_L && \text{on } \Gamma_L,\\ \psi &= g_R && \text{on } \Gamma_R,\\ \partial\psi/\partial n &= g_T && \text{on } \Gamma_T,\\ \partial\psi/\partial n &= g_B && \text{on } \Gamma_B, \end{aligned} \tag{4}$$

where Γ_L denotes the "left" boundary ($x = -0.5, -0.5 \leq y \leq 0.5$), Γ_R denotes the "right" boundary ($x = 0.5, -0.5 \leq y \leq 0.5$), Γ_T denotes the "top" boundary ($-0.5 \leq x \leq 0.5, y = 0.5$), and Γ_B denotes the "bottom" boundary ($-0.5 \leq x \leq 0.5, y = -0.5$).

This problem can be conveniently broken up into two parts. First, we can solve the Poisson equation:

$$\begin{aligned} \Delta\psi_1 &= f && \text{in } D,\\ \psi_1 &= 0 && \text{on } \Gamma_L,\\ \psi_1 &= 0 && \text{on } \Gamma_R,\\ \partial\psi_1/\partial n &= 0 && \text{on } \Gamma_T,\\ \partial\psi_1/\partial n &= 0 && \text{on } \Gamma_B. \end{aligned} \tag{5}$$

Then we can solve the Laplace equation with inhomogeneous boundary conditions:

$$\begin{aligned} \Delta\psi_2 &= 0 && \text{in } D,\\ \psi_2 &= g_L && \text{on } \Gamma_L,\\ \psi_2 &= g_R && \text{on } \Gamma_R,\\ \partial\psi_2/\partial n &= g_T && \text{on } \Gamma_T,\\ \partial\psi_2/\partial n &= g_B && \text{on } \Gamma_B. \end{aligned} \tag{6}$$

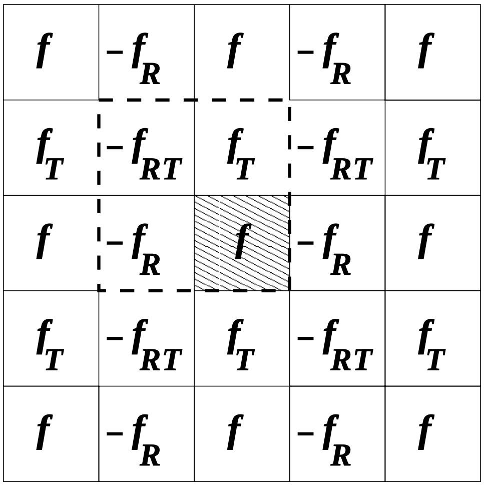
Clearly, $\psi = \psi_1 + \psi_2$.

To solve (5), suppose that we tile the plane with the pattern of images depicted in Figure 1. The shaded box is the computational domain containing the source distribution f. f_T denotes the even reflection of the function f across the top boundary Γ_T, $-f_R$ denotes the odd reflection of the function f across the right boundary Γ_R, and $-f_{RT}$ denotes the even reflection of the function $-f_R$ across the line $y = +\frac{1}{2}$. It is easy to verify that the vertical lines $x = \pm\frac{1}{2}$ are lines of odd symmetry and that the horizontal lines $y = \pm\frac{1}{2}$ are lines of even symmetry. Thus, the desired homogeneous boundary conditions are enforced if we account for the field due to all images. This task is simplified by the observation that the 2×2 supercell outlined with dashes in Figure 1 tiles the plane periodically.

To handle the inhomogeneous boundary conditions in (6), we recall the following classical results from potential theory [18, 30].

LEMMA 2.1. *Let $u(x, y)$ satisfy the Laplace equation $\Delta u = 0$ in the half-space $y > 0$ with Dirichlet boundary conditions $u(x, 0) = f(x)$. Then $u(x, y)$ is given by the double layer potential*

$$u(x, y) = 2\int_{-\infty}^{\infty} \frac{\partial G}{\partial y}(x - \xi, y)\, f(\xi)\, d\xi = \frac{1}{\pi}\int_{-\infty}^{\infty} \frac{y}{(x - \xi)^2 + y^2}\, f(\xi)\, d\xi.$$



f	$-f_R$	f	$-f_R$	f
f_T	$-f_{RT}$	f_T	$-f_{RT}$	f_T
f	$-f_R$	f	$-f_R$	f
f_T	$-f_{RT}$	f_T	$-f_{RT}$	f_T
f	$-f_R$	f	$-f_R$	f

FIG. 1. *A source distribution tiling the plane which solves the Poisson equation with the homogeneous boundary conditions described by the system* (5)*. The shaded box represents the computational domain itself.* f_T *denotes the even reflection of the function* f *across the top boundary* Γ_T*,* $-f_R$ *denotes the odd reflection of the function* f *across the right boundary* Γ_R*, and* $-f_{RT}$ *denotes the even reflection of the function* $-f_R$ *across the line* $y = +\frac{1}{2}$*. The* 2×2 *"supercell" with the dashed outline can be seen to tile the plane periodically.*

The solution satisfying Neumann boundary conditions $\frac{\partial u}{\partial n}(x, 0) = g(x)$ *is given by the single layer potential*

$$u(x, y) = 2\int_{-\infty}^{\infty} G(x - \xi, y)\, g(\xi)\, d\xi = \frac{1}{\pi}\int_{-\infty}^{\infty} \ln\sqrt{(x - \xi)^2 + y^2}\, g(\xi)\, d\xi.$$

Consider now the system of layer potentials depicted in Figure 2. We leave it to the reader to verify that, from the preceding lemma and symmetry considerations, the boundary conditions of (6) are satisfied. As with the tiling of source distributions, the task of accounting for the field due to all images is simplified by the observation that the layer potentials on boundary segments outlined with dots in Figure 2 tile the plane periodically. The evaluation of layer potentials is discussed in section 3.7.

3. Data structures and the FMM. We assume that the source distribution f in (2) is supported inside the unit square D, centered at the origin, on which is superimposed a hierarchy of refinements (a quad-tree). Grid level 0 is defined to be D itself, and grid level $l+1$ is obtained recursively by subdividing each square at level l into four equal parts. Using standard terminology, if d is a fixed square at level l, the four squares at level $l+1$ obtained by its subdivision will be referred to as its children. In order to allow for adaptivity, we do not use the same number of levels in all regions of D. We do, however, assume that the quad-tree satisfies one fairly standard restriction, namely, that two leaf nodes which share a boundary point must be no more than one refinement level apart (Figure 3).

3.1. The volume integral. We restrict our attention, for the moment, to the free-space problem. Extended volume integrals such as the ones depicted in Figure 1 will be discussed in section 3.6.

The leaf nodes on which the source distribution is given will be denoted by D_i.

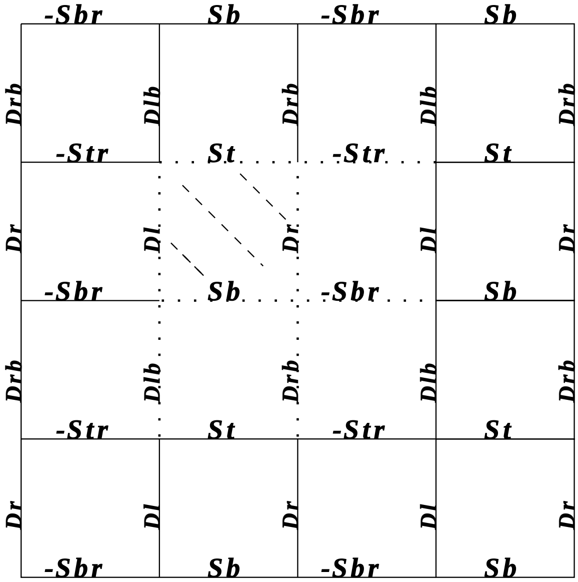


FIG. 2. *A tiling of the plane with layer potentials to solve* (6)*. The computational domain is indicated with diagonal dashes. Dl denotes a double layer potential with density g_L, and Dr denotes a double layer potential with density g_R. Their even reflections across the bottom boundary Γ_B are Dlb and Drb, respectively. St denotes a single layer potential with density g_T, and Sb denotes a single layer potential with density g_B. Their odd reflections across the right boundary Γ_R are $-Str$ and $-Sbr$, respectively. Symmetry considerations show that all four boundary conditions are satisfied. Note that the layer potentials on boundary segments outlined with dots tile the plane periodically.*

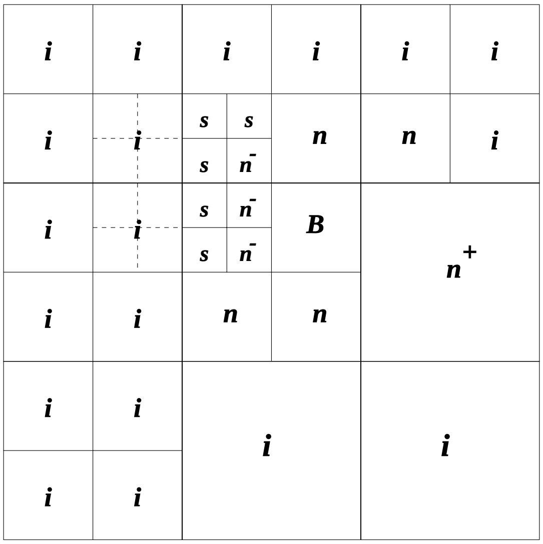


FIG. 3. *For the childless node B, colleagues are labeled n, coarse neighbors are labeled n^+, and fine neighbors are labeled n^-. The interaction list for B consists of the boxes marked i. The boxes marked by s are children of B's colleagues which are separated from B, so they are not fine neighbors. They constitute the* s-list *for B (see Definition* 3.1*).*

Thus, $D = \cup_{i=1}^{M} D_i$ and we rewrite (2) in the form

$$\psi(\mathbf{x}) = \sum_{i=1}^{M} \frac{1}{2\pi} \int_{D_i} f(\mathbf{y}) \log(|\mathbf{x} - \mathbf{y}|)\, d\mathbf{y}. \tag{7}$$

Definition 3.1. *The* colleagues *of a square B are squares at the same refinement level which share a boundary point with B. (B is considered to be a colleague of itself.) The* coarse neighbors *of B are leaf nodes at the level of B's parent which share a boundary point with B. The* fine neighbors *of B are leaf nodes one level finer than B which share a boundary point with B. Together, the union of the colleagues, coarse neighbors, and fine neighbors of B will be referred to as B's* neighbors. *The* s-list *of a box B consists of those children of B's colleagues which are not fine neighbors of B.*

The interaction region *for B consists of the area covered by the neighbors of B's parent, excluding the area covered by B's colleagues and coarse neighbors. The* interaction list *for B consists of those squares in the interaction region which are at the same refinement level, as well as leaf nodes in the interaction region which are at coarser levels. When the distinction is important, the squares at the same refinement level will be referred to as the* standard interaction list*, while the squares at coarser levels will be referred to as the* coarse interaction list.

In our FMM, following [8, 16, 17], terms in the convolution integral (7) from neighbor leaf nodes are computed directly. More distant interactions are accounted for on coarser levels, through the use of a hierarchy of far-field and local multipole expansions. We consider the local interactions first.

3.2. Local interactions. For fourth and sixth order accuracy, we assume that we are given f on a cell-centered $k \times k$ grid for each leaf node B, with $k = 4$ or 6, respectively. We can, therefore, take these k^2 data points and construct a kth order polynomial approximation to f of the form

$$f_B(x, y) \approx \sum_{j=1}^{N_k} c_B(j)\, b_j(x - x_B, y - y_B),$$

where $N_k = \frac{k(k+1)}{2}$ is the number of basis functions needed for kth order accuracy and where (x_B, y_B) denotes the center of B. The basis functions $b_1(x, y), \ldots, b_{N_k}(x, y)$ are given by

$$\{x^i y^j |\; i, j \geq 0, i + j \leq k - 1\}.$$

If we let $\vec{f_B} \in \mathbf{R}^{k^2}$ denote the given function values (in standard ordering), then the calculation of the coefficient vector $\vec{c}_B$ is clearly overdetermined. We obtain it through a least squares fit based on the singular value decomposition. The pseudoinverse matrix $\mathcal{P} \in \mathbf{R}^{N_k \times k^2}$, such that

$$\vec{c}_B = \mathcal{P}\, \vec{f_B},$$

can be precomputed and stored.

Remark 3.1. For eighth order accuracy, we assume that f is given on a scaled 8×8 classical tensor product Chebyshev grid [7] and use as basis functions

$$\{T_i(x) T_j(y) |\; i, j \geq 0, i + j \leq k - 1\},$$

where $T_i(x)$ denotes the Chebyshev polynomial of degree i. The coefficients of the Chebyshev expansion can be computed efficiently using the fast cosine transform.

Consider now a target point Q, which lies in a neighbor of B. The field induced at Q by f_B is approximated by

$$\psi_B(Q) = \sum_{n=1}^{N_k} c_B(n) f(Q,n), \tag{8}$$

where

$$f(Q,n) = \frac{1}{2\pi} \int_B b_n(x - x_B, y - y_B) \log |Q - (x,y)| \, dxdy. \tag{9}$$

Since the target points Q are regularly spaced in each neighboring square, we can precompute the weights (9) for each of the k^2 possible locations at each of 9 possible colleagues, 12 possible fine neighbors, and 12 possible coarse neighbors. To be more precise, we can precompute the weights assuming that B is the unit square $[-0.5, 0.5]^2$ because of the following straightforward lemma.

LEMMA 3.2. *Let B be a leaf node at level l and let Q denote a target point in one of B's neighbors. Let Q^* denote the scaled target point for the unit cell centered at the origin*

$$Q^* = 2^{l-1} \cdot (Q - (x_B, y_B)),$$

let

$$f^*(Q^*, n) = \frac{1}{2\pi} \int_{-1/2}^{1/2} \int_{-1/2}^{1/2} b_n(x,y) \log |Q^* - (x,y)| \, dxdy, \tag{10}$$

and let

$$\bar{f}(n,l) = \frac{1}{2\pi} \left(\frac{1}{2^{l-1}} \right)^{d+2} \int_{-1/2}^{1/2} \int_{-1/2}^{1/2} b_n(x,y) \log \left(\frac{1}{2^{l-1}} \right) dxdy. \tag{11}$$

Then the integral $f(Q,n)$ defined in (9) *is given by*

$$f(Q,n) = \left(\frac{1}{2^{l-1}} \right)^{d+2} f^*(Q^*, n) + \bar{f}(n,l),$$

where d is the degree of the polynomial basis function b_n.

Thus, we need only obtain weights for a box of unit area. Elementary counting arguments show that the storage required for this precomputation is

$$\begin{array}{rl} k \times k \cdot N_k \cdot 9 \text{ real numbers} & \text{for colleagues,} \\ k \times k \cdot N_k \cdot 12 \text{ real numbers} & \text{for fine neighbors,} \\ k \times k \cdot N_k \cdot 12 \text{ real numbers} & \text{for coarse neighbors,} \end{array}$$

for a total of approximately $17 \times k^4$ real numbers.

3.3. Far-field interactions. We turn now to the calculation of far-field interactions, which are computed by means of multipole expansions. We refer the reader to [14, 16] for more detailed discussions of potential theory. Our starting point is the usual multipole expansion for a charge distribution, which we state formally as a theorem.

THEOREM 3.3 (multipole expansion). *Let $\rho(\mathbf{y})$ be a charge distribution contained within a square D_i with center C and let $\Phi(\mathbf{x})$ denote the induced field at a point $\mathbf{x}$ in the interaction list of D_i:*

$$\Phi(\mathbf{x}) = \frac{1}{2\pi}\int_{D_i} \log|\mathbf{x}-\mathbf{y}|\rho(\mathbf{y})\, d\mathbf{y}.$$

Then $\Phi(\mathbf{x})$ can be described by the multipole expansion

$$\Phi(\mathbf{x}) = \alpha_0 \log|\mathbf{x}-C| + \Re\left(\sum_{k=1}^{\infty} \frac{\alpha_k}{(x_1+ix_2-C)^k}\right), \tag{12}$$

where C is viewed as a point in the complex plane, $\mathbf{x}=(x_1,x_2)$, and $\Re(w)$ denotes the real part of the complex quantity w. The coefficients α_k are given by

$$\begin{aligned}\alpha_0 &= \frac{1}{2\pi}\int_{D_i} \rho(y_1,y_2) dy_1\, dy_2,\\ \alpha_k &= -\frac{1}{2\pi}\int_{D_i} \frac{(y_1+iy_2-C)^k\rho(y_1,y_2)}{k} dy_1\, dy_2.\end{aligned} \tag{13}$$

In the hierarchical framework of the FMM, an upper bound for the error in truncating the expansion after n terms is given by

$$\left(\frac{1}{2}\right)^n \frac{1}{2\pi}\int_{D_i} |\rho(y_1,y_2)|\, dy_1\, dy_2. \tag{14}$$

THEOREM 3.4 (local expansion). *Let $\rho(\mathbf{y})$ be a charge distribution contained outside the neighbors of a square D_i with center C and let $\Psi(\mathbf{x})$ denote the induced field at $\mathbf{x}\in D_i$. Then $\Psi(\mathbf{x})$ can be described by a local expansion*

$$\Psi(\mathbf{x}) = \Re\left(\sum_{l=0}^{\infty} \beta_l (x_1+ix_2-C)^l\right), \tag{15}$$

where C is viewed as a point in the complex plane and $\mathbf{x}=(x_1,x_2)$.

The FMM relies on the ability to manipulate multipole and local expansions for every box in the tree hierarchy. We omit the technical details and refer the reader to the original papers [8, 14, 16, 17].

DEFINITION 3.5. *We denote by $S_{l,k}$ the kth square at refinement level l.*

We denote by $\Phi_{l,k}$ the multipole expansion describing the far field due to the source distribution supported inside $S_{l,k}$.

We denote by $\Psi_{l,k}$ the local expansion describing the field due to the source distribution outside the neighbors of $S_{l,k}$.

We denote by $\tilde{\Psi}_{l,k}$ the local expansion describing the field due to the source distribution outside the neighbors of the parent *of $S_{l,k}$.*

Remark 3.2. Let $S_{l,k}$ be a square in the quad-tree hierarchy and let $S_{l',k'}$ be a square in its interaction list. Then there is a linear operator $\mathcal{T}_{MM}$ for which

$$\Phi_{l,k} = \mathcal{T}_{MM}[\Phi(C_1), \Phi(C_2), \Phi(C_3), \Phi(C_4)], \tag{16}$$

where $\Phi(C_j)$ denotes the multipole expansion for the jth child of $S_{l,k}$. In other words, we can merge the expansions for four children into a single expansion for the parent. Similarly, there is a linear operator $\mathcal{T}_{LL}$ for which

$$[\tilde{\Psi}(C_1), \tilde{\Psi}(C_2), \tilde{\Psi}(C_3), \tilde{\Psi}(C_4)] = \mathcal{T}_{LL}\, \Psi_{l,k}, \tag{17}$$

where C_j denotes the jth child of $S_{l,k}$. In other words, we can shift the local expansion Ψ for a box to the corresponding expansion $\tilde{\Psi}$ for each of its children. Finally, there is a linear operator $\mathcal{T}_{ML}$ for which the field in $S_{l,k}$ due to the source distribution in $S_{l',k'}$ is described by $\Psi = T_{ML}\Phi_{l',k'}$. It is easy to verify that

$$\Psi_{l,k} = \tilde{\Psi}_{l,k} + \sum_{i \in \mathcal{IL}} T_{ML}\Phi_i, \tag{18}$$

where $\mathcal{IL}$ denotes the interaction list for square $S_{l,k}$.

Remark 3.3. One slight complication in the adaptive algorithm concerns the interaction between boxes of different sizes. Referring to Figure 3, we need to account for the influence of a childless square B on each box marked s and vice versa. (This interaction doesn't arise if B undergoes further refinement.) For the box marked s, its multipole expansion is rapidly convergent at each of the k^2 target points in B. Thus, its influence can be computed by direct evaluation of the truncated series. For the reverse, however, note that B's multipole expansion is not so rapidly convergent. In this case, we can map directly from the polynomial coefficients $\vec{c}_B$ of B to the local expansion in s. A more precise statement than (18) is

$$\Psi_{l,k} = \tilde{\Psi}_{l,k} + \sum_{i \in SIL} T_{ML}\Phi_i, + \sum_{i \in CIL} L_{direct}(\vec{c}_i), \tag{19}$$

where SIL denotes the standard interaction list and CIL denotes the coarse interaction list. The operator L_{direct}, which maps the coefficients of the polynomial approximation of the density in the coarse box onto the p coefficients of the local expansion can be precomputed and stored.

The bulk of the work in the FMM consists of applying the operators $\mathcal{T}_{MM}, \mathcal{T}_{LL}$, and especially $\mathcal{T}_{ML}$ in a systematic fashion. Unfortunately, these operators are dense. Using multipole and local expansions truncated after p terms, the naive cost of application is proportional to p^2. Recent improvements in the FMM have reduced this cost in both two and three space dimensions [17, 19]. A brief discussion of the technical ideas is presented in section 3.8.

3.4. The FMM algorithm.

Initialization

Comment [We assume we are given a square domain $D = S_{0,0}$, on which is superimposed an adaptive hierarchical quad-tree structure. We let M be the number of leaf nodes and denote them by D_i, $i = 1, \ldots, M$. The number of grid points is, therefore, $N = 16M$. We let p denote the order of the multipole expansion ($p \approx \log_2 \epsilon$, where ϵ is the desired accuracy). We let l_{max} denote the maximum refinement level.]

Step I: Multipole sweep

Upward pass

```
for $l = l_{max}, \ldots, 0$
    for all boxes $j$ on level $l$
        if $j$ is childless then
            form the multipole expansion $\Phi_{l,j}$ from (12)
        else
            form the multipole expansion $\Phi_{l,j}$ by merging the expansions of
            its children using the operator $\mathcal{T}_{MM}$ (see (16))
    end
end
```

Downward pass

```
Initialize the local expansion $\Psi_{0,0} = 0$.
for $l = 1, \ldots, l_{max}$
    for all squares $j$ on level $l$
        Compute $\tilde{\Psi}_{l,j}$ by shifting its parent's $\Psi$ expansion using the operator $\mathcal{T}_{LL}$
        Compute $\Psi_{l,j}$ by adding in the contributions from all squares in $j$'s
          interaction list according to (19).
        if $j$ is childless then
            for all boxes $k$ in the s-list of $j$:
                evaluate the multipole expansion $\Phi_k$ at each
                target in square $j$.
            end
            Evaluate the local expansion $\Psi_{l,j}$ at each
            target in square $j$.
        endif
    end
end
```

Cost [The upward pass requires approximately Mp^2 work, where M is the number of leaf nodes. The downward pass requires approximately $3Mp^2$ work using plane-wave expansions (see section 3.8 below).]

Step II: Local interactions

Comment [At this point, for each leaf node D_i, we have computed the influence of the source distribution f over all leaf nodes D_j outside the neighbors of D_i.]

```
do $i = 1, \ldots, M$
    For each target point in $D_i$, evaluate the influence of each
        neighbor according to (8) using the precomputed
        tables of coefficients (10).
end
```

Cost [The maximum number of neighbors a square can have is thirteen (twelve fine neighbors and itself). Thus the local work is bounded by is $13 \cdot \frac{k(k+1)}{2} \cdot N$ operations.]

3.5. Periodic boundary conditions. The inversion formula (2) and the fast algorithm described above assume that the right-hand side f is supported within a unit square. When imposing periodic boundary conditions, as mentioned in section 2, one can simply assume that the entire plane is tiled with copies of f centered at the lattice points $\{(i,j)|i,j \in \mathbf{Z}\}$. In order to account for the influence of these images, we follow the approach introduced in [16], the essence of which can already be found in

Lord Rayleigh's classic paper [25]. The main thing to notice is that, after the upward pass in the FMM, we have a net multipole expansion describing the far field due to the entire source distribution f contained in the unit cell centered at the origin:

$$\phi(\mathbf{x}) = \Re\left(\sum_{n=1}^{p} \frac{\alpha_n}{z^n}\right). \tag{20}$$

(There is no logarithmic term since we assume that the source distribution has no net charge.) This is then the expansion for each of the periodic images of the box with respect to its own center. All of these images, except for the nearest neighbors centered at {(-1,-1), (-1,0), (-1,1), (0,-1), (0,1), (1,-1), (1,0), (1,1)}, are well separated from the computational domain itself. Thus, the fields they induce inside the computational domain are accurately representable by a p-term local expansion where, as before, p is the number of terms needed to achieve a relative precision ϵ. This local representation can be written as

$$\Psi_{0,0}(w) = \Re\left(\sum_{n=0}^{p} \beta_n w^n\right). \tag{21}$$

It remains only to obtain the operator mapping the coefficients $\{\alpha_n\}$ to the coefficients $\{\beta_n\}$. We refer the reader to [4, 16, 25] for a discussion of this operator, which is based on the precomputation of certain lattice sums. The reason we denote the local expansion in (21) by $\Psi_{0,0}$ is for consistency of notation with the FMM described above; the *downward pass* is modified in the initialization step. In the remainder of the downward pass and in Step II, only two changes are required; the interaction list and the local computations must be adjusted for boxes near the boundary to account for periodic images. This involves no significant increase in the amount of work.

3.6. Other homogeneous boundary conditions. As noted in section 2, problems with homogeneous Dirichlet and Neumann conditions can be solved using the method of images. Since there is a 2 × 2 "supercell" which tiles the plane periodically, it is straightforward to embed such problems in a periodic version of the FMM. Done naively, this would entail a fourfold increase in CPU time and storage. Careful implementation considerations allow one to recover this overhead, but the details are tedious and will be omitted.

3.7. Inhomogeneous boundary conditions. In order to impose inhomogeneous boundary conditions using potential theory, we need to consider arrangements of single and double layer potentials such as the one depicted in Figure 2. These can be viewed as singular charge distributions and can be handled by the same FMM as above, with three modifications. First, the far field due to a box B with a single layer density σ and a double layer density μ along its boundary Γ is given by

$$\phi(\mathbf{x}) = \Re\left(\alpha_0 \log|x_1 + ix_2 - C| + \sum_{k=1}^{\infty} \frac{\alpha_k}{(x_1 + ix_2 - C)^k}\right), \tag{22}$$

where

$$\alpha_0 = -\frac{1}{2\pi}\int_\Gamma \sigma(s)ds \tag{23}$$

and

$$\alpha_k = -\frac{1}{2\pi}\int_\Gamma \frac{(y_1(s) + iy_2(s) - C)^k \sigma(s)}{k} + (y_1(s) + iy_2(s) - C)^{k-1}\mu(s)\, ds. \tag{24}$$

Here, $(y_1(s), y_2(s))$ is an arclength parametrization of Γ. Second, contributions to the local field from a leaf node containing layer potentials are precomputed as in section 3.2. Finally, the interaction list and the local computations must be adjusted for boxes near the boundary.

3.8. Fast translation operators. Consider a box B centered at X_B, containing sources $\{z_1, \ldots, z_M\}$ with source strengths $\{q_1, \ldots, q_M\}$ and a target box D centered at X_D in its interaction list. We assume for the moment that $\Re(X_D) > \Re(X_B)$. In the original FMM, the field outside B is represented as

$$\phi(\mathbf{x}) = \Re\left(\alpha_0 \log(x_1 + ix_2 - X_B) + \sum_{k=1}^{p} \frac{\alpha_k}{(x_1 + ix_2 - X_B)^k}\right). \tag{25}$$

The field inside D is represented as

$$\phi(\mathbf{x}) = \Re\left(\sum_{l=0}^{p} \beta_l \,(x_1 + ix_2 - X_D)^l\right) \tag{26}$$

with

$$\beta_0 = \alpha_0 \log(X_D - X_B) \;+\; \sum_{k=1}^{\infty} \frac{\alpha_k}{(X_D - X_B)^k}(-1)^k,$$

$$\beta_l = -\frac{\alpha_0}{l \cdot (X_D - X_B)^l} \;+\; \frac{1}{(X_D - X_B)^l} \sum_{k=1}^{\infty} \frac{\alpha_k}{(X_D - X_B)^k} \binom{l+k-1}{k-1} (-1)^k \quad \text{for } l \geq 1.$$

This describes the translation operator denoted by $\mathcal{T}_{ML}$ in section 3.3 and requires $O(p^2)$ work to apply. In [19], Hrycak and Rokhlin suggest an alternative representation of ϕ, based on the formula

$$\frac{1}{z - w} = \int_0^{\infty} e^{-\lambda(z-w)}\, d\lambda. \tag{27}$$

This integral can be discretized using generalized Gaussian quadratures [31] which take into account the nature of the integrand as well as the precise geometry of the interaction list. The number of quadrature nodes needed to achieve a precision ϵ is less than or equal to the number of multipole coefficients. Tables of weights and nodes for various values of ϵ are provided in [31]. For numerical purposes, we begin with an approximation of the form

$$\frac{1}{z_i - w} \approx \sum_{k=1}^{p} w_k e^{-\lambda_k (z_i - w)}.$$

Integrating both sides, we have

$$\log(z_i - w) \approx \sum_{k=1}^{p} \frac{-w_k}{\lambda_k} e^{-\lambda_k (z_i - w)} + C,$$

where C is a constant of integration. Choosing

$$C = \sum_{k=1}^{p} \frac{w_k}{\lambda_k} e^{-\lambda_k}$$

enforces the condition that $\log(1) = 0$.

Instead of the classical multipole expansion (25), we instead work with the *exponential representation*

$$\phi(\mathbf{x}) = \Re\left(\sum_{k=1}^{p} a_k e^{-\lambda_k\,(x_1+ix_2-X_B)} + C'\right), \tag{28}$$

where the coefficients a_k are exponential moments of the charge distribution:

$$a_k = \sum_{j=1}^{M} q_j e^{+\lambda_k\,(z_j-X_B)}$$

and

$$C' = \left(\sum_{j=1}^{M} q_j\right) \cdot C.$$

The advantage of this approach is that translation has been diagonalized. Transmitting the expansion from box B to D is carried out by computing

$$\psi(\mathbf{x}) = \Re\left(\sum_{k=1}^{p} b_k e^{-\lambda_k\,(x_1+ix_2-X_D)} + C'\right), \tag{29}$$

where the new coefficients b_k are obtained from the a_k through the translation formula

$$b_k = a_k\, e^{-\lambda_k(X_D-X_B)}.$$

The details of how to incorporate such expansions into an adaptive two-dimensional FMM code can be found in [19]. For the three-dimensional analogue, see [17].

4. Numerical results. Fast Poisson solvers using the algorithms described above have been implemented in Fortran 77. Here, we demonstrate their performance on four problems involving varying degrees of adaptivity. All of the timings listed below correspond to calculations performed on a 440MHz SUN Ultra-10 with 256 MB RAM using the compiler option (-fast).

There are few efficient adaptive solvers which are widely available. Therefore, we have chosen a simple and stringent standard for comparison: the time taken by a classical FFT-based code for the same number of degrees of freedom (grid points). Using the second order accurate FORTRAN code HWSCRT by Swartztrauber [27] and Swartztrauber and Sweet [28] (available from www.netlib.org), with the same machine and compiler option as above, we obtain the data shown in Table 1.

We have, as yet, said little about our adaptive refinement strategy. It is straightforward. Let B be a leaf node with $k \times k$ grid points, as discussed in section 3.2 and let $f_B(\mathbf{x})$ denote the kth order polynomial used to approximate the right-hand side on B. We then evaluate $f_B(x, y)$ on a $2k \times 2k$ grid covering B and compute the discrete L_2 error $E_2 = \|f(x, y) - f_B(x, y)\|_2$ over these target points. If $E_2 > tol$, the leaf node B is subdivided. Of course, the tree obtained by this procedure may not satisfy the level restriction that neighboring leaf nodes be at most one level apart. It is a straightforward matter to "fix" the tree in a subsequent sweep. We omit the details.

Table 1

Timing results for the FFT-based second order accurate code HWSCRT. N denotes the number of grid points, T_{hwscrt} denotes the required solution time in seconds, and rate denote the number of grid points "processed" per second (N/T_{hwscrt}).

N	T_{hwscrt}	Rate
256×256	0.17	$3.8\ 10^5$
512×512	0.78	$3.4\ 10^5$
1024×1024	4.0	$2.6\ 10^5$
2048×2048	19.4	$2.2\ 10^5$

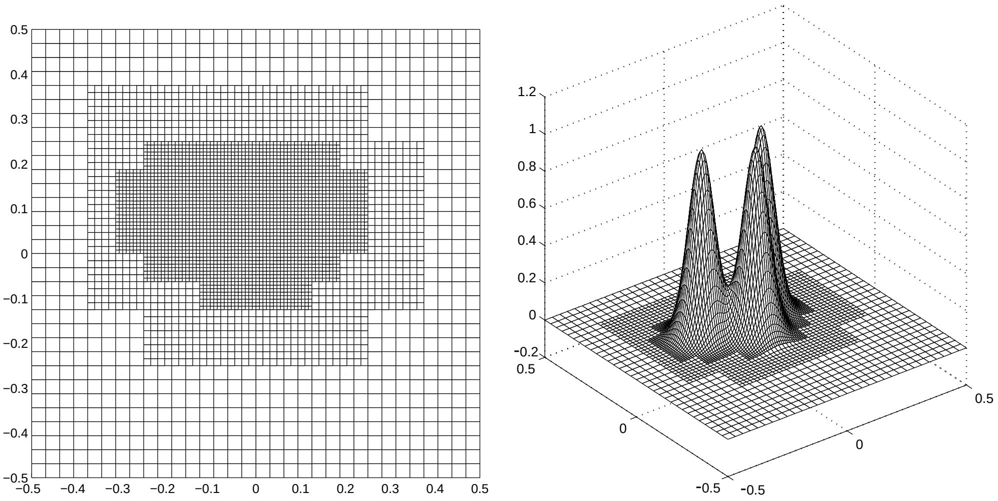


Fig. 4. *The left-hand figure shows an adaptive mesh resolving the source distribution in* (30). *The right-hand figure shows a surface plot of the solution.*

Example 4.1. In our first experiment, we consider the equation

$$\Delta\psi(\mathbf{x}) = \sum_{i=1}^{3} (4\alpha^2\ \|\mathbf{x} - \mathbf{x_i}\|^2 - 4\alpha) e^{-\alpha\|\mathbf{x}-\mathbf{x_i}\|^2} \tag{30}$$

in free space, for which the exact solution (Figure 4) is the sum of three Gaussians

$$\psi(\mathbf{x}) = \sum_{i=1}^{3} e^{-\alpha\|\mathbf{x}-\mathbf{x_i}\|^2}. \tag{31}$$

We consider the case where $\alpha = 250$, $\mathbf{x_1} = (.1, .1)$, $\mathbf{x_2} = (0, 0)$, and $\mathbf{x_3} = (-.15, .1)$. The right-hand side in (30) is supported, with an exponentially small error, in the box $[-0.5, 0.5]^2$, which we use as the computational domain. Our adaptive mesh is depicted in Figure 4. Note that fine grids are created only near the centers of the Gaussians. The performance of the fourth, sixth, and eighth order codes are summarized in Table 2.

Table 2

Timing results for the fourth, sixth, and eighth order accurate codes in Example 4.1. ϵ_{FMM} *denotes the requested precision from far-field interactions within the FMM,* ϵ_{RHS} *denotes the requested precision in discretizing the right-hand side, and* N_{lev} *denotes the number of levels used in the FMM hierarchy.* E_2 *and* E_∞ *denote the relative* L_2 *and* L_∞ *errors of the computed solution,* N *denotes the number of grid points used,* T_{FMM} *denotes the required solution time in seconds, and rate denote the number of grid points "processed" per second* (N/T_{FMM})*.*

ϵ_{FMM}	ϵ_{RHS}	N_{lev}	E_2	E_∞	N	T_{FMM}	Rate
Fourth order							
10^{-3}	10^{-3}	7	$3.7\ 10^{-4}$	$1.2\ 10^{-4}$	11488	0.08	$1.4\ 10^{5}$
10^{-3}	10^{-6}	9	$7.0\ 10^{-5}$	$1.4\ 10^{-4}$	96592	0.64	$1.5\ 10^{5}$
10^{-6}	10^{-6}	9	$4.9\ 10^{-6}$	$1.3\ 10^{-6}$	96592	1.08	$8.9\ 10^{4}$
10^{-6}	10^{-9}	10	$8.4\ 10^{-8}$	$4.9\ 10^{-7}$	821824	8.38	$9.8\ 10^{4}$
10^{-9}	10^{-9}	10	$1.4\ 10^{-8}$	$3.4\ 10^{-9}$	821824	12.17	$6.8\ 10^{4}$
Sixth order							
10^{-3}	10^{-3}	6	$8.2\ 10^{-5}$	$1.2\ 10^{-4}$	10296	0.08	$1.3\ 10^{5}$
10^{-3}	10^{-6}	7	$7.0\ 10^{-5}$	$1.6\ 10^{-4}$	43236	0.29	$1.5\ 10^{5}$
10^{-6}	10^{-6}	7	$1.5\ 10^{-7}$	$4.2\ 10^{-7}$	43236	0.39	$1.1\ 10^{5}$
10^{-6}	10^{-9}	9	$8.6\ 10^{-8}$	$5.6\ 10^{-7}$	279432	2.45	$1.1\ 10^{5}$
10^{-9}	10^{-9}	9	$2.4\ 10^{-9}$	$2.2\ 10^{-9}$	279432	3.48	$8.0\ 10^{4}$
10^{-9}	10^{-12}	10	$2.3\ 10^{-10}$	$2.4\ 10^{-9}$	1725984	17.19	$1.0\ 10^{5}$
10^{-12}	10^{-12}	10	$2.0\ 10^{-12}$	$8.4\ 10^{-13}$	1725984	27.28	$6.3\ 10^{4}$
Eighth order							
10^{-3}	10^{-3}	6	$1.0\ 10^{-4}$	$2.0\ 10^{-4}$	13888	0.16	$8.7\ 10^{4}$
10^{-3}	10^{-6}	7	$9.0\ 10^{-5}$	$2.0\ 10^{-4}$	63616	0.68	$9.4\ 10^{4}$
10^{-6}	10^{-6}	7	$1.7\ 10^{-7}$	$6.8\ 10^{-7}$	63616	0.80	$8.0\ 10^{4}$
10^{-6}	10^{-9}	8	$1.4\ 10^{-7}$	$6.8\ 10^{-7}$	273280	3.11	$8.8\ 10^{4}$
10^{-9}	10^{-9}	8	$4.4\ 10^{-10}$	$2.7\ 10^{-9}$	273280	3.62	$7.5\ 10^{4}$
10^{-9}	10^{-12}	9	$4.2\ 10^{-10}$	$2.8\ 10^{-9}$	1281472	16.02	$8.4\ 10^{4}$
10^{-12}	10^{-12}	9	$9.2\ 10^{-13}$	$6.1\ 10^{-13}$	1281472	21.68	$5.9\ 10^{4}$

Example 4.2. For our second experiment, we consider the singular equation

$$\begin{aligned} \Delta\psi &= 0 \quad \text{in } D, \\ \psi &= 0 \quad \text{on } \Gamma_L, \\ \psi &= 0 \quad \text{on } \Gamma_R, \\ \psi &= 1 \quad \text{on } \Gamma_T, \\ \psi &= 0 \quad \text{on } \Gamma_B, \end{aligned} \tag{32}$$

In Figure 5, we plot the solution obtained with our solver on an adaptive grid, the solution obtained by HWSCRT on a uniform 64 × 64 mesh, and the error in the HWSCRT solution. Note that we resolve the corner singularities adaptively and that our solution is exact (up to the requested FMM tolerance), since the data is piecewise polynomial (here, constant).

Remark 4.1. There is an enormous difference in the meaning of "order of accuracy" in our solver and in standard finite difference or finite element codes. Our solver is *exact* for a certain order of approximation of the data. A kth order accurate PDE-based solver on an $N \times N$ mesh has a global error which decays like $(1/N)^k$ with a constant of proportionality which depends on the kth derivative of the solution. In the present example, our solver is exact. The finite difference code is only first order

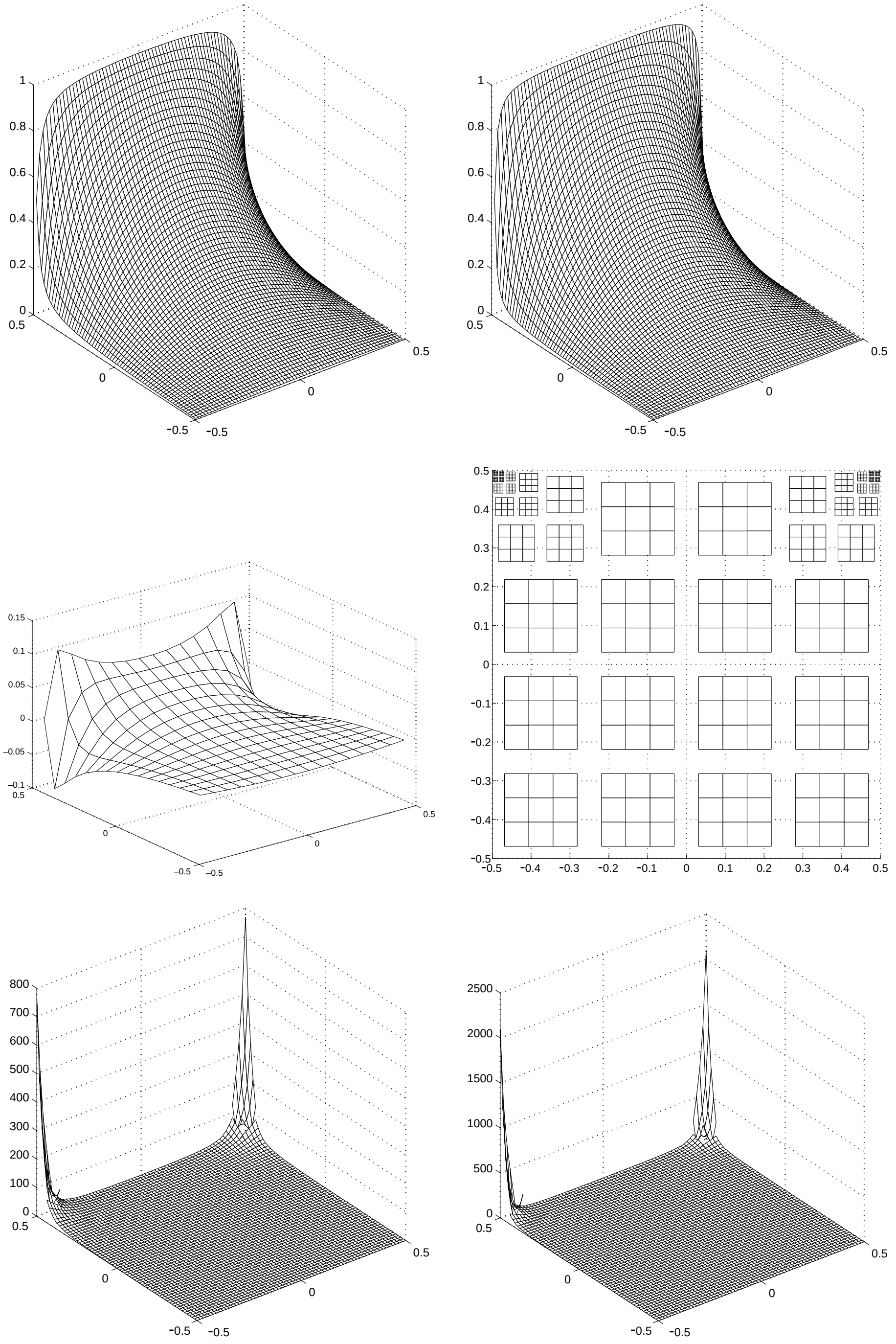


FIG. 5. *The upper left-hand figure shows the solution obtained by HWSCRT to Example* 4.2 *and the upper right-hand figure shows the solution obtained using our solver. The middle figures show the error in the solution obtained by HWSCRT and the adaptive grid use by our solver, respectively. The lower figures show the electrostatic energy* $\|\nabla\psi\|^2$ *obtained from HWSCRT and our solver, respectively.*

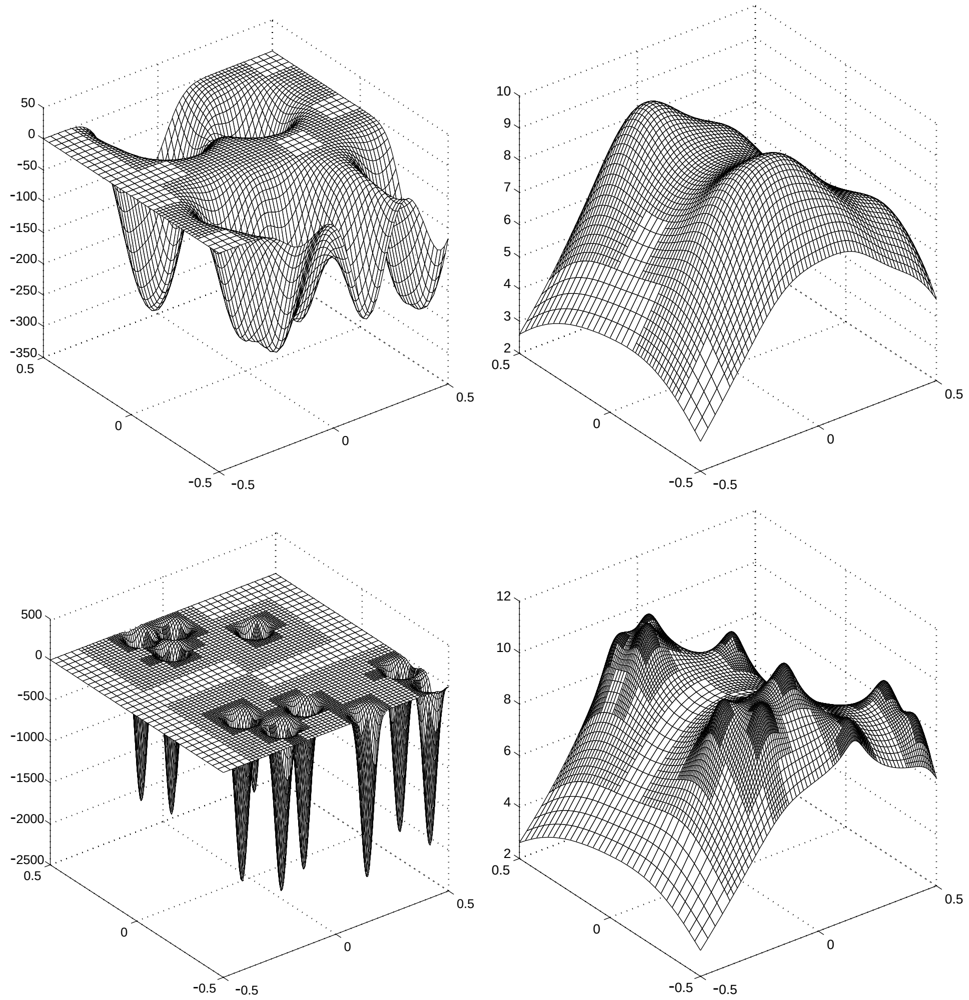


FIG. 6. *The left-most figure shows a surface plot of the right-hand side for* 10 *randomly placed Gaussians as described in* (33) *with* $\alpha = 100$ *and the figure to the right of it shows the corresponding solution. The two lower figures show both the right-hand side and corresponding solution for* $\alpha =$ 1000*.*

accurate because of the corner singularities.

Example 4.3. In order to evaluate the performance of our code with widely varying degrees of adaptivity, we consider the source distribution

$$\Delta\psi(\mathbf{x}) = \sum_{i=1}^{10} -2\alpha e^{-\alpha\|\mathbf{x}-\mathbf{x_i}\|^2} \tag{33}$$

in free space. With $\alpha = 100$, the distribution is fairly smooth, while with $\alpha = 1000$, the Gaussians fall off sharply and require many levels of refinement near the centers (see Figure 6). The experiment was run using the fourth order code and $\epsilon_{FMM} = \epsilon_{RHS} = 10^{-6}$. Table 3 lists our timing data for various values of α.

In addition to comparing various levels of adaptivity, it is also worth noting that

Table 3
Timings for the FMM-based solver in Example 4.3.

M	N	T_{FMM}	Rate	N	T_{FMM}	Rate
	$\alpha = 100$			$\alpha = 250$		
5	43969	0.44	9.9 10^4	73840	0.76	9.7 10^4
10	70864	0.68	1.0 10^5	138640	1.43	9.7 10^4
25	119296	1.33	8.9 10^4	212800	2.06	1.0 10^5
100	233296	2.43	9.6 10^4	262144	2.60	1.0 10^5
	$\alpha = 500$			$\alpha = 1000$		
5	97648	1.08	9.0 10^4	108544	1.17	9.3 10^4
10	196336	2.09	9.4 10^4	225616	2.40	9.4 10^4
25	377248	4.03	9.4 10^4	460912	4.96	9.3 10^4
100	785632	8.39	9.4 10^4	916336	9.44	9.7 10^4

Table 4
Timings for the FMM-based solver in Example 4.3 *using free-space and periodic boundary conditions.*

M	N	T_{FMM}	Rate	N	T_{FMM}	Rate
	Free space			Periodic		
10	138640	1.43	9.7 10^4	139696	1.63	8.6 10^4
30	241168	2.38	1.0 10^5	241552	2.63	9.2 10^4
50	253696	2.41	1.0 10^5	253840	2.62	9.7 10^4
100	262144	2.60	1.0 10^5	262192	2.94	8.9 10^4

the periodic and free-space solvers execute in nearly the same time. To show this, we consider a right-hand side given by

$$\Delta\psi(\mathbf{x}) = \sum_{i=1}^{M} (-1)^i 2\alpha e^{-\alpha\|\mathbf{x}-\mathbf{x_i}\|^2}. \tag{34}$$

This ensures that the net charge in the periodic cell is zero. Table 4 compares the performance of the fourth order accurate code with either free-space or periodic boundary conditions with $\alpha = 250$ and $\epsilon_{FMM} = \epsilon_{RHS} = 10^{-6}$.

Example 4.4. Most of the preceding examples involve adaptive refinement around a "point-like" singularity. Here we consider the case of a singularity along a curve. We simply define a source distribution which takes the value 1 inside a circle of radius .25 and 0 outside. The right-hand side is shown in Figure 7 along with the computed solution. Using the fourth order solver with 7 levels or refinement and 7936 grid points, the L_2 and L_∞ errors are less than 10^{-3} and the solver executes at the rate 1.3 10^5 points per second. Using the sixth order solver with 12 levels or refinement and 681,408 grid points, the L_2 and L_∞ errors are less than 10^{-6} and the solver executes at the rate 8.5 10^4 points per second. Both of these timings are comparable to those in Example 4.1.

The following observations can be made from the preceding data.

1. The timings for the FMM-based solver grow linearly with the number of unknowns. For fourth order accuracy with a three-digit FMM tolerance, the present implementation achieves a processing speed between 5.9 10^4 and 1.5 10^5 points per second. The classical second order FFT-based solver processes 2.6 10^5 points per second on a 1024 × 1024 grid.
2. For three-digit accuracy, the fourth order accurate code is fastest ($\approx$ 1.4 10^5 points per second), while for six-digit accuracy, the sixth order accurate code

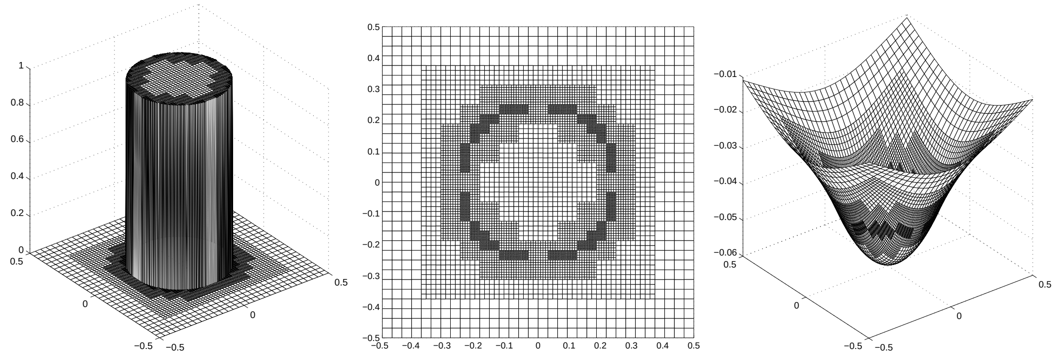


Fig. 7. *The left-hand figure shows the right-hand side. The middle figure shows the grid and the right figure shows the solution.*

is fastest ($\approx 1.1\ 10^5$ points per second). For twelve-digit accuracy, the sixth and eighth order codes are only about twice as slow ($\approx 6.3\ 10^4$ points per second).

5. Conclusions. We have developed a new adaptive, high order accurate solver for the Poisson equation in two dimensions. The method is direct, fast-multipole-based, and allows for the specification of a variety of boundary conditions on a unit square. These include free-space conditions, periodic boundary conditions, Dirichlet, Neumann, and a variety of mixed conditions. The amount of work scales linearly with the number of degrees of freedom in the computational domain and is competitive with classical FFT-based solvers in terms of *work per grid point*, despite the flexibility of adaptive mesh refinement.

In order to develop a black box Poisson solver of broad interest, of course, we need to allow for complex geometry. It would also be of value to be able to solve the Helmholtz and linearized Poisson–Boltzmann equations,

$$\Delta u + \lambda^2 u = f \qquad \text{and} \qquad \Delta u - \lambda^2 u = f,$$

with a similar approach. These extensions are underway and will be reported at a later date.

REFERENCES

[1] A. S. Almgren, J. B. Bell, P. Colella, and L. H. Howell, *An adaptive projection method for the incompressible Euler equation*, in Proceedings of the Eleventh AIAA Computational Fluid Dynamics Conference, 1993, pp. 530–539.

[2] C. Anderson, *Domain decomposition techniques and the solution of Poisson's equation in infinite domains*, in Proceedings of the Second International Symposium on Domain Decomposition Methods, 1988, pp. 129–139.

[3] M. J. Berger and P. Colella, *Local adaptive mesh refinement for shock hydrodynamics*, J. Comput. Phys., 53 (1989), pp. 484–512.

[4] C. L. Berman and L. Greengard, *A renormalization method for the evaluation of lattice sums*, J. Math. Phys., 35 (1994), pp. 6036–6048.

[5] A. Brandt, *Multi-level adaptive solutions to boundary value problems*, Math. Comp., 31 (1977), pp. 330–390.

[6] B. L. Buzbee, G. H. Golub, and C. W. Nielson, *On direct methods for solving Poisson's equations*, SIAM J. Numer. Anal., 7 (1970), pp. 627–656.

[7] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang, *Spectral Methods in Fluid Dynamics*, Springer-Verlag, New York, 1988.

[8] J. Carrier, L. Greengard, and V. Rokhlin, *A fast adaptive multipole algorithm for particle simulations*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 669–686.

[9] T. F. Chan, R. Glowinski, J. Periaux, and O. B. Widlund, eds., *Domain Decomposition Methods*, SIAM, Philadelphia, 1989.

[10] T. Chan and B. Smith, *Domain decomposition and multigrid algorithms for elliptic problems on unstructured meshes*, Electron. Trans. Numer. Anal., 2 (1994), pp. 171–182.

[11] G. Chesshire and W. D. Henshaw, *Composite overlapping meshes for the solution of partial differential equations*, J. Comput. Phys., 90 (1991), pp. 1–64.

[12] Z. Gimbutas, L. Greengard, and M. Minion, *Coulomb interactions on planar structures: Inverting the square root of the Laplacian*, SIAM J. Sci. Comput., 22 (2001), pp. 2093–2108.

[13] D. Gottlieb and S. A. Orszag, *Numerical Analysis of Spectral Methods: Theory and Applications*, SIAM, Philadelphia, 1977.

[14] L. Greengard, *The Rapid Evaluation of Potential Fields in Particle Systems*, MIT Press, Cambridge, MA, 1988.

[15] L. Greengard and J. Y. Lee, A direct adaptive Poisson solver of arbitrary order accuracy, J. Comput. Phys., 125 (1996), pp. 415–424.

[16] L. Greengard and V. Rokhlin, *A fast algorithm for particle simulations*, J. Comput. Phys., 73 (1987), pp. 325–348.

[17] L. Greengard and V. Rokhlin, *A new version of the fast multipole method for the Laplace equation in three dimensions*, Acta Numer., 6 (1997), pp. 229–269.

[18] R. B. Guenther and J. W. Lee, *Partial Differential Equations of Mathematical Physics and Integral Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1988.

[19] T. Hrycak and V. Rokhlin, *An improved fast multipole algorithm for potential fields*, SIAM J. Sci. Comput., 19 (1998), pp. 1804–1826.

[20] H. Johansen and P. Colella, *A Cartesian grid embedded boundary method for Poisson's equation on irregular domains*, J. Comput. Phys., 147, (1998), pp. 60–85.

[21] S. McCormick, ed., *Multigrid Methods*, SIAM, Philadelphia, 1987.

[22] A. McKenney, L. Greengard, and A. Mayo, *A fast Poisson solver for complex geometries*, J. Comput. Phys., 118 (1995), pp. 348–355.

[23] M. L. Minion, *A projection method for locally refined grids*, J. Comput. Phys., 127 (1996), pp. 158–177.

[24] A. T. Patera, *A spectral element method for fluid dynamics: Laminar flow in a fluid expansion*, J. Comput. Phys., 54 (1984), pp. 468–488.

[25] Lord Rayleigh, *On the influence of obstacles arranged in a rectangular order upon the properties of the medium*, Philos. Mag., 34 (1892), p. 481.

[26] V. Rokhlin, *Rapid solution of integral equations of classical potential theory*, J. Comput. Phys., 60 (1985), pp. 187–207.

[27] P. N. Swarztrauber, *The methods of cyclic reduction, Fourier analysis and the FACR algorithm for the discrete solution of Poisson's equation on a rectangle*, J. Comput. Phys., 15 (1974), pp. 46–54.

[28] P. Swarztrauber and R. Sweet, *Efficient Fortran Subprograms for the Solution of Elliptic Partial Differential Equations*, NCAR Technical Note NCAR-TN/IA-109, 1975, pp. 135–137.

[29] G. Russo and J. Strain, *Fast triangulated vortex methods for the 2D Euler equations*, J. Comput. Phys., 111 (1994), pp. 291–323.

[30] I. Stakgold, *Boundary Value Problems of Mathematical Physics*, Macmillan, New York, 1968.

[31] N. Yarvin and V. Rokhlin, *Generalized Gaussian quadratures and singular value decompositions of integral operators*, SIAM J. Sci. Comput., 20 (1999), pp. 699–718.

SIAM J. SCI. COMPUT.
Vol. 23, No. 3, pp. 761–780

QMR-BASED PROJECTION TECHNIQUES FOR THE SOLUTION OF NON-HERMITIAN SYSTEMS WITH MULTIPLE RIGHT-HAND SIDES[*]

MISHA KILMER[†], ERIC MILLER[‡], AND CAREY RAPPAPORT[‡]

Abstract. In this work we consider the simultaneous solution of large linear systems of the form $Ax^{(j)} = b^{(j)}, j = 1, \ldots, K$, where A is sparse and non-Hermitian. We describe single-seed and block-seed projection approaches to these multiple right-hand side problems that are based on the QMR and block QMR algorithms, respectively. We use (block) QMR to solve the (block) seed system and generate the relevant biorthogonal subspaces. Approximate solutions to the nonseed systems are simultaneously generated by minimizing their appropriately projected (block) residuals. After the initial (block) seed has converged, the process is repeated by choosing a new (block) seed from among the remaining nonconverged systems and using the previously generated approximate solutions as initial guesses for the new seed and nonseed systems. We give theory for the single-seed case that helps explain the convergence behavior under certain conditions. Implementation details for both the single-seed and block-seed algorithms are discussed and advantages of the block-seed algorithm in cache-based serial and parallel environments are noted. The computational savings of our methods over using QMR to solve each system individually are illustrated in two examples.

Key words. QMR, projection, Krylov subspace, iterative methods, block Krylov

AMS subject classifications. 65F10, 65N22

PII. S1064827599355542

1. Introduction.

In many applications one desires the solution of multiple linear systems of the form

$$Ax^{(j)} = b^{(j)}, \quad j = 1, \ldots, K, \tag{1.1}$$

involving the same $N \times N$ coefficient matrix A but K different right-hand sides $b^{(j)}$, all of which are available simultaneously. Such problems arise, for instance, in the numerical solution of frequency-domain electromagnetic wave scattering; here, the right-hand sides might correspond to incident fields over the scatterer induced either by plane waves at various angles of incidence or by excitation sources at different locations.

Systems involving large, sparse matrices make good candidates for solution by iterative Krylov subspace methods since storage is kept to a minimum and matrix-vector products can be done efficiently. However, the naive approach of solving each of the K linear systems independently using a Krylov subspace method does not take advantage of the fact that the $b^{(j)}$'s, and hence the $x^{(j)}$'s, may be *closely related* due to the underlying physical nature of the problem. By closely related, we mean that the solution to the jth system has large components in the initial directions of the k-dimensional ($k \ll N$) Krylov subspace generated from one of the other systems. Projection-type techniques for both the Hermitian and non-Hermitian cases,

[*] Received by the editors May 4, 1999; accepted for publication (in revised form) September 29, 2000; published electronically August 15, 2001. This work was supported by the Army Research Office Demining MURI under grant DAAG55-97-1-0013.

http://www.siam.org/journals/sisc/23-3/35554.html

[†] Department of Mathematics, Tufts University, Medford, MA 02155 (mkilme01@tufts.edu).

[‡] Electrical and Computer Engineering Department, Northeastern University, Boston, MA 02115 (elmiller@ece.neu.edu, rappaport@neu.edu).

discussed in more detail below, that specifically exploit such shared information have been proposed (e.g., see [5, 26, 24] and the references therein).

Another alternative is to use a block Krylov subspace algorithm to solve the systems simultaneously [16, 23, 4, 8, 15]. Essentially these methods seek solutions in block Krylov subspaces, or some deflated version thereof, generated by the matrix A and the $N \times K$ matrix $R = B - AX_0$; here the columns of B are the $b^{(j)}$ and the columns of X_0 are the initial estimates for each of the systems. However, this approach can be more expensive in terms of storage than projection techniques because the length of the recurrences to update the iterates depends on the number of right-hand sides, or, in the case of deflation [8], the number of right-hand sides corresponding to the deflated Krylov sequences. Also, if a deflation technique is used, a deflation tolerance must be specified in advance, and we have found in experiments that the performance and convergence of the systems depend in a somewhat unpredictable manner on this value. On the other hand, block Krylov subspace algorithms have the advantage that they are better suited to parallelism [17, 14, 15] and make use of higher level BLAS [2]. Therefore, in this paper we develop new single and block-seed projection approaches based on the QMR and block QMR algorithms, respectively; our block-seed method exploits the best properties of the block QMR algorithm while preserving the basic properties of our sequential projection technique. To our knowledge, ours is the first block-seed projection algorithm for non-Hermitian linear systems with multiple right-hand sides; in particular, as our algorithm is built around the BL-QMR algorithm of [8], it incorporates a deflation strategy.

Specifically, the idea of a projection technique is to first select one of the systems as “seed” and solve it by an iterative Krylov subspace method. As the relevant subspaces are generated, the approximations to the other systems are simultaneously updated by projecting the residual onto a particular subspace and by either enforcing a Galerkin-type condition [11, 25] or minimizing the projected residual [24]. Such methods are sometimes referred to as Lanczos–Galerkin approaches [21].

Smith [25] and Joly [11] both consider a projection approach based on BiCG for nonsymmetric A. In [11] a similar approach for CGS is also given. However, the BiCG-projection approach can exhibit the potentially erratic convergence behavior observed when applying BiCG to a single linear system (see the results in [24]). Simoncini and Gallopoulos [24] also present an approach to solving (1.1) when A is nonsymmetric.

Our single-seed projection algorithm is most similar in concept to the project-minimize approach in [24]. However, as a result of the underlying unsymmetric Lanczos process, we do not need to store the basis vectors, we do not need to predetermine a subspace dimension, and we show that the approximate solutions and residuals to the nonseed systems are cheaply computed and available at every stage of the algorithm because they are updated with short-term recurrences. Since we are minimizing quantities rather than enforcing Galerkin conditions, the convergence behavior should be less erratic than methods based on the latter. As noted and as we illustrate in Theorem 3.4, the success of our single-seed method depends on the closeness, in the sense described above, among the right-hand sides. The block-seed approach we introduce here is more efficient when the right-hand sides are not all close.

This paper is organized as follows. In section 2, we give the necessary background on the QMR approach. We give an outline of our single-seed projection approach and discuss its convergence in section 3. Background on the block QMR algorithm is presented in section 4. The block variant of our QMR-projection algorithm is reported in section 5, and related computational issues are discussed in section 6. Section 7 gives numerical results, and section 8 reports conclusions and future work.

2. Background: The QMR algorithm.

The quasi-minimal residual (QMR) algorithm was introduced by Freund and Nachtigal in [9]. The original algorithm was based on three-term recurrences. In [10], the authors proposed a mathematically equivalent algorithm which employed coupled two-term recurrences. Since the latter variant has been found to be more numerically stable for solving linear systems, in numerical experiments we use this implementation. However, to simplify the notation in this section and in section 3, and to be consistent with the notation in section 5, we will follow the notation in [9]. Further, for simplicity, we consider only the version without look-ahead but note that our algorithm could be adapted to account for look-ahead.

In the remainder of the paper, the notation $\|\cdot\|$ always refers to the Euclidean norm $\|\cdot\|_2$. The superscript T is used to denote the transpose operation, and superscript $*$ is used to denote the conjugate transpose operation.

A Krylov subspace of dimension k generated by a matrix G and a vector q is defined according to

$$K_k(G, q) = \text{span}\{q, Gq, G^2q, \ldots, G^{k-1}q\}.$$

The QMR algorithm is a Krylov-subspace-based iterative method that can be used to solve non-Hermitian linear systems of the form $Ax = b, \;\; A \in \mathcal{C}^{N \times N}$. At the kth iteration, the current solution estimate has the form

$$x_k = x_0 + V_k z_k, \tag{2.1}$$

where x_0 denotes the initial guess and $V_k = [v_1, v_2, \ldots, v_k]$ is an $N \times k$ matrix whose columns are basis vectors for $K_k(A, v_1)$ with $v_1 = r_0/\|r_0\|$ and $r_0 = b - Ax_0$. The length k vector z_k is chosen as the solution to a particular minimization problem, as discussed below. Those basis vectors are generated via the nonsymmetric Lanczos process (see [22]) and are constructed to be biorthogonal to vectors $w_i, i = 1, \ldots, k$, which form a basis for $K_k(A^T, w_1)$.[1] The columns of the $N \times k$ matrix W_k are the w_i.

From biorthogonality it follows that

$$W_{k+1}^T V_{k+1} = D_{k+1}, \;\; D_{k+1} = \text{diag}(\delta_1, \ldots, \delta_{k+1}), \qquad \delta_i \neq 0. \tag{2.2}$$

Also as a result of the Lanczos algorithm we obtain the relation

$$AV_k = V_{k+1}\bar{T}_k, \tag{2.3}$$

where $\bar{T}_k$ is a $(k+1) \times k$ tridiagonal matrix. Using (2.1), (2.2), and (2.3), and setting $\beta = \|r_0\|$, at the kth iteration the residual, $r_k = b - Ax_k$, is given by [22]

$$r_k = V_{k+1}(\beta e_1 - \bar{T}_k z_k), \tag{2.4}$$

where e_1 denotes the first Cartesian unit vector. Since the columns of V_{k+1} are not orthonormal,

$$\|r_k\| \leq \|V_{k+1}\| \|\beta e_1 - \bar{T}_k z_k\|.$$

[1] Here we always take $w_1 \equiv v_1$, but note that other choices are possible. A version of the algorithm is also possible using $K_k(A^*, w_1)$ for the second Krylov subspace.

The QMR algorithm determines z_k by minimizing the norm of the quasi-residual term [9]; that is,

$$z_k = \arg \min_{z \in \mathcal{C}^{k \times 1}} \|\beta e_1 - \bar{T}_k z\|.$$

We make the following alternate observation. From (2.4) and using biorthogonality,

$$\|D_{k+1}^{-1} W_{k+1}^T r_k\| = \|\beta e_1 - \bar{T}_k z_k\|.$$

Thus, the z_k which defines the kth QMR iterate can also be thought of as the one that minimizes the norm of the residual projected onto a smaller dimensional subspace. We will make use of this alternate definition of the QMR iterates in subsequent sections.

3. The single-seed QMR-projection algorithm. In this section we describe a single-seed QMR-projection algorithm for solving linear systems of the form (1.1). Our algorithm proceeds as follows. First, we select one system, say, system j, to serve as "seed" and apply QMR (without look-ahead) to the seed system. In the following, we use $r_0^{j,l}$ to denote the initial residual to system l, where l denotes the index of any of the nonconverged systems given the starting guess $x_0^{j,l}$. We use $r_k^{j,l}$ to denote the residual of system l after k iterations. Since different choices of seed lead to different Krylov subspaces and hence different iterates, the superscript j is used to denote that this is the residual at the kth iteration for system l when system j was used as seed. By the beginning of the kth iterate, QMR has generated biorthogonal bases for two k-dimensional Krylov subspaces, $K_k(A, r_0^{j,j})$ and $K_k(A^T, r_0^{j,j})$. We denote the respective bases by the vectors $v_{j,i}$ and $w_{j,i}$, $i = 1, \ldots, k$: the subscript j is used to indicate that this particular set was generated using system j as seed. These vectors are the columns of the $N \times k$ matrices $V_{j,k}$ and $W_{j,k}$, respectively. The corresponding $(k+1) \times k$ tridiagonal matrix is denoted as $\bar{T}_{j,k}$ (compare to (2.3)). By the end of the kth iterate, QMR has also generated the unnormalized versions of the vectors $v_{j,k+1}$ and $w_{j,k+1}$ for use in the $(k+1)$st iteration.

Let us comment on the values of $x_0^{j,l}$. If we suppose that j was the seed system and converged after m steps and that the index of the next seed system is j^*, then we set $x_0^{j^*,l} = x_m^{j,l}$ for all indices l such that system l has not already converged.

In the previous section, we have seen that the kth iterate corresponding to the seed system is given by

$$x_k^{j,j} = x_0^{j,j} + V_{j,k} z_k^{j,j} \quad \text{with } z_k^{j,j} = \arg \min_{z \in \mathcal{C}^{k \times 1}} \|\beta e_1 - \bar{T}_{j,k} z\|.$$

Now we also want the kth iterate of the (nonconverged) nonseed system, say, system l, to lie in $x_0^{j,l} + K_k(A, r_0^{j,l})$, in other words,

$$x_k^{j,l} = x_0^{j,l} + V_{j,k} z_k^{j,l}, \quad l \neq j. \tag{3.1}$$

Next we must decide how to define $z_k^{j,l}$. From (2.3) (with $V_{j,k}$ in place of V_k) and (3.1),

$$\begin{aligned} r_k^{j,l} &= b^{(l)} - A x_k^{j,l} \\ &= r_0^{j,l} - V_{j,k+1} \bar{T}_{j,k} z_k^{j,l}. \end{aligned} \tag{3.2}$$

Therefore, using biorthogonality,

$$\|D_{j,k+1}^{-1} W_{j,k+1}^T r_k^{j,l}\| = \|D_{j,k+1}^{-1} W_{j,k+1}^T r_0^{j,l} - \bar{T}_{j,k} z_k^{j,l}\|.$$

Finally, we use the above equality to determine $z_k^{j,l}$:

$$z_k^{j,l} = \arg \min_{z \in \mathcal{C}^{k \times 1}} \|D_{j,k+1}^{-1} W_{j,k+1}^T r_0^{j,l} - \bar{T}_{j,k} z\|. \tag{3.3}$$

3.1. Computational issues. Let us sketch how to efficiently compute the iterates and residuals of the nonseed systems. More details can be found in [13]. As above, we will use the index l to denote an arbitrary nonseed system and j to denote the seed system.

Let the QR decomposition of $\bar{T}_{j,k}$ be

$$\bar{T}_{j,k} = Q_{j,k}^* \begin{bmatrix} R_{j,k} \\ 0 \end{bmatrix},$$

where $Q_{j,k}$ is a product of Givens rotations and $R_{j,k}$ is $k \times k$ upper triangular with upper bandwidth 2. It can be shown [13] that using the QR factorization to solve (3.3) results in the easily solved system

$$z_k^{j,l} = R_{j,k}^{-1} t_k^{j,l}, \tag{3.4}$$

where $t_k^{j,l}$ differs from $t_{k-1}^{j,l}$ only in its last entry, which we shall denote by $y_k^{j,l}$. The norm of the projected residual in (3.3) is given by the scalar $|\tau_{k+1}^{j,l}|$. Further, $\tau_{k+1}^{j,l}$ and $y_k^{j,l}$ can be updated from $\gamma_{k+1}^{j,l}$ and $\tau_k^{j,l}$ with $\delta_{j,k} \equiv w_{j,k}^T v_{j,k}$:

$$\begin{bmatrix} y_k^{j,l} \\ \tau_{k+1}^{j,l} \end{bmatrix} = \begin{bmatrix} c_{j,k} & s_{j,k} \\ -\bar{s}_{j,k} & c_{j,k} \end{bmatrix} \begin{bmatrix} \tau_k^{j,l} \\ \gamma_{k+1}^{j,l} \end{bmatrix} \quad \text{with } \gamma_{k+1}^{j,l} \equiv \frac{1}{\delta_{j,k+1}} w_{j,k+1}^T r_0^{j,l}, \tag{3.5}$$

where $c_{j,i} \in \mathcal{R},\ s_{j,i} \in \mathcal{C},\ c_{j,i}^2 + |s_{j,i}|^2 = 1$.

As in equation (4.8) of [9], we define $P_{j,k} = [p_{j,1}, p_{j,2}, \ldots, p_{j,k}] \equiv V_{j,k} R_{j,k}^{-1}$. Since $R_{j,k}$ is upper triangular with bandwidth 2, there is a short-term recurrence relation for the $p_{j,k}$ [9]. Using (3.1) and (3.4), the kth iterate of the lth system is given by

$$x_k^{j,l} = x_{k-1}^{j,l} + y_k^{j,l} p_{j,k}. \tag{3.6}$$

From this, we derive an iterative update for $r_k^{j,l}$ that does not require any additional matrix-vector products per iteration as follows.

LEMMA 3.1. *The residual at the kth iteration corresponding to the lth system is given by*

$$r_k^{j,l} = r_{k-1}^{j,l} - y_k^{j,l} f_{j,k}, \quad \textit{where} \quad f_{j,k} \equiv A p_{j,k},$$

and can be computed in $O(N)$ flops.

Proof. The proof follows from (3.6), the bandedness of $R_{j,k}$, and the definition of $p_{j,k}$ above. (A detailed proof can be found in [13].) □

3.2. Seed selection. Clearly, the success of our QMR-projection approach at reducing the total number of matrix-vector products needed to solve all the systems to the desired tolerance depends on which and in what order systems are selected as seed. We use the approach in [24]; namely, we choose the seed index j such that the norm of the residual of the corresponding system is larger than all the remaining nonconverged systems. Developing more informed selection heuristics remains a subject for future research.

3.3. Theory. Suppose that QMR has been run once and that the initial seed system has converged after m steps. Our algorithm proceeds by choosing another seed and using as its initial guess that solution obtained via projection as the first system was solved. One of the results of this section is that in exact arithmetic, assuming A is diagonalizable, if some of the right eigenvectors are well approximated by Ritz vectors corresponding to the first Krylov subspace generated, the rate of convergence of the second seed system behaves as if the corresponding part of the spectrum of A is cut off. The proof technique follows along the lines of the proof of Lemma 3.2 in [5]. $\kappa_2(\cdot)$ denotes the 2-norm condition number of the argument.

We assume A is diagonalizable with eigendecomposition $A = \hat{Z}\Lambda\hat{S}$, where $\hat{S} = \hat{Z}^{-1}$. Here $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_N)$. We use $\hat{z}_j$ to denote the jth column of $\hat{Z}$ and $\hat{s}_j^*$ to denote the jth row of $\hat{S}$. Without loss of generality, we may assume that $\|\hat{z}_j\| = 1$. After step m of QMR applied to the seed system, let $T_{1,m}U_{1,m} = U_{1,m}\Sigma_{1,m}$ be the eigendecomposition of $T_{1,m}$, where $T_{1,m}$ denotes the tridiagonal $m \times m$ leading submatrix of $\bar{T}_{1,m}$. Since QMR is built on top of the unsymmetric Lanczos process, in exact arithmetic if m is large enough, we expect some of the Ritz vectors given by the columns of $V_{1,m}U_{1,m}$ to be good approximations to, say, $n \leq m$ of the right eigenvectors[2] [1, 6, 7, 2] (assuming these eigenvectors are present in the starting vector for the first seed system). Under these definitions and assumptions, we have the following.

THEOREM 3.2. *Consider two systems $Ax^{(1)} = b^{(1)}$ and $Ax^{(2)} = b^{(2)}$ with A an $N \times N$ matrix. Let $x_0^{2,2}$ be the starting vector for the second system obtained via our projection approach after m steps of QMR have been run using the first system as seed; that is, $x_0^{2,2} = x_m^{1,2}$.*

Define $\mathcal{I}$ as the set of indices of the n right eigenvectors that are well approximated by $n \leq N$ of the m Ritz vectors $V_{1,m}U_{1,m}$. Define $\bar{x}_0^{2,2}$ such that $x^{(2)} - \bar{x}_0^{2,2}$ is the projection of $x^{(2)} - x_0^{2,2}$ on $\text{span}\{\hat{z}_j, j \notin \mathcal{I}\}$. Let $\bar{x}_i^{2,2}$ be the ith iterate of GMRES applied to system 2 with initial guess $\bar{x}_0^{2,2}$. Then for any i

$$\|b^{(2)} - Ax_i^{2,2}\| \leq \kappa_2(V_{2,i+1})(\|b^{(2)} - A\bar{x}_i^{2,2}\| + \delta),$$

where

$$\delta = \sum_{k \in \mathcal{I}} |\lambda_k \tilde{p}(\lambda_k)||\hat{s}_j^* P_m e_0^{1,2}|,$$

$\tilde{p}$ is a particular i-degree polynomial with constant term 1, $e_0^{1,2} \equiv x^{(2)} - x_0^{1,2}$, and $P_m \equiv I - V_{1,m}(\bar{T}_{1,m}^* \bar{T}_{1,m})^{-1}\bar{T}_{1,m} D_{1,m+1}^{-1} W_{1,m+1}^T A$ is a projector.

Proof. Since A is diagonalizable, $e_0^{2,2} \equiv x^{(2)} - x_0^{2,2} = \sum_{k=1}^{N} \phi_k \hat{z}_k$ for some expansion coefficients ϕ_k. Hence, by definition

$$b^{(2)} - Ax_0^{2,2} = \sum_{k=1}^{N} \phi_k \lambda_k \hat{z}_k, \tag{3.7}$$

$$b^{(2)} - A\bar{x}_0^{2,2} = \sum_{k \notin \mathcal{I}} \phi_k \lambda_k \hat{z}_k. \tag{3.8}$$

[2] We caution the reader that our notation is somewhat unconventional, as we use n simply to denote an index $n \leq N$ and N to denote the dimension of the matrix.

Now if $\bar{x}_i^{2,2}$ is the ith GMRES iterate with $\bar{x}_0^{2,2}$ as the initial guess, there exists a polynomial $\tilde{p}_i$ of degree less than or equal to i with $\tilde{p}_i(0) = 1$ such that

$$b^{(2)} - A\bar{x}_i^{2,2} = \sum_{k \notin \mathcal{I}} \phi_k \lambda_k \tilde{p}_i(\lambda_k) \hat{z}_k, \tag{3.9}$$

where $\tilde{p}_i$ satisfies

$$\tilde{p}_i = \arg\min_{p \in \bar{\Pi}^i} \|p(A)(b^{(2)} - A\bar{x}_0^{2,2})\|. \tag{3.10}$$

Here, $\bar{\Pi}^i$ denotes the set of all polynomials with degree less than or equal to i with constant term 1. From Theorem 7.1 in [22], we have a bound on the ith QMR residual in terms of the ith GMRES residual:

$$\|b^{(2)} - Ax_i^{2,2}\| \le \kappa_2(V_{2,i+1}) \min_{p \in \bar{\Pi}^i} \|p(A)(b^{(2)} - Ax_0^{2,2})\|. \tag{3.11}$$

Also,

$$\min_{p \in \bar{\Pi}^i} \|p(A)(b^{(2)} - Ax_0^{2,2})\| \le \|\tilde{p}_i(A)(b^{(2)} - Ax_0^{2,2})\| = \|\sum_{k=1}^{N} \phi_k \lambda_k \tilde{p}_i(\lambda_k) \hat{z}_k\|. \tag{3.12}$$

Substituting this into (3.11) and using (3.9) and the triangle inequality gives

$$\|b^{(2)} - Ax_i^{2,2}\| \le \kappa_2(V_{2,i+1}) \left(\|b^{(2)} - A\bar{x}_i^{2,2}\| + \sum_{k \in \mathcal{I}} |\lambda_k \tilde{p}_i(\lambda_k)||\phi_k| \right). \tag{3.13}$$

Using the definition of $x_0^{2,2}$ as $x_m^{1,2}$, it is straightforward to show

$$e_0^{2,2} = (I - V_{1,m}(\bar{T}_{1,m}^* \bar{T}_{1,m})^{-1} \bar{T}_{1,m}^* D_{1,m+1}^{-1} W_{1,m+1}^T A) e_0^{1,2} = P_m e_0^{1,2}.$$

It is easy to see that P_m is a projector since $(P_m)^2 = P_m$. By the definition of $e_0^{2,2}$ it follows that $P_m e_0^{1,2} = \sum_{k=1}^N \phi_k \hat{z}_k$. Using $\hat{s}_j^* \hat{z}_k = 1$ if $j = k$ and 0 otherwise leads to

$$|\hat{s}_j^* P_m e_0^{1,2}| = |\phi_j| \quad j \in \mathcal{I}. \tag{3.14}$$

Substituting (3.14) into (3.13), we obtain the desired result. □

Now let us discuss why we expect δ to be small. First, it is clear that in exact arithmetic the Ritz vectors lie in $\mathcal{N}(P_m)$. For any vector $q \in \mathcal{C}^N$, since Z is full rank we may write $q = \hat{Z}c$ for the vector of expansion coefficients $c = \hat{Z}^{-1}q = \hat{S}q$ with components $c_j = \hat{s}_j^* q$. Now suppose $q \in \mathcal{R}(P_m)$. Since the columns of Z with indices in $\mathcal{I}$ are approximated by n of the m Ritz vectors, by assumption, and since the Ritz vectors are in $\mathcal{N}(P_m)$, this implies $c_j \approx 0, j \in \mathcal{I}$, which by definition means $\hat{s}_j^* q \approx 0, j \in \mathcal{I}$. Therefore, with $q \equiv P_m e_0^{1,2}$, $|\phi_j| \approx 0, j \in \mathcal{I}$, so δ should be small in exact arithmetic. It is clear that the quality of the Ritz vector approximation and loss of biorthogonality (e.g., the actual $\mathcal{N}(P_m)$) will affect the size of δ in practice.

Using the definition of $\tilde{p}_i$ in (3.10), we obtain in analogy with the standard GMRES upper bound for diagonalizable matrices [22] the following corollary.

Corollary 3.3. *Let $\hat{Z}_n$ denote the $N \times (N-n)$ matrix with columns $\hat{z}_j$ for $j \notin \mathcal{I}$. Then with δ defined as in Theorem* 3.2 *and P being the $(N-n) \times N$ matrix whose rows are the transposed unit vectors $e_k^T, k \notin \mathcal{I}$,*

$$\|b^{(2)} - Ax_i^{2,2}\|_2 \le \kappa_2(V_{2,i+1}) \left(\min_{p \in \Pi^i} \max_{\substack{\lambda_k \\ k \notin \mathcal{I}}} |p(\lambda_k)| \, \|\hat{Z}_n\| \, \|P\hat{S}\bar{r}_0^{2,2}\| + \delta \right).$$

Proof. The proof follows from the result of the theorem by first writing (3.9) as $\hat{Z}_n\tilde{p}(\hat{\Lambda})\sum_{k\notin\mathcal{I}}\phi_k\lambda_k e_k$, where $\hat{\Lambda}$ is $\mathrm{diag}(\lambda_k)_{k\notin\mathcal{I}}$, using the identity $e_k = P\hat{S}\hat{z}_k$, and taking norms. □

Thus, under the assumptions we stated at the beginning of this section so that n right eigenvectors with indices in $\mathcal{I}$ of A have been captured when the first seed system is solved, the second seed system converges in exact arithmetic as if the corresponding part of the spectrum of A has been cut off. The strength of this statement in practice is based on the size of δ, which is affected in finite precision arithmetic as noted above.

In the next theorem, we bound the residual norms of the nonseed systems. A proof and detailed discussion of the size of the upper bound are given in [13]. In short, when the right-hand sides are close and the quasi-residual of the seed system is being efficiently reduced, so are the residuals of the nonseed system.

THEOREM 3.4. *Let j denote the index of the seed system and l denote the index of a (nonconverged) nonseed system. Then*

$$\begin{aligned}(3.15)\quad \|r_k^{j,l}\| \le \sqrt{k+1}&\left(|\gamma_1^{j,l}|\left|\prod_{i=0}^{k-1}s_{j,k-i}\right| + \sum_{i=0}^{k-1}|\gamma_{k-i+1}^{j,l}||c_{j,k-i}|\left|\prod_{m=0}^{i-1}s_{j,k-m}\right|\right)\\ &+\sqrt{N-k-1}\|h_k^{j,l}\|,\end{aligned}$$

where $h_k^{j,l} = [\gamma_{k+2}^{j,l},\ldots,\gamma_N^{j,l}]^T$ and $s_{j,k}, c_{j,k}$ are as defined in section 3.1.

4. BL-QMR background.

The BL-QMR algorithm of Freund and Malhotra attempts to solve (1.1) in the following way. First, given K vectors r_i and p vectors l_i, they define

$$R = [r_1,\ldots,r_K], \quad L = [l_1,\ldots,l_p].$$

The block Krylov sequences generated by R, A and L, A^T are

$$(4.1)\qquad \left\{R, AR, A^2R,\ldots,A^{j-1}R,\ldots\right\} \text{ and } \left\{L, A^TL,\ldots,(A^T)^{j-1}L,\ldots\right\}.$$

However, if $A^{j-1}r_i$ (likewise $(A^T)^{j-1}l_i$) is linearly or nearly linearly dependent on the previous vectors, so are all A^kr_i (likewise $(A^T)^kl_i$) for $k \ge j$. Thus, Freund and Malhotra propose scanning the vectors in the two sequences in (4.1) from left to right and deleting those which are linearly or nearly linearly dependent on previous ones. In the process they obtain deflated Krylov sequences whose vectors are linearly independent. We refer to the n-dimensional subspaces generated by these deflated sequences as $K_n^{\mathrm{dl}}(A,R)$ and $K_n^{\mathrm{dl}}(A^T,L)$. Note that in the presence of no deflation, $K_n^{\mathrm{dl}}(A,R)$ and $K_n^{\mathrm{dl}}(A^T,L)$ are spanned by the first n columns of (4.1) with $n \le jK$ or $n \le jp$, respectively.

Within BL-QMR is a Lanczos-type algorithm which incorporates the deflation as mentioned above in order to generate biorthogonal bases for $K_n^{\mathrm{dl}}(A,R)$ and $K_n^{\mathrm{dl}}(A^T,L)$: that is, two sequences of right and left Lanczos vectors

$$v_1,\ldots,v_n \text{ and } w_1,\ldots,w_n, \quad n = 1,2,\ldots,$$

such that

$$(4.2)\qquad \mathrm{span}\{v_1,\ldots,v_n\} = K_n^{\mathrm{dl}}(A,R), \quad \mathrm{span}\{w_1,\ldots,w_n\} = K_n^{\mathrm{dl}}(A^T,L),$$

$$w_j^Tv_k = \left\{\begin{array}{cl} 0 & \mathrm{if} j \ne k, \\ \delta_j \ne 0 & \mathrm{if} j = k. \end{array}\right.$$

Defining the $N \times n$ matrices $V_n = [v_1, \ldots, v_n]$ and $W_n = [w_1, \ldots, w_n]$, it follows that

$$W_n^T V_n = D_n \equiv \operatorname{diag}(\delta_1, \ldots, \delta_n), \quad n = 1, 2, \ldots.$$

Also, the matrix equation relating the v's is

$$AV_\mu = V_n T_\mu + \hat{V}_\mu^{\mathrm{dl}}, \quad \mu \geq 1,$$

where $\mu = n - m_{\mathrm{cr}}$ and m_{cr} is defined by the fact that $K - m_{\mathrm{cr}}$ is the total number of deflations performed in the v sequence up to iteration n in the Lanczos algorithm. Further, T_μ is $n \times \mu$ with lower bandwidth $K+1$ and upper bandwidth $p+1$. Also, $\hat{V}_\mu^{\mathrm{dl}} = V_\mu^{\mathrm{dl}} + E_\mu$, where V_μ^{dl} is $N \times \mu$ but has only $K - m_{\mathrm{cr}}$ nonzero columns corresponding to vectors that are deflated and E_μ has nonzero entries in row i column $p+j, j = 1, \ldots,$ only if a deflation of the ith w occurs for $i > K$. We note that if *deftol* is the deflation tolerance, then $\|V_\mu^{\mathrm{dl}}\| \leq$ *deftol* $\sqrt{K - m_{\mathrm{cr}}}$. For further details, the reader is referred to [8].

Now let us assume $R = [r_0^{(1)}, r_0^{(2)}, \ldots, r_0^{(K)}]$; that is, the matrix R contains the initial residuals of each of the K systems we would like to solve. Thus, the v's correspond to the initial residuals. The way the deflation strategy in [8] works is that if a v is deflated, one linear system is also set aside; then upon convergence of the remaining systems, the solution to the deflated system is updated using the solutions of the other systems. Thus, in what follows we consider only the updates to the nondeflated linear systems, and we denote with a subscript "cr" submatrices of the originals with m_{cr} columns that correspond to these systems.

Recall that when QMR is applied to a single linear system, the μth iterate is an appropriate linear combination of the Lanczos vectors, plus the initial guess. Similarly, the block QMR iterate is defined as

$$X_{\mu,\mathrm{cr}} = X_{0,\mathrm{cr}} + V_\mu Z, \;\; Z \in \mathcal{C}^{\mu \times m_{\mathrm{cr}}}.$$

As with QMR, then, we need to find the matrix Z which determines the appropriate linear combination. Following [8], the residual block $R_{\mu,\mathrm{cr}}$ related to $X_{\mu,\mathrm{cr}}$ satisfies

$$\begin{aligned} R_{\mu,\mathrm{cr}} &= B_{\mathrm{cr}} - AX_{\mu,\mathrm{cr}} \\ &= R_{0,\mathrm{cr}} - AV_\mu Z \\ &= R_{0,\mathrm{cr}} - V_n T_\mu Z - \hat{V}_\mu^{\mathrm{dl}} Z \\ &= V_n \left(\begin{bmatrix} \beta_{\mathrm{cr}} \\ 0 \end{bmatrix} - T_\mu Z \right) - \hat{V}_\mu^{\mathrm{dl}} Z, \end{aligned}$$

where β_{cr} is $m_1 \times m_{\mathrm{cr}}$ defined by taking the appropriate columns of β, with

$$V_{m_1} \beta + V_0^{\mathrm{dl}} = R,$$

and m_1 is the number of columns of R (recall R has K columns) that are not deflated as the first K Lanczos vectors are created ($m_1 \leq K$). Here B contains the $b^{(j)}$'s as its columns, and B_{cr} is the submatrix of B with the appropriate m_{cr} columns.

Because the columns of V_n are not unitary and $\hat{V}_\mu^{\mathrm{dl}}$ has nonzero columns, one cannot find Z such that $\|R_{\mu,\mathrm{cr}}\|$ is minimal. Rather, we seek $Z = Z_\mu$ such that

$$Z_\mu = \arg \min_{Z \in \mathcal{C}^{\mu \times m_{\mathrm{cr}}}} \left\| \begin{bmatrix} \beta_{\mathrm{cr}} \\ 0 \end{bmatrix} - T_\mu Z \right\|.$$

Since T_μ is banded, the standard approach based on the QR factorization of T_μ is used to implicitly determine Z_μ and ultimately determine short-term recurrences for $X_{\mu,\mathrm{cr}}$. Following [8] we have

$$T_\mu = (Q^{(\mu)})^* \begin{bmatrix} R^{(\mu)} \\ 0 \end{bmatrix}$$

for a unitary $n \times n$ matrix $Q^{(\mu)}$ and a nonsingular, $\mu \times \mu$, upper triangular matrix $R^{(\mu)}$. Thus,

$$Z_{(\mu)} = (R^{(\mu)})^{-1} t_\mu, \quad \text{where} \quad \begin{bmatrix} t_\mu \\ \tau_\mu \end{bmatrix} = Q^{(\mu)} \begin{bmatrix} \beta_{\mathrm{cr}} \\ 0 \end{bmatrix}.$$

Finally,

$$\begin{aligned} X_{\mu,\mathrm{cr}} &= X_{0,\mathrm{cr}} + V_{(\mu)}(R^{(\mu)})^{-1} t_\mu \\ &= X_{\mu-1,\mathrm{cr}} + p_\mu y_\mu^T, \end{aligned} \tag{4.3}$$

where p_μ and y_μ^T are given by (refer to [8, equations (5.10), (5.8)])

$$p_\mu = \left(v_\mu - \sum_{i=j^*}^{\mu-1} p_i \theta_i \right) / \theta_\mu, \qquad \begin{bmatrix} y_\mu^T \\ \tau_\mu \end{bmatrix} = Q_\mu \begin{bmatrix} \tau_{\mu-1} \\ 0 \end{bmatrix}. \tag{4.4}$$

The θ_i are scalars corresponding to the last column of $R^{(\mu)}$, Q_μ (not to be confused with $Q^{(\mu)}$) is a particular matrix of Givens rotations described in (5.2) of [8], and j^* is an index described in [8] satisfying $(\mu - j^*) \le 2m$.

5. The block QMR-projection method. In a manner similar to section 3, we describe a block QMR-projection approach to solving (1.1) that combines the advantageous properties of the block QMR algorithm and our single-seed projection algorithm.

Suppose that we select a subset of size $m < K$ linear systems to serve as "seed" from among the original K. Let $\mathcal{I}_{m_1}$ be the index set $i_1, \ldots, i_m$ of the chosen systems. We use $\mathcal{I}_{m_1}^c$ to denote the indices from 1 to K which are not in $\mathcal{I}_{m_1}$. Let $b^{(j)}$ with $j \in \mathcal{I}_{m_1}$ be the m columns of the matrix $B^{(1)}$ and let the remaining $J = K - m$ right-hand sides (corresponding to nonseed systems indexed by $\mathcal{I}_{m_1}^c$) be the columns of the matrix $B^{(2)}$. We define $X_0^{(1)}$ as the matrix $[x_0^{(i_1)}, \ldots, x_0^{(i_m)}]$ of initial guesses for the m seed systems, and $X_0^{(2)}$ as the matrix of initial guesses for the nonseed systems. The corresponding initial block residuals are $R_0^{(1)} = B^{(1)} - AX_0^{(1)}$ and $R_0^{(2)} = B^{(2)} - AX_0^{(2)}$.

The idea is to set R (and L) defined in the previous section to $R_0^{(1)}$ and run BL-QMR to solve the seed systems while using a projection idea to update the nonseed systems. Once BL-QMR converges on the seed system, the process is repeated by choosing a new subset, indexed by $\mathcal{I}_{m_2} \subset \mathcal{I}_{m_1}^c$, of the nonconverged, nonseed systems. The systems indexed by $\mathcal{I}_{m_2}$ now serve as seed, where the columns of $X_0^{(1)}$ are understood to be the estimated solutions, generated in the first round of projected BL-QMR, to the systems with indices in $\mathcal{I}_{m_2}$. The remaining systems, indexed by $\mathcal{I}_{m_2}^c = \mathcal{I}_{m_1}^c \backslash \mathcal{I}_{m_2}$, are updated by projection. In the following, $X_\mu^{(1)}$ ($R_\mu^{(1)}$) denotes the μth block iterate (residual) of the current block seed while $X_\mu^{(2)}$ ($R_\mu^{(2)}$) denotes the μth block iterate (residual) of the current nonseed block. We shall further assume that m

is the number of *current* seed systems and J is the number of *current* nonconverged, nonseed systems. The numbers m and J can change at each round.

At iteration μ, we want our nonseed systems to also lie in the current Krylov subspace. That is, we desire

$$X_\mu^{(2)} \in X_0^{(2)} + K_\mu^{\text{dl}}(A, R_0^{(1)}).$$

Since the columns of V_μ span this subspace, this means

$$X_\mu^{(2)} = X_0^{(2)} + V_\mu Z_\mu^{(2)}$$

for some $\mu \times J$ matrix $Z_\mu^{(2)}$. Now we must decide how to define $Z_\mu^{(2)}$. We observe

$$\begin{aligned} R_\mu^{(2)} &= B^{(2)} - A(X_0^{(2)} + V_\mu Z_\mu^{(2)}) \\ &= R_0^{(2)} - V_n T_\mu Z_\mu^{(2)} - \hat{V}_\mu^{\text{dl}} Z_\mu^{(2)}. \end{aligned}$$

Using biorthogonality

$$D_n^{-1} W_n^T R_\mu^{(2)} = D_n^{-1} W_n^T R_0^{(2)} - T_\mu Z_\mu^{(2)} - D_n^{-1} W_n^T \hat{V}_\mu^{\text{dl}} Z_\mu^{(2)}.$$

Then

$$\|D_n^{-1} W_n^T R_\mu^{(2)}\| \leq \|D_n^{-1} W_n^T R_0^{(2)} - T_\mu Z_\mu^{(2)}\| + \|D_n^{-1} W_n^T \hat{V}_\mu^{\text{dl}} Z_\mu^{(2)}\|.$$

Note that if no deflations have occurred, $\hat{V}_\mu^{\text{dl}}$ is zero, so we have equality rather than inequality. Therefore, in analogy with the single-seed algorithm of section 3, we define

$$Z_\mu^{(2)} \equiv \arg \min_{Z \in \mathcal{C}^{\mu \times J}} \|D_n^{-1} W_n^T R_0^{(2)} - T_\mu Z\|.$$

Using the QR factorization of T_μ described in the previous section, we obtain

$$Z_\mu^{(2)} = \arg \min_{Z \in \mathcal{C}^{\mu \times J}} \left\| Q^{(\mu)} G_n - \begin{bmatrix} R^{(\mu)} \\ 0 \end{bmatrix} Z \right\|,$$

where G_n is the $n \times J$ matrix $G_n = D_n^{-1} W_n^T R_0^{(2)}$. If

$$\begin{bmatrix} t_\mu^{(2)} \\ \tau_\mu^{(2)} \end{bmatrix} = Q^{(\mu)} G_n, \tag{5.1}$$

we obtain

$$Z_\mu^{(2)} = (R^{(\mu)})^{-1} t_\mu^{(2)}, \tag{5.2}$$

so that

$$\|D_n^{-1} W_n^T R_0^{(2)} - T_\mu Z_\mu^{(2)}\| = \|\tau_\mu^{(2)}\|. \tag{5.3}$$

Using $G_n^T = \left[G_{n-1}^T, g_n\right]$, together with (5.1) and the definition of $Q^{(\mu)}$ in (5.1) of [8], it is easy to show that

$$\begin{bmatrix} t_\mu^{(2)} \\ \tau_\mu^{(2)} \end{bmatrix} = \begin{bmatrix} I_{\mu-1} & 0 \\ 0 & Q_\mu \end{bmatrix} \begin{bmatrix} t_{\mu-1}^{(2)} \\ \tau_{\mu-1}^{(2)} \\ g_n^T \end{bmatrix} = \begin{bmatrix} t_{\mu-1}^{(2)} \\ Q_\mu \begin{bmatrix} \tau_{\mu-1}^{(2)} \\ g_n^T \end{bmatrix} \end{bmatrix}. \tag{5.4}$$

Thus, $t_\mu^{(2)}$ differs from $t_{\mu-1}^{(2)}$ only in its last row, which we call $(y_\mu^{(2)})^T$:

$$t_\mu^{(2)} = \left[\begin{array}{c} t_{\mu-1}^{(2)} \\ (y_\mu^{(2)})^T \end{array} \right], \text{ where } (y_\mu^{(2)})^T \in \mathcal{C}^{1\times J}.$$

From the above relation and (5.4) it follows that to obtain $(y_\mu^{(2)})^T$ one need only perform a product with Q_μ:

$$\left[\begin{array}{c} (y_\mu^{(2)})^T \\ \tau_\mu^{(2)} \end{array} \right] = Q_\mu \left[\begin{array}{c} \tau_{\mu-1}^{(2)} \\ g_n^T \end{array} \right],$$

which, since Q_μ by definition is a product of m_{cr} Givens rotations, is an easy task.

With p_i defined as in (4.4), it is now easy to show that the μth nonseed block iterate is

$$X_\mu^{(2)} = X_{\mu-1}^{(2)} + p_\mu (y_\mu^{(2)})^T. \tag{5.5}$$

Thus, we may readily show

$$R_\mu^{(2)} = R_{\mu-1}^{(2)} - Ap_\mu (y_\mu^{(2)})^T. \tag{5.6}$$

However, using the definition of p_μ, we find an update formula for the block residual which does not actually require any additional matrix-vector products.

Lemma 5.1. *$R_\mu^{(2)}$ can be updated from $R_{\mu-1}^{(2)}$ in at most $O(N(J+2m))$ additional floating point operations.*

Proof. By substituting (4.4) into (5.6), we obtain a formula for updating $R_\mu^{(2)}$:

$$R_\mu^{(2)} = R_{\mu-1}^{(2)} - f_\mu (y_\mu^{(2)})^T \quad \text{with} \quad f_i \equiv Ap_i = \frac{1}{\theta_i}\left(Av_i - \sum_{k=j^*}^{i-1} \theta_k f_k \right). \tag{5.7}$$

Consider forming f_μ. Now the matrix-vector product Av_μ is computed in the course of the Lanczos process at iteration μ and need not be recomputed. Therefore, it is clear that to compute the length N vector f_μ requires at most $O(2mN)$ flops since $(\mu - j^*) \leq 2m$ by definition (see section 5 of [8]). We note that the computation of the outer product $f_\mu (y_\mu^{(2)})^T$ requires $O(JN)$ operations, and the proof is complete. □

We note that a similar update is valid for $R_\mu^{(1)}$:

$$R_\mu^{(1)} = R_{\mu-1}^{(1)} - f_\mu (y_\mu^{(1)})^T. \tag{5.8}$$

6. Issues in practical computation for the block algorithm.

6.1. Block-seed selection heuristic. Clearly, the performance of our multiple-seed algorithm in terms of savings of matrix-vector products depends on which, and how many, systems are chosen to be seed. Deflation in BL-QMR solves the problem of removing redundancy if systems with starting residuals which are nearly linearly dependent are chosen as seed. Ideally, however, we would like to choose as seed systems some subset of the nonconverged systems which are in some sense optimally independent in order to increase the chance that the solutions to the nonseed systems will lie nearly in the Krylov subspaces generated by the seed systems, thereby ensuring the effectiveness of the projection process.

In our examples, we used the following heuristics to determine which and how many seed systems to use. First, we let $B = [b^{(1)}, \ldots, b^{(K)}]$. Since $K \ll N$ was not too large in these examples, we computed a compact pivoted QR factorization of B,

$$\tilde{B} \equiv B\Pi = QR, \quad Q \in \mathcal{C}^{N \times K}, \ R \in \mathcal{C}^{K \times K},$$

to determine which of the remaining were most independent. Here, Π is just a permutation matrix which serves to permute the columns of B such that the first few columns of $B\Pi$ are the most independent. In particular, if ρ denotes the diagonal entries of R and if $|\rho(1)|/|\rho(i)| > \alpha$ for any $1 \leq i \leq K$, then we discard the corresponding column of $\tilde{B}$. The remaining m columns of $\tilde{B}$ serve as the seed systems. We set the maximum value of α to 10^5 to ensure the columns were not too linearly dependent, but adjusted it lower if necessary so that the size of the seed block was no bigger than about $K/2$. On the next round of projected BL-QMR, however, we simply decided on a new number of seeds to use ($m \leftarrow \lceil m/2 \rceil$) and took those with the largest m relative residuals to serve as seed. More efficient means of selecting m for each round and for determining the m seeds need to be examined in the future.[3]

6.2. Loss of biorthogonality. One additional problem that we encountered in practice in using either our single-seed or our multiple-seed algorithm was that loss of biorthogonality could affect the accuracy of the $\gamma_n^{jl} = (1/\delta_{j,n}) w_{j,n}^T r_0^{jl}$, or $g_n^T = (1/\delta_n) w_n^T R_0^{(2)}$. This loss of accuracy would thereby adversely affect the convergence of the computed solution. To avoid this difficulty for the block projection algorithm, we used the following fact. If no deflations were performed up to the μth iteration when solving the single-seed system,

$$R_{\mu-1}^{(2)} = R_0^{(2)} - V_{n-1} T_{\mu-1} Z_{\mu-1}^{(2)} \quad \Rightarrow \quad g_n^T = \frac{1}{\delta_n} w_n^T R_{\mu-1}^{(2)} = \frac{1}{\delta_n} w_n^T R_0^{(2)},$$

where it is understood that $R_j^{(2)} = R_0^{(2)}, j < 0$. Thus, at the beginning of iteration $\mu \geq 1$, we computed g_n^T based on the current residual estimate, then updated the residual estimate using Lemma 5.1. If deflations do occur, observe

$$g_n^T = \frac{1}{\delta_n} w_n^T R_{\mu-1}^{(2)} - \frac{1}{\delta_n} w_n^T \hat{V}_{\mu-1}^{\mathrm{dl}} Z_{\mu-1}^{(2)}.$$

In our examples, the second term was on the order of the deflation tolerance. This was because $\hat{V}_\mu^{\mathrm{dl}} = V_\mu^{\mathrm{dl}}$ since no w deflations occurred for indices larger than J. Hence nonzero columns of V_{dl} were nearly linear combinations of the first m_1 v's for which $\frac{1}{\delta_n} w_n^T v = 0$. In this work we choose to ignore the second term rather than go to the extra expense of computing inner products with the nonzero columns of $\hat{V}_{\mathrm{dl}}$.

Likewise, for the single-seed algorithm we use

$$\gamma_n^{jl} = \frac{1}{\delta_{j,n}} w_{j,n}^T r_0^{jl} = \frac{1}{\delta_{j,n}} w_{j,n}^T r_{n-2}^{jl}, \quad n \geq 2.$$

An investigation into the reason behind the success of these approaches in finite precision arithmetic will be the subject of future work. We note that a similar phenomenon was observed in [20] with respect to practical implementation of GMRES variants, and an explanation for such behavior in finite precision arithmetic was given.

[3] In the worst case, if the seed block is too small, then it could require many rounds and much computation for all the systems to converge, and performance could be worse than BL-QMR without projection. If the seed block is too large, gains in execution time over BL-QMR would probably also be reduced, and our algorithm's behavior would become more dependent on the deflation tolerance.

6.3. Computational aspects of the block-seed algorithm. It may be relatively expensive in terms of execution time to move data between the smaller, faster cache and the larger, slower main memory. When data is available in cache, it is desirable to use it as much as possible. The level-2 and level-3 BLAS are better for achieving this than level-1 BLAS operations. Thus, one advantage that the block Krylov subspace algorithms enjoy over standard Krylov subspace algorithms (and our single-seed algorithm) is that the former can be implemented to be rich in higher level BLAS operations, whereas the latter class of algorithms requires a large number of level-1 BLAS operations [14, 2].

Further, Krylov subspace algorithms require a large number of vector inner products relative to the remaining number of computations. These inner products, when implemented on a distributed memory parallel machine, correspond to synchronization points (that is, computation cannot proceed until all processors receive the result of the inner product) and require numerous smaller messages among processors [17]. Our single-seed method inherits these problems, although updates to the seed and nonseed systems can be done independently. Block Krylov subspace algorithms, however, can be implemented to provide more computation between communications and larger but fewer messages among processors [14, 17]. Below, we provide one implementation of the block-seed projection algorithm from the proceeding section. We do not claim that this implementation is optimal in terms of cache utilization or parallelism; our goal is to illustrate the potential efficiencies of the block-seed algorithm and show that it retains the same advantages that the block QMR algorithm enjoys.

Suppose that m is the number of right-hand sides in a given seed block and that J is the number of systems in the nonseed block. Let $m_1 \leq m$ be the number of linearly independent right Lanczos vectors that are formed, using deflation, from the initial residuals to the seed block. In the following, $V_{(k)} = [v_{m_1 k+1}, \ldots, v_{m_1(k+1)}]$ and $W_{(k)} = [w_{m_1 k+1}, \ldots, w_{m_1(k+1)}], 0 \leq k$.

Algorithm 1. $\mu = 1 = \phi$. Given $X_0^{(1)}, R_0^{(1)} \in C^{N \times m_1}$ and $X_0^{(2)}, R_0^{(2)} \in C^{N \times J}$; Given the m_1 columns of $V_{(1)}$ and $W_{(1)}$ and $D_{(1)} = \text{diag}(\delta_1, \ldots, \delta_{m_1})$.

```
For $k = 2$ until seed block converges do
  1. $V_{(k)} = AV_{(k-1)}; W_{(k)} = A^T W_{(k-1)}.$
  2. If deflations occurred in the $W$ (or $V$) sequence, update $V_{(k)}$ (or $W_{(k)}$).
  3. Biorthogonalize the columns of $V_{(k)}$ against the columns of $V_{(k-2)}$, $V_{(k-1)}$;
     biorthogonalize the columns of $W_{(k)}$ against the columns of $W_{(k-2)}, W_{(k-1)}$.
  4. Set $i = 0, j = 0, s = 1, \hat{s} = 1$. Set $\mu = \mu + 1$; $\phi = \phi + 1$.
  5. For $n = (k-1)m_1+1, \ldots, km_1$, set $i = i + 1$ and do
     (a) If $V_{(k)}(:, i)$ does not exist, put $V_{(k)}(:, i) = AV_{(k)}(:, s)$ and biorthogonalize
         against $\phi_\mu$ previous Lanczos pairs; $s = s + 1$.
     (b) If $\|V_{(k)}(:, i)\| < deftol$, then $j = j + 1$ and deflate:
           i. Compute $(y_\mu^{(1)})^T, (y_\mu^{(2)})^T, \tau_\mu^{(1)}, \tau_\mu^{(1)}$; compute $f_\mu, p_\mu$ (via gaxpy's).
          ii. "Delete" $i$th column of $V_{(k)}$, "shift" remaining columns left 1.
         iii. Compute which system gets dropped from the seed block (that row
              of $y_\mu^{(1)}$ will have a zero entry).
          iv. Let $y_\mu^{(i_v)}, f_\mu, p_\mu$ be the $j$th columns of $Y^{(i_v)}, i_v = 1, 2; F; P$.
           v. Update deflation index sets. Set $\mu = \mu + 1$, goto step 5(a).
     (c) Normalize $V_{(k)}(:, i)$ and set $\mu_n = \mu$; $j = j + 1$:
     (d) If $W_{(k)}(:, i)$ does not exist, put $W_{(k)}(:, i) = A^T W_{(k)}(:, \hat{s})$ and biorthogo-
         nalize against $\mu_\phi$ previous Lanczos pairs; $\hat{s} = \hat{s} + 1$.
     (e) If $\|W_{(k)}(:, i)\| < deftol$, then deflate:
            i. "Delete" $i$th column of $W_{(k)}$, "shift" remaining columns left 1.
            ii. Update deflation index sets. Set $\phi = \phi + 1$, goto step 5(d).
        (f) Normalize $W_{(k)}(:,i)$. Set $\phi_n = \phi$ and $\delta_n = W_{(k)}(:,i)^T V_{(k)}(:,i)$.
        (g) Continue the MGS[4] process on columns $\geq i$ of $V_{(k)}, W_{(k)}$.
        (h) Set $g_n^T = (1/\delta_n) W_{(k)}(:,i)^T R_{\mu-j}^{(2)}$.
        (i) Compute $(y_\mu^{(1)})^T, (y_\mu^{(2)})^T, \tau_\mu^{(1)}, \tau_\mu^{(2)}$; compute $f_\mu, p_\mu$ (via gaxpy's).
        (j) Let $y_\mu^{(i_v)}, f_\mu p_\mu$ be the $j$th columns of $Y^{(i_v)}, i_v = 1,2; F; P$.
    6. $X_\mu^{(1)} = X_{\mu-j}^{(1)} + P(Y^{(1)})^T$;   $R_\mu^{(1)} = R_{\mu-j}^{(1)} - F(Y^{(1)})^T$.
    7. $X_\mu^{(2)} = X_{\mu-j}^{(2)} + P(Y^{(2)})^T$;   $R_\mu^{(2)} = R_{\mu-j}^{(2)} - F(Y^{(2)})^T$.
    8. "Remove" deflated systems from $X_\mu^{(1)}$.
```

One benefit of this implementation is that A, A^T are accessed only once each block iteration, if no deflation occurs, and computing products of A and A^T with dense, rectangular matrices of Lanczos vectors makes better use of cache than products of A with a single vector. In step 2 (also 5(a), 5(d)) some columns of either or both of the current blocks may have to be biorthogonalized against some previous Lanczos vectors if certain previous deflations occurred in the V and/or W sequence. In step 3, the current blocks of Lanczos vectors are biorthogonalized against the appropriate columns[5] of the previous two blocks. Considering the matrix $V_{(k)}$ (or $W_{(k)}$) rather than its columns separately, we can do this using level-2 BLAS with a two-sided modified Gram–Schmidt approach or, at the expense of some stability, we could accomplish this with level-3 BLAS via a block modified Gram–Schmidt approach [14]. For each deflation step, however, we incur the price of one matrix-vector product and several vector-wise inner products. It is possible to reorganize the algorithm so that a Lanczos block effectively decreases in size after deflation (possibly leaving left and right blocks of different sizes) and thereby put off this extra work until it can be done with higher-level BLAS, but as the notation is more tedious, we use the current implementation to illustrate our points.

Notice we are using modified Gram–Schmidt to biorthogonalize within the current block, but that computation toward updating the solution and residual blocks is done between each step of the process. Notice also that the block iterates and residuals are only updated after a new block of Lanczos vectors has been generated; this was done in order to minimize the number of accesses to the block iterates and residuals and to incorporate level-3 BLAS operations. The updating could be done (via level-2 BLAS operations) inside the innermost loop according to (4.3), (5.5), (5.7), (5.8), or one might opt to track the size of $\tau_\mu^{(1)}$ and update only when necessary.

One way to implement the algorithm on a distributed memory parallel machine is to row partition [17, 14] the matrices $F, P, V_{(k)}, W_{(k)}, X_\mu^{(i_v)}, R_\mu^{(i_v)}, i_v = 1, 2$. Thus, the matrix-multiplications with A, A^T, the biorthogonalization steps, and steps 5(b), (c), (e), (f), (h) require communication among processors; most of the other steps require only local updates of portions of the rectangular matrices. As in [17, section 3.1], it may be possible in a parallel implementation to exploit any computations that are mostly decoupled: for example, the updates to solution and residual blocks are somewhat independent of the generation of the Lanczos vectors and of each other.

[4] Modified Gram–Schmidt.

[5] If deflations have occurred in the W sequence, then one need only biorthogonalize $V_{(k)}$ against some of the columns of $W_{(k-2)}$ (or $W_{(k-1)}$), rather than against the whole block, and similarly for computing $W_{(k)}$ if deflations occurred in the V sequence.

Techniques for performing matrix-vector products with A, A^T in parallel should also be employed.

Before executing Algorithm 1, using our heuristic in section 6.1, we do a compact pivoted QR factorization of B to determine the seed block and therefore should not incur much overhead beyond what the BL-QMR algorithm in [8] would need to generate its initial block using deflation. Every other time that a seed must be chosen, we select those $\lceil m/2 \rceil$ systems with largest residual norm, so a small amount of additional computations/communication are needed at the seed selection steps.

Clearly, if the first time Algorithm 1 is called $J = 0$, it is just the block QMR algorithm with deflation. If $J \neq 0$, then the first time Algorithm 1 is executed, for a given k, the same number of solution vectors and residual vectors must be updated (at most) as if J had been 0: the difference is that updates are separated into updates on two different solution and residual blocks, and these updates are independent of one another. Thus, if Algorithm 1 is implemented in parallel, the independent updates may help compensate for the execution time that is due to processor communication. The computation of g_n^T and $(y_\mu^{(2)})^T$ are the most notable differences between the unprojected and projected algorithms. Overall, the block-seed projection approach may yield the following advantages over block QMR:

- fewer accesses to A and less communication among processors;
- smaller seed block sizes with the block-projection approach mean fewer flops are needed to compute products with A, A^T and fewer flops are needed to construct Lanczos pairs and to factor T_μ;
- less storage per processor during a given run of Algorithm 1 (as fewer vectors are needed in the recurrences for p_μ, f_μ and the number of columns of $V_{(k)}, W_{(k)}$, etc., are reduced);
- updating the seed and nonseed blocks can be done independently.

7. Numerical results. In this section we give numerical results which indicate the potential effectiveness of our approach on electromagnetic scattering problems. All experiments were conducted in Matlab using IEEE double precision floating point arithmetic on a 600 MHz Pentium II processor. We compare our results with results from the Matlab implementation of block QMR with deflation, algorithm BL-QMR in [8]. For comparison purposes, the implementation of our block-seed algorithm mirrors the implementation of BL-QMR in [8] with modifications where necessary, rather than the one in the preceding section.

Mathematically, we would like to solve a two-dimensional Helmholtz-type equation for the scattered electric field $E(x, y)$:

$$(\Delta + k^2(x,y))E(x,y) = \chi_m(x,y)E_0(x,y) \quad \text{in } \Omega \tag{7.1}$$

with perfectly matched layer (PML) boundary conditions [3, 18]: the specific mathematical formulation we use is described in [12]. Here $k^2(x, y) = \omega^2\mu_0\epsilon(x, y)$ is the square of the wave number, with ω representing angular frequency and μ_0 a constant denoting the magnetic permeability. The function $\epsilon(x, y)$, called the electrical permittivity, is defined as $\epsilon = \epsilon_0\epsilon_{rel} + i\frac{\sigma}{\omega}$ for some real $\sigma \geq 0, \epsilon_{rel} \geq 1$ with $i = \sqrt{-1}$ and ϵ_0 a constant (the permittivity of free-space). The value σ is the conductivity of the material. The function χ_m describes the properties of the buried object and has support only over the object location. $E_0(x, y)$ is the known incident electric field.

We discretize using finite differences, which leads to a matrix equation involving the matrix A which is $N \times N$, sparse, complex, and structured but neither symmetric nor Hermitian due to the boundary conditions. Because the matrix is highly indefinite,

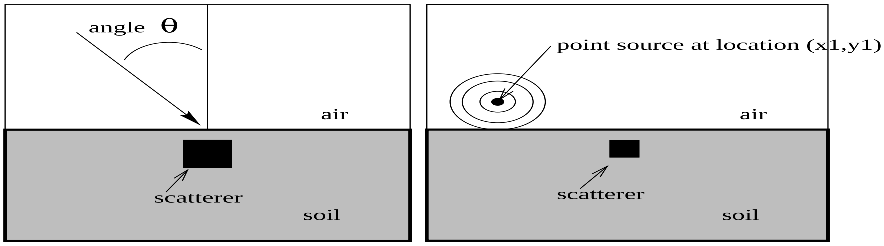


FIG. 7.1. *Physical configurations for Example* 1 *(left) and Example* 2 *(right).*

we need to use a preconditioner to speed convergence. The preconditioner we use is the one described in [12], and we perform all preconditioning from the right.

For all algorithms, we take the initial starting guesses $x_0^{(j)}$ to be zero. We stop running our algorithms when the relative residual norms of all of the systems are less than $tol = 10^{-7}$. For the two block-based algorithms, ours and BL-QMR, we update (seed) block residuals via (5.8) ((5.7) is used for the nonseed block[6]). We monitor convergence of the current seed block by checking norms of the columns of $R_\mu^{(1)}$. However, the true norms of the residuals in the seed block were computed and checked to satisfy the convergence tolerance before the block was deemed to have converged. Since for these examples the major computational expense per iteration is the two matrix-vector products with applications of the preconditioner, we consider the total number of matrix-vector products required for all the systems to converge as our primary measure of success and discuss some timing results.

7.1. Example 1. In this experiment we would like to find the scattered electric fields caused when plane waves at various angles impinge on a horizontal air-soil interface and scatter from a 7cm × 6cm object buried 5cm below the surface. Each angle corresponds to a different E_0 in (7.1), which in turn corresponds to a different right-hand side $b^{(j)}$ in (1.1). Figure 7.1 gives a physical illustration of the problem.

In this example, we use a soil type (referred to as "Seabee" in the literature [19]) and conduct experiments at two different frequencies, $\omega/(2\pi) = 45$ MHz and 475 MHz. At 45 MHz, Seabee has $\epsilon_{rel} = 35.65$ and $\sigma = .13$, while at 475 MHz $\epsilon_{rel} = 21.31$ and $\sigma = .23$. We assumed that the buried object has $\epsilon_{rel} = 2.9$ and $\sigma = .001$ at both frequencies. For air, $\epsilon_{rel} = 1$, $\sigma = 0$. We have discretized at a rate of 50 points per wavelength of soil at 45 MHz and 20 points per wavelength at 475 MHz. In both cases, the total number of unknowns (N) is $(2^7 + 15)^2$.

We centered the buried object (refer to Figure 7.1) and considered the scattered field due to plane waves impinging on the surface at evenly spaced angles from −60 to 60 degrees from the normal. The second columns in Tables 7.1 and 7.2 give the total number of matrix-vector products needed if preconditioned QMR is applied to each system separately.[7] The third column gives the total number of matrix-vector products needed if our preconditioned QMR with projection algorithm is used. The next several columns give the total number of matrix-vector products computed when solving the problem using BL-QMR with various deflation tolerances. The final

[6] Note that the latter needs to be computed each iteration to determine the update for the nonseed block, whereas one could use the bound in [8] to monitor convergence of the seed block at the possible expense of computing many extra matrix-vector products.

[7] Note that we define the matrix-vector product count as the number of multiplies by AM^{-1} or its transpose where M is the preconditioner.

Table 7.1

Example 1. *45 MHz: Number of matrix-vector products required for convergence by each of the methods (each system independently, single-seed QMR-projection, BL-QMR with deflation tolerances* $10^{-8}, 10^{-9}$, *and* 10^{-10}, *and block QMR with projection) for experiments involving different numbers of right-hand sides (RHS). Dashes indicate no convergence of the method in under* 300 *iterations.*

	No. matrix-vector products, 45 MHz						
No. RHS	Seq.	Prj.	BQ 1E–8	BQ 1E–9	BQ 1E–10	BQ 1E–11	BQ + Prj.
7	856	292	–	201	219	219	173
13	1590	254	–	211	227	239	180
25	3054	280	–	233	243	267	207

Table 7.2

Example 1. *475 MHz: Number of matrix-vector products required for convergence by each of the methods for experiments involving different numbers of right-hand sides.*

	No. matrix-vector products, 475 MHz						
No. RHS	Seq.	Prj.	BQ 1E–7	BQ 1E–8	BQ 1E–9	BQ 1E–10	BQ + Prj.
7	664	420	223	223	223	223	221
13	1236	468	–	303	321	319	237
25	2378	492	–	321	319	–	245

column shows results when our block-seed approach is used ($deftol = 10^{-9}$), where the seed blocks are chosen using the heuristic outlined in section 6.1 with $\alpha = 10^5$. Dashes indicate that the convergence tolerance was not met within $maxit = 300$ iterations.

As Table 7.1 shows, for the 45 MHz case, BL-QMR failed to converge after 300 iterations in all cases when the deflation tolerance was set to 10^{-8}, but it outperformed our single-seed projection method if the deflation tolerance was small enough. Our block-seed projection approach performs better than all the other methods in terms of the number of matrix-vector products; however, we note that for these nonoptimized implementations, the execution times for the best BL-QMR runs and our block algorithm are about the same. Solving sequentially took 4.5 times longer than our single-seed method and over 6 times longer than block-based algorithms.

At 475 MHz, we expected our $x^{(j)}$'s not to be as close as in the previous case due to the underlying physics of the problem, and therefore we did not expect as much savings with our single-seed projection approach. Indeed, Table 7.2 shows that the difference between the second and third columns is not as dramatic as in Table 7.1. Table 7.2 also shows that BL-QMR, with the deflation tolerance set at either 10^{-8} or 10^{-9}, outperforms our single-seed projection approach. However, comparing the last column with the others, we find that our block-seed projection approach can provide substantial savings over the other methods. There is also a difference in execution times: for example, for 25 systems our block-seed projection method takes about 6.2 minutes while BL-QMR with deflation tolerance of 10^{-9} takes about 7.1 minutes.

7.2. Example 2. For our second example, each of our K systems corresponds to solving for the scattered electric field from a buried object when the source of the incident field is a point source, located at position x_i, y_i above the earth (see Figure 7.1). We consider the case when the frequency is 480 MHz, and the soil has $\epsilon_{rel} = 6.5$ and $\sigma = .019$. As before, the buried object has $\epsilon_{rel} = 2.9$ and $\sigma = .001$. The buried object is 7cm × 4cm buried 5cm deep and centered left to right. The width of each cell in the discrete grid is 1cm, and the total number of unknowns (N) is $(2^6 + 15)^2$. Our point sources are each located 3cm above the earth and either

Table 7.3

Example 2. *Number of matrix-vector products required for convergence by each of the methods for experiments involving different numbers of right-hand sides. Dashes indicate no convergence of the method in under* 300 *iterations.*

	No. matrix-vector products					
No. RHS	Seq.	Prj.	BQ 1E–9	BQ 1E–11	BQ 1E–12	BL-QMR + Prj.
25	1894	1084	–	–	345	209
35	2658	1468	–	–	411	289

Table 7.4

Example 2. *Approximate execution time in minutes. Dashes indicate no convergence of the method in under* 300 *iterations.*

	Minutes					
No. RHS	Seq.	Prj.	BQ 1E–9	BQ 1E–11	BQ 1E–12	BQ + Prj.
25	4.2	4.5	–	–	10.6	2.0
35	5.9	7.4	–	–	33.1	3.7

vary in the horizontal direction, with 0 being in the middle, from −12cm to 12cm in 1cm increments or −17cm to 17cm in 1cm increments. The numbers of matrix-vector products required by each of the different methods to solve these systems are given in Table 7.3. However, as illustrated by the timing results in Table 7.4, both the single- and block-seed projection give dramatic improvements over BL-QMR without projection.[8]

8. Conclusions and future work. We have introduced two new projection approaches, based on QMR and block QMR, respectively, for solving multiple linear systems with the same coefficient matrix but different right-hand sides. Compared to solving each system separately by QMR, both approaches can significantly reduce the work and execution time needed to solve all the systems to within a specified tolerance provided there is sufficient shared information among the right-hand sides; the block-seed algorithm requires less shared information to perform well. We provided theory for the single-seed approach which suggests that under certain conditions in exact arithmetic, QMR on subsequent seed systems converges as if part of the spectrum has been cut off; we also gave an upper bound for the rate of convergence of the nonseed systems. More work needs to be done to determine convergence behavior of the block-seed algorithm in both exact and finite precision arithmetic.

As our numerical results showed, with appropriate deflation tolerance, the BL-QMR algorithm [8] could outperform our single-seed QMR-projection method in terms of matrix-vector product savings (although not always reflected in the execution times) particularly when the right-hand sides are not as close; however, our block-seed projection method consistently exhibited the greatest savings in such cases. The performance of our block-seed approach depends on our choices of successive seed blocks, and overall execution time depends on the actual implementation. We gave one block-seed selection heuristic and discussed possible efficiencies of block-seed algorithm. Determining good seed selection strategies, efficient serial and parallel implementations, and formal time comparisons with other methods remain subjects for future research.

[8] Part of the improvement can be attributed to the difference in size of the BL-QMR block iterate with the size of the seed for our methods.

REFERENCES

[1] Z. Bai, *Error analysis of the Lanczos algorithm for the nonsymmetric eigenvalue problem*, Math. Comp., 62 (1994), pp. 209–226.

[2] Z. Bai, D. Day, and Q. Ye, *ABLE: An adaptive block Lanczos method for non-Hermitian eigenvalue problems*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 1060–1082.

[3] J. Berenger, *A perfectly matched layer for the absorption of electromagnetic waves*, J. Math. Phys., 114 (1994), pp. 185–200.

[4] W. E. Boyse and A. A. Seidl, *A block QMR method for computing multiple simultaneous solutions to complex symmetric systems*, SIAM J. Sci. Comput., 17 (1996), pp. 263–274.

[5] T. F. Chan and W. L. Wan, *Analysis of projection methods for solving linear systems with multiple right-hand sides*, SIAM J. Sci. Comput., 18 (1997), pp. 1698–1721.

[6] J. Cullum, *Arnoldi versus nonsymmetric Lanczos algorithms for solving nonsymmetric eigenvalue problems*, BIT, 36 (1996), pp. 470–493.

[7] D. Day, *An efficient implementation of the nonsymmetric Lanczos algorithm*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 566–589.

[8] R. Freund and M. Malhotra, *A block QMR algorithm for non-Hermitian linear systems with multiple right-hand sides*, Linear Algebra Appl., 254 (1997), pp. 197–257.

[9] R. Freund and N. Nachtigal, *QMR: A quasi-minimal residual method for non-Hermitian linear systems*, Numer. Math., 60 (1991), pp. 315–339.

[10] R. W. Freund and N. M. Nachtigal, *An implementation of the QMR method based on coupled two-term recurrences*, SIAM J. Sci. Comput., 15 (1994), p. 313–337.

[11] P. Joly, *Résolution de Systèms Linéaires avec Plusieurs Seconds Membres par la Méthode du Gradient Conjugué*, Tech. report R-91012, Publications du Laboratoire d'Analyse Numeriqué, Université Pierre et Marie Curie, Paris, 1991.

[12] M. Kilmer, E. Miller, and C. Rappaport, *Preconditioners for Structured Matrix Problems Arising in Subsurface Object Detection*, preprint, http://www.tufts.edu/~mkilme01.

[13] M. Kilmer, E. Miller, and C. Rappaport, *QMR-Based Projection Techniques for the Solution of Non-Hermitian Systems with Multiple Right Hand Sides*, Tech. report TR-CDSP-00-51, Center for Communications and Digital Signal Processing, Northeastern University, Boston, MA, 2000.

[14] G. Li, *A block variant of the GMRES method on massively parallel processors*, Parallel Comput., 23 (1997), pp. 1005–1019.

[15] A. A. Nikishin and A. Yu. Yeremin, *Variable block CG algorithms for solving large sparse symmetric positive definite linear systems on parallel computers.* I. *General iterative scheme*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 1135–1153.

[16] D. P. O'Leary, *The block conjugate gradient algorithm and related methods*, Linear Algebra Appl., 29 (1980), pp. 293–322.

[17] D. P. O'Leary, *Parallel implementation of the block conjugate gradient algorithm*, Parallel Comput., 5 (1987), pp. 127–139.

[18] C. Rappaport, *Interpreting and improving the PML absorbing boundary condition using anisotropic lossy mapping of space*, IEEE Trans. Magn., 32 (1996), pp. 968–974.

[19] E. M. Rosen and T. W. Altshuler, *Analysis of UXO and clutter signatures from the DARPA background clutter experiment*, in Proceedings of the UXO Forum, 1998.

[20] M. Rozložník and Z. Strakoš, *Variants of the residual minimizing Krylov space methods*, in Proceedings of the Eleventh Summer School Software and Algorithms of Numerical Mathematics, I. Marek, ed., University of West Bohemia, Plzen, Czech Republic, 1995, pp. 208–225.

[21] Y. Saad, *On the Lanczos method for solving symmetric linear systems with several right hand sides*, Math. Comp., 48 (1987), pp. 651–662.

[22] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing, New York, 1996.

[23] V. Simoncini, *A stabilized QMR version of block BICG*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 419–434.

[24] V. Simoncini and E. Gallopoulos, *An iterative method for nonsymmetric systems with multiple right-hand sides*, SIAM J. Sci. Comput., 16 (1995), pp. 917–933.

[25] C. F. Smith, *The Performance of Preconditioned Iterative Methods in Computational Electromagnetics*, Ph.D. thesis, University of Illinois at Urbana-Champaign, Illinois, 1987.

[26] C. F. Smith, A. F. Peterson, and R. Mittra, *A conjugate gradient algorithm for the treatment of multiple incident electromagnetic fields*, IEEE Trans. Antennas and Propagation, 37 (1989), pp. 1490–1493.

SIAM J. SCI. COMPUT. © 2001 Society for Industrial and Applied Mathematics
Vol. 23, No. 3, pp. 781–804

ON AN ADAPTIVE MULTIGRID SOLVER FOR CONVECTION-DOMINATED PROBLEMS*

WOLFGANG DAHMEN†, SIEGFRIED MÜLLER†, AND THOMAS SCHLINKMANN†

Abstract. The objective of this paper is to develop and describe an adaptive multigrid scheme for the numerical solution of convection-diffusion-reaction equations with dominating convection or reaction term. Specifically, we propose a systematic construction of problem and *scale-dependent* restriction, prolongation and smoothing operators in combination with an *adaptive mesh refinement* strategy. In particular, adaptivity is not only to reduce computational complexity but also to *stabilize* standard Galerkin discretizations so that adding artificial viscosity can be avoided. The approach is suited for any choice of underlying multiresolution sequences, in particular for standard finite element discretizations on nested but possibly nonuniform meshes. Moreover, all concepts are in principle independent of the spatial dimension and avoid any special flow-dependent enumeration of unknowns. The performance of the scheme is illustrated by various examples in one and two spatial dimensions. They are chosen so as to exhibit possibly many types of adverse or singular behavior in connection with constant and varying convection or the interplay between convective and reactive terms. In the light of these tests those issues are identified that require further research.

Key words. convection dominated problems, stable completions, scale-dependent transfer operators and smoothers, full multigrid, adaptive refinements

AMS subject classifications. 65M12, 35J20, 76Rxx, 65N30

PII. S106482759935544X

1. Introduction.

This paper is concerned with the numerical solution of *convection-diffusion equations* which appear as core ingredients of more complex models such as the Navier–Stokes equations or models for two-phase flow problems. To be specific, we consider scalar boundary value problems

$$-\nabla \cdot (\boldsymbol{\nu}\, \nabla\, u) + \boldsymbol{\beta} \cdot \nabla\, u + \gamma\, u = f \quad \text{in } \Omega \subset \mathbb{R}^d, \quad u = 0 \ \text{ on } \partial\Omega, \tag{1}$$

where the principal part of the differential operator is assumed to be self-adjoint and positive definite, i.e., $\boldsymbol{\nu} = \boldsymbol{\nu}^{\mathrm{T}}$ and there exists some $\lambda > 0$ such that $\boldsymbol{\xi}^T\, \boldsymbol{\nu}(\mathbf{x})\, \boldsymbol{\xi} \geq \lambda\, \boldsymbol{\xi}^T\, \boldsymbol{\xi}$ for all $\mathbf{x} \in \overline{\Omega}$, $\boldsymbol{\xi} \in \mathbb{R}^d$. The flow direction $\boldsymbol{\beta}$ as well as $\gamma \geq 0$ may vary in $\mathbf{x}$.

The corresponding variational formulation of (1) then reads as follows: Find $u \in H_0^1(\Omega)$ such that

$$a_{\boldsymbol{\nu}}(v, u) = b(v) \quad \forall\, v \in H_0^1(\Omega), \tag{2}$$

where the bilinear form $a_{\boldsymbol{\nu}} : H_0^1(\Omega) \times H_0^1(\Omega) \to \mathbb{R}$ and the linear form $b : H_0^1(\Omega) \to \mathbb{R}$ are defined by the integrals

$$a_{\boldsymbol{\nu}}(v, u) := \int_\Omega (\nabla\, v)^T\, (\boldsymbol{\nu}\, \nabla\, u) + (\boldsymbol{\beta} \cdot \nabla\, u)\, v + \gamma\, u\, v\, d\Omega, \quad b(v) := \int_\Omega f\, v\, d\Omega. \tag{3}$$

*Received by the editors April 28, 1999; accepted for publication (in revised form) March 15, 2001; published electronically August 15, 2001.

http://www.siam.org/journals/sisc/23-3/35544.html

†RWTH Aachen, Institut für Geometrie und Praktische Mathematik, Templergraben 55, 52056 Aachen, Germany (dahmen@igpm.rwth-aachen.de, mueller@igpm.rwth-aachen.de, thomas@igpm.rwth-aachen.de). The work of the first author was supported in part by the European Commission within the TMR project (Training and Mobility for Researchers) "Wavelets and Multiscale Methods in Numerical Analysis and Simulation," ERB FMRX CT98 018TA. The work of the second author was supported in part by the SFB 401 of the Deutsche Forschungsgemeinschaft.

Our notation indicates the dependence on $\boldsymbol{\nu}$ because we will be particularly interested in those cases where $\boldsymbol{\nu}$ is small compared with $\boldsymbol{\beta}$ and γ.

Under the assumptions $\nu_{ik} = \nu_{ki} \in C(\overline{\Omega})$, $\beta_i \in C^1(\overline{\Omega})$, $\gamma \in C(\overline{\Omega})$, $\gamma \geq 0$, $f \in L_2(\Omega)$, and $\gamma(\mathbf{x}) - \frac{1}{2}(\operatorname{div} \boldsymbol{\beta})(\mathbf{x}) > -\lambda \min(1, \alpha)$ for all $\mathbf{x} \in \overline{\Omega}$, where the constant $\alpha > 0$ results from Friedrich's first inequality, the existence of a unique solution can be inferred immediately from the Lax–Milgram lemma. The verification of the coercivity condition, i.e., the existence of finite positive constants c, C such that

$$c\|v\|^2_{H^1(\Omega)} \leq a(v,v) \leq C\|v\|^2_{H^1(\Omega)} \quad \forall\, v \in H^1_0(\Omega),$$

can be found, e.g., in [1].

When $\|\boldsymbol{\nu}\|$ is large compared with $\|\boldsymbol{\beta}\|$ and $\|\gamma\|$ the behavior of the solution to (1) is essentially governed by the dominating second order elliptic operator. Correspondingly standard Galerkin discretizations are appropriate and fast multilevel solvers are available that facilitate an efficient numerical treatment. However, when the convective part dominates, i.e., when $\|\boldsymbol{\beta}\| \gg \|\boldsymbol{\nu}\|$, at least the following two serious obstructions arise.

Robustness. Multilevel or multigrid schemes which are asymptotically optimal in the elliptic case $\boldsymbol{\beta} = \mathbf{0}$ lose their efficiency or even fail to converge. Therefore considerable effort has been spent in attempts to increase the *robustness* of the schemes. Strategies for restoring robustness concern algebraic multigrid concepts [29], the development of scale- and problem-dependent multigrid ingredients [32, 16, 24], local Fourier analysis [25, 26], or special enumeration schemes combined with incomplete LU factorizations as smoothers [18, 33, 19, 8] when the flow is by and large acyclic.

Stability. Standard Galerkin discretizations become *unstable*, i.e., if the *grid-Péclet number* $P_h := \|\boldsymbol{\beta}\|h/(2\|\nu\|)$ is of the order 1 (where h denotes the meshsize of the underlying trial space), the solution of the discrete problem may exhibit oscillations that have nothing to do with the true solution. However, a-priorily fixed *stabilization techniques* such as *Petrov–Galerkin* discretizations or *streamline diffusion* [21, 10, 20, 22] add in one way or the other some *artificial viscosity* which, at least in certain regions (often of primary interest), diminish accuracy.

Thus overall, it is fair to say that the problem has by far not been solved to complete satisfaction with regard to both adequate discretization as well as efficiency of the solution process.

Our objectives. Our point of view here is different in that we aim at *jointly* addressing *both stability and robustness* together with the issue of *accuracy*. Our primary goal is to retain accuracy also in critical regions through *dynamically adapting* discretizations. Our main point is that thereby we restore stability of the discretization, as well as efficiency and also to some extent robustness of the solver without a priori stabilization. This is similar in spirit to the approach in [3, 2] where adaptivity is based on a defect correction scheme for finite difference discretizations. The stabilizing effect is also reminiscent of *Shishkin-grids* for which stability of Galerkin discretizations can be shown in special situations [27].

Our starting point is the familiar hierarchical two-by-two blocking of the linear discrete system as in the hierarchical basis multigrid (HBMG) scheme from [5]. The level-dependent multigrid ingredients will be automatically generated through approximately *decoupling* such block systems quite similar in spirit to the approach in [7].

An essential *distinction* from previous work lies in stabilization solely through adaptation and in the way adaptation is realized. Here this becomes possible through an *adaptive full multigrid* scheme combined with a *problem homotopy.* This means that on coarse levels viscosity is being added which will then be successively removed when progressing to higher (possibly local) refinement levels. Unless one decides to still use some weak a priori stabilization the refinement process only stops when all artificial viscosity has been removed everywhere. The adaptation criterion is based on wavelet thresholding combined with the concept of *stable completions* from [9]. This requires no a priori knowledge of the solution and, in particular, avoids regularity assumptions as in [3, 2]. Finally, our decoupling mechanism *automatically* modifies the test spaces resulting in a *Petrov–Galerkin* discretization with increasing upwind effect when descending to coarser scales.

Although the scheme will be tested only for one- and two-dimensional examples we emphasize that all ingredients work in higher dimensions in the same way. In particular, flow-dependent numberings are avoided.

The contents. The paper is organized as follows. In section 2 we briefly describe the general discretization setting. Section 3 is devoted to the development of the basic multigrid ingredients such as scale-dependent intergrid transfer operators and smoothers. An "ideal" two-grid scheme will serve as a guideline for the multigrid scheme. In section 4 an adaptive full multigrid scheme is developed.

In section 5 the performance of this concept is tested for problems which exhibit different types of obstructions or singular behavior. This concerns the formation of boundary layers due to a strong constant convection and reaction terms or their interplay or the effect of variable convection, especially for mass accumulation.

2. Discretization. We will base the numerical treatment of (1) on a hierarchy $\mathcal{S}$ of *nested* trial spaces $S_0 \subset S_1 \subset \cdots \subset H_0^1(\Omega)$ whose union is dense in $H_0^1(\Omega)$. Each space S_j is spanned by a finite set $\Phi_j = \{\phi_{j,k} : k \in \mathcal{I}_j\}$ of basis functions, i.e., each $u_j \in S_j$ has a unique expansion

$$u_j = \sum_{k \in \mathcal{I}_j} u_{j,k} \phi_{j,k} =: \Phi_j^{\mathrm{T}} \mathbf{u}_j.$$

Moreover, we will always assume that the bases Φ_j are uniformly stable in the sense that for some finite positive constants c, C

$$c\|\boldsymbol{u}_j\|_{\ell_2(\mathcal{I}_j)} \leq \|\Phi_j^T \boldsymbol{u}_j\|_{L_2(\Omega)} \leq C\|\boldsymbol{u}_j\|_{\ell_2(\mathcal{I}_j)}, \tag{4}$$

where $\|\cdot\|_{\ell_2(\mathcal{I}_j)}, \|\cdot\|_{L_2(\Omega)}$ denote the respective Euclidean and L_2-norms. Canonical choices of S_j are spaces of globally continuous piecewise linear functions with respect to a nested sequence of successively refined mesh partitions of Ω. The stability of the corresponding bases of *hat functions* is known to hold as long as the triangles are shape regular. It will be helpful to always think in terms of this example although any standard finite element spaces are covered.

Although we are ultimately interested in *standard Galerkin* discretizations of (1) it will be adequate to consider the more general setting of *Petrov–Galerkin* schemes, i.e., fixing some highest level J we look for $u_J \in S_J$ such that

$$a_{\boldsymbol{\nu}}(v, u_J) = b(v) \quad \forall\, v \in \tilde{S}_J, \tag{5}$$

where the $\tilde{S}_j = \operatorname{span}\{\tilde{\phi}_{j,k} : k \in \mathcal{I}_j\}$ are suitable nested *test spaces.* Clearly this amounts to solving the linear system

$$\mathbf{A}_J \, \mathbf{u}_J = \mathbf{b}_J, \tag{6}$$

where the stiffness matrix $\mathbf{A}_J$ and the right-hand side $\mathbf{b}_J$ are defined by

$$\mathbf{A}_J := \Big(a_{\boldsymbol{\nu}}(\tilde{\phi}_{J,i}, \phi_{J,j})\Big)_{i,j\in\mathcal{I}_J}, \quad \mathbf{b}_J := \Big(b(\tilde{\phi}_{J,i})\Big)_{i\in\mathcal{I}_J}. \tag{7}$$

It will often be convenient to use the abbreviation $\mathbf{A}_J = a_{\boldsymbol{\nu}}(\tilde{\Phi}_J, \Phi_J)$, i.e., for any two finite collections Θ, Ξ we set $a_{\boldsymbol{\nu}}(\Theta, \Xi) := (a_{\boldsymbol{\nu}}(\theta, \xi))_{\theta\in\Theta,\xi\in\Xi}$.

3. Space splittings and multigrid ingredients. Our goal is to solve (6) efficiently by exploiting lower scale discretizations. We proceed with briefly recalling the mechanisms for problem-dependent space splittings.

3.1. Stable completions. A key idea is to split a given trial space S_{j+1} in the above hierarchy into a *coarse* subspace S_j and a *complement space* W_{j+1} that represents the *high frequencies.* Since it suffices to consider two successive spaces we simplify the notation and write $S_j =: S_H$, $S_{j+1} =: S_h$, $W_{j+1} =: W$ and accordingly Φ_H, Φ_h for the corresponding bases. The analogous quantities on the test side are denoted by $\tilde{S}_H$, $\tilde{S}_h$, $\tilde{W}$, $\tilde{\Phi}_H$, $\tilde{\Phi}_h$.

If we view the bases as vectors, the fact that test and trial spaces are nested implies that there are $\#\mathcal{I}_h \times \#\mathcal{I}_H$ matrices $\boldsymbol{P}_0, \tilde{\boldsymbol{P}}_0$ such that

$$\Phi_H^T = \Phi_h^T \boldsymbol{P}_0, \quad \tilde{\Phi}_H^T = \tilde{\Phi}_h^T \tilde{\boldsymbol{P}}_0. \tag{8}$$

Denoting by $\Psi = \{\psi_k : k \in \mathcal{J}\}$, and by $\tilde{\Psi} = \{\tilde{\psi}_k : k \in \mathcal{J}\}$ bases for the complement spaces $W, \tilde{W}$, respectively, the fact that $\Psi \subset S_h$ and $\tilde{\Psi} \subset \tilde{S}_h$ implies that there must be $\#\mathcal{I}_h \times \#\mathcal{J}$ matrices $\boldsymbol{P}_1, \tilde{\boldsymbol{P}}_1$, respectively, such that

$$\Psi^T = \Phi_h^T \boldsymbol{P}_1, \quad \tilde{\Psi}^T = \tilde{\Phi}_h^T \tilde{\boldsymbol{P}}_1. \tag{9}$$

The relations

$$S_h = S_H \oplus W, \quad \tilde{S}_h = \tilde{S}_H \oplus \tilde{W} \tag{10}$$

are equivalent to the fact that the composed matrices $\boldsymbol{P} := (\boldsymbol{P}_0, \boldsymbol{P}_1)$, $\tilde{\boldsymbol{P}} := (\tilde{\boldsymbol{P}}_0, \tilde{\boldsymbol{P}}_1)$ are invertible. Their inverses will be denoted by $\boldsymbol{R} = \binom{\boldsymbol{R}_0}{\boldsymbol{R}_1}$ and $\tilde{\boldsymbol{R}} = \binom{\tilde{\boldsymbol{R}}_0}{\tilde{\boldsymbol{R}}_1}$, respectively. For the complement bases to also be stable in the sense of (4) it is necessary and sufficient that

$$\|\boldsymbol{P}\|, \ \|\boldsymbol{R}\| \leq c \tag{11}$$

holds for some constant c independent of the current level; see [9]. Here $\|\cdot\|$ denotes the spectral norm. The matrices $\boldsymbol{P}_1$, $\tilde{\boldsymbol{P}}_1$ are called *completions* of $\boldsymbol{P}_0$, $\tilde{\boldsymbol{P}}_0$. The point is that choosing a complement space W is equivalent to finding a *completion* $\boldsymbol{P}_1$ of $\boldsymbol{P}_0$ to an invertible matrix. When (11) holds (uniformly with respect to the refinement level) then the completions are called (uniformly) *stable.*

Note that since $\Phi_H^T \boldsymbol{u}_H = \Phi_h^T(\boldsymbol{P}_0 \boldsymbol{u}_H) =: \Phi_h^T \boldsymbol{u}_h$ the matrix $\boldsymbol{P}_0$ (and likewise $\tilde{\boldsymbol{P}}_0$) represents a *prolongation operator.* Conversely, in view of $\Phi_h^T \boldsymbol{u}_h = \Phi_H^T(\boldsymbol{R}_0 \boldsymbol{u}_h) + \Psi^T(\boldsymbol{R}_1 \boldsymbol{u}_h)$, the matrix $\boldsymbol{R}_0$ acts as a *restriction operator.* The matrices $\boldsymbol{P}_1, \boldsymbol{R}_1$ (and likewise $\tilde{\boldsymbol{P}}_1, \tilde{\boldsymbol{R}}_1$) play analogous roles with respect to the complement spaces.

Hierarchical complements. As in [5] it will be of crucial importance that in the present context of Lagrange finite elements *hierarchical complement* spaces and corresponding completions are easily realized. In this case it is particularly convenient to adopt the convention to identify the *indices* in $\mathcal{I}_H$ with the *nodes* in the underlying mesh, i.e., one simply has to take $\mathcal{J} := \mathcal{I}_h \backslash \mathcal{I}_H$ and $\psi_k := \phi_{h,k}, k \in \mathcal{J}$ where $k \in \mathcal{I}_H$ or $k \in \mathcal{J}$ also denote points in Ω. In this case the matrices $\boldsymbol{P}_e$, $\boldsymbol{R}_e, e \in \{0,1\}$, are easily identified. In particular, they are very sparse, i.e., both prolongation and restriction operators are *local.*

Since this will frequently be referred to, we recall for the convenience of the reader some (well-known) details when the underlying trial spaces are standard continuous piecewise linear functions on a sequence of uniformly refined triangulations [35]. Denoting by $\mathcal{N}_k \subset \mathcal{J}$ the set of neighbors of $k \in \mathcal{I}_H$ in $\mathcal{I}_h$, one obviously has $\phi_{H,k} = \sum_{m\in\{k\}\cup\mathcal{N}_k} \frac{\phi_{H,k}(m)}{\|\phi_{h,m}\|_{L_\infty}}\phi_{h,m}$. One easily checks that

$$(12)\quad (\boldsymbol{P}_0)_{m,k} = \frac{\phi_{H,k}(m)}{\|\phi_{h,m}\|_{L_\infty}},\quad m\in\mathcal{I}_h,\ k\in\mathcal{I}_H,\qquad (\boldsymbol{P}_1)_{m,k} = \delta_{m,k},\quad m\in\mathcal{I}_h,\ k\in\mathcal{J}.$$

Moreover, the relation $\phi_{h,m} = 2\phi_{H,m} - \sum_{k\in\mathcal{N}_m}\frac{1}{2}\psi_{H,k}$ implies that

$$(13)\ (\boldsymbol{R}_0)_{k,m} = 2\delta_{k,m},\ m\in\mathcal{I}_h,\ k\in\mathcal{I}_H,\quad (\boldsymbol{R}_1)_{k,m} = \begin{cases} -\frac{1}{2}, & m\in\mathcal{I}_H, k\in\mathcal{N}_m,\\ \delta_{k,m}, & k,m\in\mathcal{J},\\ 0 & \text{else.}\end{cases}$$

One readily checks that (11) is satisfied in this case so that the hierarchical completions are stable in the above sense.

The importance of the above hierarchical completions is that they provide easily accessible *initial* completions from which other *problem adapted* completions will be derived. In this context the following remarks should be kept in mind. When S_H, S_h are both given then $\boldsymbol{P}_0$ is *determined.* It will sometimes be useful to change the point of view. Given S_h and *prescribing* $\boldsymbol{P}_0$ determines a *subspace* S_H of S_h spanned by the collection $\boldsymbol{P}_0^T\Phi_h$ regardless of whether S_h was originally obtained through a refinement process or not.

A key idea of the following development is to *adapt* the complements W (and hence restriction and prolongation operators) to the problem at hand. We will therefore make systematic use of the following *modification of stable completions* through coarse grid corrections [9, 13] of the initial hierarchical stable completion. For any $(\#\mathcal{I}_H)\times(\#\mathcal{J})$ matrix $\check{\boldsymbol{L}}$ we infer from $\boldsymbol{PR}=\boldsymbol{I}$ that

$$(14)\qquad \boldsymbol{I} = \boldsymbol{P}\begin{pmatrix}\boldsymbol{I} & \check{\boldsymbol{L}}\\ \boldsymbol{0} & \boldsymbol{I}\end{pmatrix}\begin{pmatrix}\boldsymbol{I} & -\check{\boldsymbol{L}}\\ \boldsymbol{0} & \boldsymbol{I}\end{pmatrix}\boldsymbol{R} =: \check{\boldsymbol{P}}\check{\boldsymbol{R}},$$

so that

$$(15)\qquad \check{\boldsymbol{P}}_0 = \boldsymbol{P}_0,\quad \check{\boldsymbol{P}}_1 = \boldsymbol{P}_0\check{\boldsymbol{L}} + \boldsymbol{P}_1,\quad \check{\boldsymbol{R}}_0 = \boldsymbol{R}_0 - \check{\boldsymbol{L}}\boldsymbol{R}_1,\quad \check{\boldsymbol{R}}_1 = \boldsymbol{R}_1.$$

Thus, one obtains a new complement space

$$(16)\qquad \check{W} := \operatorname{span}\check{\Psi},\quad \check{\Psi}^T := \Phi_h^T\check{\boldsymbol{P}}_1.$$

Retaining the prolongation $\boldsymbol{P}_0$ to S_h means that the original coarse space remains unchanged. Modifications of this sort will be referred to as *type* (I) modification and will be applied to the splitting of trial spaces.

In contrast to [7] we apply a *type* (II) modification to the test spaces. This time we observe that

$$\boldsymbol{I} = \tilde{\boldsymbol{R}}^T \tilde{\boldsymbol{P}}^T = \tilde{\boldsymbol{R}}^T \begin{pmatrix} \boldsymbol{I} & \hat{\boldsymbol{L}} \\ \boldsymbol{0} & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{I} & -\hat{\boldsymbol{L}} \\ \boldsymbol{0} & \boldsymbol{I} \end{pmatrix} \tilde{\boldsymbol{P}}^T =: \hat{\boldsymbol{R}}^T \hat{\boldsymbol{P}}^T, \tag{17}$$

which yields

$$\hat{\boldsymbol{R}}_0 = \tilde{\boldsymbol{R}}_0, \quad \hat{\boldsymbol{R}}_1 = \hat{\boldsymbol{L}}^T \tilde{\boldsymbol{R}}_0 + \tilde{\boldsymbol{R}}_1, \quad \hat{\boldsymbol{P}}_0 = \tilde{\boldsymbol{P}}_0 - \tilde{\boldsymbol{P}}_1 \hat{\boldsymbol{L}}^T, \quad \hat{\boldsymbol{P}}_1 = \tilde{\boldsymbol{P}}_1. \tag{18}$$

This time the restriction stays the same while the *prolongation changes*, i.e., the coarse space is now given by span $\hat{\boldsymbol{P}}_0^T \tilde{\Phi}_h$.

Later these modifications may be different for each level j. As long as the corresponding matrices $\hat{\boldsymbol{L}}$, $\check{\boldsymbol{L}}$ have uniformly bounded spectral norms, the corresponding modified bases stay stable [9].

3.2. Stable multiscale bases. The successive splitting of a fine scale trial space into coarser versions and complement spaces gives rise to *hierarchical representations* of the trial functions. The coefficients in such an expansion reflect the detail information needed to upgrade the coarse scale approximations. We will exploit this fact later for adaptive mesh refinement. However, this will require a tight interrelation between the norms of the whole coefficient arrays over all levels and the relevant function norm of the corresponding function. Unfortunately, the hierarchical basis representations based on the stable completions (12) and (13), although being uniformly stable on each individual level, do not give rise to such norm equivalences across levels for either the L_2-norm or the H^1-norm. However, suitable type (I) modifications of stable completions according to (15) allow one to efficiently generate new complement bases $\check{\Psi}_j = \{\check{\psi}_{j,k} : k \in \mathcal{J}_j\}$ with better stability properties. For instance, the coarse grid stabilization proposed in [31] offers one way of determining matrices $\check{\boldsymbol{L}}$ (see also [13] and [12]) to generate stable wavelets from hierarchical basis functions as initial stable completions through the type (I) modifications from (15). An alternative is offered in [9]; see also [23]. The corresponding matrices $\check{\boldsymbol{P}}_j, \check{\boldsymbol{R}}_j$ from (15) remain sparse in all cases.

We will therefore employ later for a posteriori local error control multiscale basis functions $\check{\psi}_{j,k}$ that are H^s-stable for some regularity range of $s \in [0, \alpha)$, $\alpha > 1$, in the sense that

$$c\|\boldsymbol{D}^s \check{\boldsymbol{d}}\|_{\ell_2} \le \left\| \sum_{k \in \mathcal{I}_{j_0}} u_{j_0,k} \phi_{j_0,k} + \sum_{j=j_0+1}^{\infty} \sum_{k \in \mathcal{J}_j} \check{d}_{j,k} \check{\psi}_{j,k} \right\|_{H^s(\Omega)} \le C \|\boldsymbol{D}^s \check{\boldsymbol{d}}\|_{\ell_2}, \tag{19}$$

where $\check{\boldsymbol{d}} = \{u_{j_0,k} : k \in \mathcal{I}_{j_0}\} \cup \{\check{d}_{j,k} : k \in \mathcal{J}_j, j \ge j_0\}$ and $\boldsymbol{D}$ is the diagonal matrix $(\boldsymbol{D})_{(j,k),(j',k')} = 2^j \delta_{(j,k),(j',k')}$. Specifically, in connection with multilinear shape functions $\phi_{j,k}$ on regular quadrilateral meshes, one can employ tensor product wavelets, which will be the choice in our numerical examples below. Their univariate versions are determined as follows: The columns of (the univariate) $\check{\boldsymbol{P}}_0$ are given by $\sqrt{2}/4, \sqrt{2}/2, \sqrt{2}/4$, those of $\check{\boldsymbol{P}}_1$ by $-\sqrt{2}/6, -\sqrt{2}/3, \sqrt{2}, -\sqrt{2}/3, -\sqrt{2}/6$ for interior wavelets, by $-7/12, 1/6, 1/12$ for the left boundary wavelet, and by $1/12, 1/6, -7/12$ for the right boundary wavelet. A typical row of $\check{\boldsymbol{R}}_0$ is given by $-\sqrt{2}/8, \sqrt{2}/4, 3\sqrt{2}/4, \sqrt{2}/4, -\sqrt{2}/8$, of $\check{\boldsymbol{R}}_1$ by $-3\sqrt{2}/16, 3\sqrt{2}/8, -3\sqrt{2}/16$ with the left and right boundary versions $-3/2, 3/4$, respectively, $3/4, -3/2$. In this case (19) holds for $\alpha = 3/2$.

3.3. Two-by-two block systems. Recall that the analogue to (6) for a current discretization level h reads

$$a_{\boldsymbol{\nu}}(\tilde{\Phi}_h, \Phi_h)\boldsymbol{u}_h = \boldsymbol{f}_h := (\tilde{\Phi}_h, f), \tag{20}$$

where $(u, v) := \int_\Omega u(x)v(x)dx$ is the standard inner product on Ω. This system is obviously equivalent to

$$\hat{\boldsymbol{P}}^T a_{\boldsymbol{\nu}}(\tilde{\Phi}_h, \Phi_h)\boldsymbol{P}\boldsymbol{R}\boldsymbol{u}_h = \hat{\boldsymbol{P}}^T \boldsymbol{f}_h, \tag{21}$$

where the matrices $\hat{\boldsymbol{P}}, \hat{\boldsymbol{R}}$ are defined by (18). We will think of the new system matrix $\hat{\boldsymbol{P}}^T a_{\boldsymbol{\nu}}(\tilde{\Phi}_h, \Phi_h)\boldsymbol{P}$ to be obtained in two steps. Starting with easily computable *initial* stable completions $\boldsymbol{P}_1$, $\tilde{\boldsymbol{P}}_1$ and the corresponding complement bases $\Psi, \tilde{\Psi}$ we have as in [5] the *initial blocking*

$$\tilde{\boldsymbol{P}}^T a_{\boldsymbol{\nu}}(\tilde{\Phi}_h, \Phi_h)\boldsymbol{P} := \begin{pmatrix} \mathbf{A}_{0,0} & \mathbf{A}_{0,1} \\ \mathbf{A}_{1,0} & \mathbf{A}_{1,1} \end{pmatrix}. \tag{22}$$

The two-by-two block matrix $\mathbf{A} = (\mathbf{A}_{i,l})_{i,l=0}^1$ is the stiffness matrix relative to the *initial* two-level bases $\Phi_H \cup \Psi$ and $\tilde{\Phi}_H \cup \tilde{\Psi}$ for the trial and test spaces according to the splitting (10). Setting briefly $\Psi^0 := \Phi_H$, $\Psi^1 := \Psi$ and $\tilde{\Psi}^0 := \tilde{\Phi}_H$, $\tilde{\Psi}^1 := \tilde{\Psi}$, one clearly has $\mathbf{A}_{i,l} := a_{\boldsymbol{\nu}}(\tilde{\Psi}^i, \Psi^l)$, $i, l = 0, 1$. Recalling from (14) and (17) how the modified completions are related to the initial stable completions, we may rewrite (21), in turn, in the form

$$\begin{pmatrix} \boldsymbol{I} & -\hat{\boldsymbol{L}} \\ \boldsymbol{0} & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \mathbf{A}_{0,0} & \mathbf{A}_{0,1} \\ \mathbf{A}_{1,0} & \mathbf{A}_{1,1} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_H \\ \boldsymbol{d} \end{pmatrix} = \begin{pmatrix} \hat{\boldsymbol{f}}_0 \\ \hat{\boldsymbol{f}}_1 \end{pmatrix}, \tag{23}$$

where $\boldsymbol{u}_H = \boldsymbol{R}_0\boldsymbol{u}_h$, $\boldsymbol{d} = \boldsymbol{R}_1\boldsymbol{u}_h$, $\hat{\boldsymbol{f}}_0 = \hat{\boldsymbol{P}}_0^T \boldsymbol{f}_h$, $\hat{\boldsymbol{f}}_1 = \hat{\boldsymbol{P}}_1^T \boldsymbol{f}_h$.

The choice of the stable completions for the trial spaces will always be the above hierarchical ones. In principle the test spaces may (and will) be different from the trial spaces. Let us postpone for the moment the discussion of the stable completions for the test spaces. Straightforward calculations show that the matrix on the left-hand side of (23) is given by

$$\hat{\boldsymbol{A}} = \begin{pmatrix} \mathbf{A}_{0,0} - \hat{\boldsymbol{L}}\mathbf{A}_{1,0} & \mathbf{A}_{0,1} - \hat{\boldsymbol{L}}\mathbf{A}_{1,1} \\ \mathbf{A}_{1,0} & \mathbf{A}_{1,1} \end{pmatrix} =: \begin{pmatrix} \hat{\boldsymbol{A}}_{0,0} & \hat{\boldsymbol{A}}_{0,1} \\ \hat{\boldsymbol{A}}_{1,0} & \hat{\boldsymbol{A}}_{1,1} \end{pmatrix}. \tag{24}$$

The upper left block in (24) represents the *coarse grid problem.*

As mentioned earlier, methods based on similar blockings have been studied extensively in the literature; see, e.g., [5, 4]. Specifically, the approach in [7, 6] is closely related. In the present terminology, a type (II) modification (17) is applied also to the trial spaces in order to annihilate both off diagonal blocks in (24); see also the discussion in [13] for the merits of these options.

3.4. The choice of $\hat{\boldsymbol{L}}$. Our main objective in exploiting the degrees of freedom offered by the matrices $\hat{\boldsymbol{L}}$ is to decouple the system (23) by (approximately) annihilating the upper right block $\hat{\boldsymbol{A}}_{0,1}$, i.e., choosing $\hat{\boldsymbol{L}}$ so that $\mathbf{A}_{0,1} = \hat{\boldsymbol{L}}\mathbf{A}_{1,1}$, thereby rendering $\hat{\boldsymbol{A}}$ block lower triangular. Thus, our first *ideal* choice would therefore be

$$\hat{\boldsymbol{L}} = \mathbf{A}_{0,1}\mathbf{A}_{1,1}^{-1}. \tag{25}$$

Except in the one-dimensional case where $\mathbf{A}_{1,1}$ is diagonal, the exact solution of the matrix equation (25) (which guarantees $\hat{\boldsymbol{A}}_{0,1} = \mathbf{0}$ with $\check{\boldsymbol{L}} = \mathbf{0}$) would be by far too expensive. In fact, except when $\mathbf{A}_{1,1}$ is diagonal, the matrix $\hat{\boldsymbol{L}}$ would be densely populated. Nevertheless, solving (25) *approximately* will later be seen to lead to a natural multigrid preconditioner.

Given an initial blocking $\mathbf{A}$ we wish to determine a *nearly decoupling* $\hat{\boldsymbol{L}}$ which satisfies (25) approximately, i.e.,

$$\hat{\boldsymbol{L}}\mathbf{A}_{1,1} \approx \mathbf{A}_{0,1}. \tag{26}$$

Note that the rows of $\hat{\boldsymbol{L}}$ can be determined independently of each other, which strongly supports parallelization. Of course, in this context it will be essential to quantify the meaning of $\approx$ in (26). The issue is to find a good compromise between efficiency and accuracy. As long as $\mathbf{A}_{1,1}$ is diagonally dominant (as in the univariate case and in general for moderate convection) the rows of $\hat{\boldsymbol{L}}$ can be determined efficiently with the aid of simple Jacobi iterations.

However, for more demanding problems, e.g., when $\nu \leq 2^{-8}$ while $\|\boldsymbol{\beta}\|$ is at least of the order 1, Jacobi iterations are no longer appropriate. A suitable approximate decoupling then gradually becomes the most demanding part of the whole concept. In this regime of very small diffusion any a-priorily chosen pattern for nonzero entries in $\hat{\boldsymbol{L}}$ turns out to fail. Therefore we have developed a suitable modification of the *sparse approximate inverse* (SPAI) *technique* from [17] where the nonzero pattern for $\hat{\boldsymbol{L}}$ is determined *adaptively.* The objective is to generate for each relevant index $k \in \mathcal{I}_H$, for which the kth row in $\mathbf{A}_{0,1}$ is not identically zero, an index set $\mathcal{P} = \mathcal{P}(k) \subset \mathcal{J}$, the index set of the current complement basis, and a vector $\boldsymbol{x} = \boldsymbol{x}(k)$ supported in $\mathcal{P}$ representing the kth row of $\hat{\boldsymbol{L}}$. $\mathcal{P}$ and $\boldsymbol{x}$ will be successively updated starting with some initialization. We refer to this scheme as $\mathbf{DEC}\,(\mathbf{A}) = \mathbf{DEC}\,(\mathbf{A}, \mathrm{Tol}) \to \hat{\boldsymbol{L}}$, where Tol measures the accuracy of the approximate decoupling; see [30] for a detailed description of this scheme. In fact, when $\boldsymbol{x}^T$ and $\boldsymbol{b}^T$ denote corresponding rows in $\hat{\boldsymbol{L}}$ and $\mathbf{A}_{0,1}$ the scheme continues until $\|\boldsymbol{x}^T\mathbf{A}_{1,1} - \boldsymbol{b}^T\|_{\ell_2}/\|\boldsymbol{b}\|_{\ell_2} \leq \mathrm{Tol}$. A detailed description with all distinctions from the original version in [17] can be found in [14]. The main distinctions result from our primary objective to minimize the size of $\mathcal{P}$ relative to the desired accuracy and to keep the number of least squares solutions as small as possible.

3.5. The basic multigrid iteration. Consider the transformed system $\hat{\boldsymbol{A}}\boldsymbol{u} = \hat{\boldsymbol{f}}$, where $\boldsymbol{u} = (\boldsymbol{u}_0^T, \boldsymbol{u}_1^T)^T = (\boldsymbol{u}_H^T, \boldsymbol{d}^T)^T$ (see (23)), $\hat{\boldsymbol{f}} = (\hat{\boldsymbol{f}}_0^T, \hat{\boldsymbol{f}}_1^T)^T$, and set $\boldsymbol{r}^\mu := \hat{\boldsymbol{f}} - \hat{\boldsymbol{A}}\boldsymbol{u}^\mu = (\boldsymbol{r}_0^\mu, \boldsymbol{r}_1^\mu)^T$. For the above *ideal* choice $\hat{\boldsymbol{A}}_{0,1} = \mathbf{0}$ implied by (25) block elimination applied to the block lower triangular matrix $\hat{\boldsymbol{A}}$ is equivalent to one iteration of the scheme

$$\boldsymbol{u}^{\mu+1} = \boldsymbol{u}^\mu + \left\{ \begin{pmatrix} \hat{\boldsymbol{A}}_{0,0}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} + \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ -\mathbf{A}_{1,1}^{-1}\mathbf{A}_{1,0}\hat{\boldsymbol{A}}_{0,0}^{-1} & \mathbf{A}_{1,1}^{-1} \end{pmatrix} \right\} \boldsymbol{r}^\mu \tag{27}$$

for any initial guess $\boldsymbol{u}^\mu$. The application of the first matrix on the right-hand side of (27) corresponds to a *coarse grid correction* and amounts to solving

$$\hat{\boldsymbol{A}}_{0,0}\boldsymbol{e}_0 = \boldsymbol{r}_0^\mu = \hat{\boldsymbol{f}}_0 - \hat{\boldsymbol{A}}_{0,0}\boldsymbol{u}_0^\mu, \tag{28}$$

which gives $\boldsymbol{u}_0^{\mu+1} = \boldsymbol{u}_0^\mu + \boldsymbol{e}_0$. The second matrix affects only the *high frequency part* of the iterates. Substituting (28), we see that its application is equivalent to

$$\boldsymbol{u}_1^{\mu+1} = \boldsymbol{u}_1^\mu + \mathbf{A}_{1,1}^{-1}\left(\boldsymbol{r}_1^\mu - \mathbf{A}_{1,0}\boldsymbol{e}_0\right), \tag{29}$$

which, in turn, amounts to solving

$$\mathbf{A}_{1,1}\boldsymbol{e}_1 = \boldsymbol{r}_1^{\mu} - \mathbf{A}_{1,0}\boldsymbol{e}_0, \tag{30}$$

to obtain $\boldsymbol{u}_1^{\mu+1} = \boldsymbol{u}_1^{\mu} + \boldsymbol{e}_1$.

Of course, the above scheme is idealized because it involves the *exact* inversion of matrices. However, an iterative two-grid scheme is obtained when viewing the first matrix on the right-hand side of (27) as a coarse grid correction and *approximating* the second matrix to define a *smoother* which can be written as a *basic iteration* of the form

$$\boldsymbol{u}^{\mu+1} = (\boldsymbol{I} - \boldsymbol{G}\hat{\boldsymbol{A}})\boldsymbol{u}^{\mu} + \boldsymbol{G}\boldsymbol{f} = \boldsymbol{u}^{\mu} + \boldsymbol{G}\boldsymbol{r}^{\mu},$$

where

$$\boldsymbol{G} = \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \boldsymbol{M} & \boldsymbol{Q} \end{pmatrix}, \quad \left(\boldsymbol{I} - \boldsymbol{G}\hat{\boldsymbol{A}}\right) = \begin{pmatrix} \boldsymbol{I} & \mathbf{0} \\ -\left(\boldsymbol{M}\hat{\boldsymbol{A}}_{0,0} + \boldsymbol{Q}\mathbf{A}_{1,0}\right) & \boldsymbol{I} - \boldsymbol{Q}\mathbf{A}_{1,1} \end{pmatrix}, \tag{31}$$

and

$$\boldsymbol{Q} \approx \mathbf{A}_{1,1}^{-1}, \quad \boldsymbol{M} \approx -\boldsymbol{Q}\mathbf{A}_{1,0}\hat{\boldsymbol{A}}_{0,0}^{-1}. \tag{32}$$

Thus the *high frequency part* of the error is reduced when

$$\left\| - \left(\boldsymbol{M}\hat{\boldsymbol{A}}_{0,0} + \boldsymbol{Q}\mathbf{A}_{1,0}\right)\boldsymbol{e}_0 + (\boldsymbol{I} - \boldsymbol{Q}\mathbf{A}_{1,1})\,\boldsymbol{e}_1 \right\| < \eta\|\boldsymbol{e}_1\| \tag{33}$$

holds for some $\eta < 1$ and a suitable norm $\|\cdot\|$.

Note that one application of the smoother (31) takes the form

$$\boldsymbol{u}^{\mu+1} = \begin{pmatrix} \boldsymbol{u}_0^{\mu} \\ \boldsymbol{u}_1^{\mu} + \boldsymbol{e}_1 \end{pmatrix}, \quad \boldsymbol{e}_1 = \boldsymbol{M}\boldsymbol{r}_0^{\mu} + \boldsymbol{Q}\boldsymbol{r}_1^{\mu}.$$

The coarse grid correction (28) as well as the smoothing step require solving a system with matrix $\hat{\boldsymbol{A}}_{0,0}$. Note that $\hat{\boldsymbol{A}}_{0,0}$ can be viewed as an *approximate Schur complement* of $\mathbf{A}$ with respect to the complement block $\mathbf{A}_{1,1}$; see [13]. One immediately obtains now a multigrid scheme by applying the same procedure to $\hat{\boldsymbol{A}}_{0,0}$ and so on. To do this we need to specify the initial blocking also for subsequent coarser levels. In the subsequent numerical experiments we have employed the following choice: The initial complement functions $\Psi = \Psi_j$ on the trial side are always the hierarchical ones, that is, when H stands for the jth level, $\boldsymbol{P}_1 = \boldsymbol{P}_{j,1}$ is given by (12). Likewise when h corresponds to the currently highest level we let $\tilde{\Phi}_h = \Phi_h = \Phi_{j+1}$ and $\tilde{\Psi}_{j+1} = \Psi_{j+1}$, i.e., $\tilde{\boldsymbol{P}}_j = \boldsymbol{P}_j = (\boldsymbol{P}_{j,0}, \boldsymbol{P}_{j,1})$. Although on lower levels j, due to the dual modification (18) applied to the test side, the coarse test spaces are no longer spanned by standard hat functions, we can always use the *same* stable completions for the initial blocking of the coarse system $\hat{\boldsymbol{A}}_{0,0}^{j}$ defined by $\boldsymbol{P}_{j-1}^{T}\hat{\boldsymbol{A}}_{0,0}^{j}\boldsymbol{P}_{j-1}$, where $\boldsymbol{P}_{j-1}$ is given by (12). By (11) the corresponding initial complement functions always form uniformly stable bases in the sense of (4).

Recall from [13] that in the univariate case the complement block $\mathbf{A}_{1,1}$ is diagonal and hence easily invertible so that $\hat{\boldsymbol{L}}$ according to (25) can be computed exactly. The following figures illustrate that this choice of $\hat{\boldsymbol{L}}$ produces more and more deformed basis functions for the test spaces when descending to lower levels. In case of dominating convection this has a clear *upwind* effect which in the limit appears to reduce to an upwind finite difference scheme; see Figure 1. In fact, it is known that in the univariate case (with $\check{\boldsymbol{L}} = \mathbf{0}$) decoupling is equivalent to employing L-splines. Figure 2 illustrates the analogous effect for zero convection and large reaction. It would be interesting to relate this to L-splines and so-called exact discretizations [34, 28].

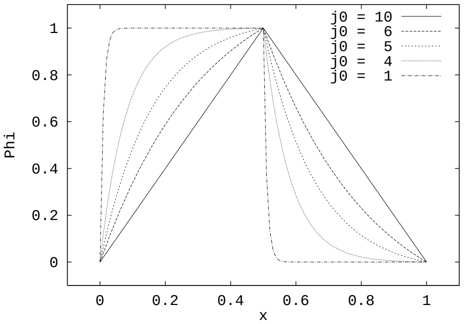


FIG. 1. $\nu = 1$, $\beta = 100$, $\gamma = 0$.

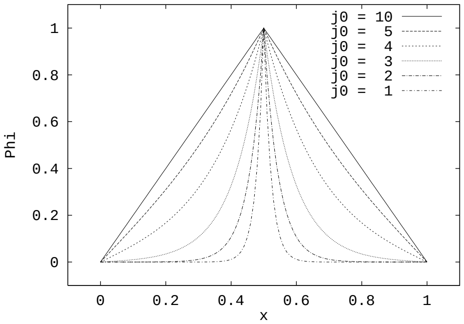


FIG. 2. $\nu = 1$, $\beta = 0$, $\gamma = 2000$.

We can now describe the resulting basic multigrid iteration. It refers to a hierarchy $\mathcal{S}$ of nested trial spaces (which will later not necessarily arise from *uniform* refinements). As before j denotes the current discretization level.

$\mathbf{MG}\,(j, \mathcal{S}, \mathbf{A}_j, \boldsymbol{f}_j, \boldsymbol{u}_j^{\mu}) \rightarrow \boldsymbol{u}_j^{\mu+1}$.

- **Input:** $\mathbf{A}_j := a_{\boldsymbol{\nu}}(\tilde{\Phi}_j, \Phi_j)$ and a current approximation $\boldsymbol{u}_j^{\mu}$ to the solution of the system $\mathbf{A}_j \boldsymbol{u}_j = \boldsymbol{f}_j$.
- **Initial blocking:** Compute $(\mathbf{A}_{i,l})_{i,l=0}^{1} := \boldsymbol{P}^T \mathbf{A}_j \boldsymbol{P}$ where $\boldsymbol{P} := \boldsymbol{P}_{j-1}$ is induced by the initial hierarchical decomposition (10). (In the case of uniform refinements one can, for instance, take $\boldsymbol{P}$ defined by (12).)
- **Decoupling:** Determine $\check{\boldsymbol{L}}$, apply $\mathbf{DEC}\,(\mathbf{A}) \rightarrow \hat{\boldsymbol{L}}$. Compute the matrix $\hat{\boldsymbol{A}}$ according to (24) as well as

$$\begin{pmatrix} \boldsymbol{f}_0 \\ \boldsymbol{f}_1 \end{pmatrix} = \hat{\boldsymbol{P}}_j^T \boldsymbol{f}_j, \quad \begin{pmatrix} \boldsymbol{u}_0 \\ \boldsymbol{u}_1 \end{pmatrix} = \boldsymbol{R}_j \boldsymbol{u}_j^{\mu}.$$

- **Smoothing and coarse grid correction:**
  (1) Compute $\boldsymbol{r}_0 := \boldsymbol{f}_0 - \hat{\boldsymbol{A}}_{0,0}\boldsymbol{u}_0 - \hat{\boldsymbol{A}}_{0,1}\boldsymbol{u}_1$, $\boldsymbol{r}_1 := \boldsymbol{f}_1 - \hat{\boldsymbol{A}}_{1,0}\boldsymbol{u}_0 - \hat{\boldsymbol{A}}_{1,1}\boldsymbol{u}_1$; and if $j > j_0 + 1$ for a coarsest level j_0 apply $\mathbf{MG}\,(j-1, \mathcal{S}, \hat{\boldsymbol{A}}_{0,0}, \boldsymbol{r}_0, \mathbf{0}) \rightarrow \boldsymbol{e}_0'$ to obtain an approximate solution of the coarse grid problem

$$\hat{\boldsymbol{A}}_{0,0}\boldsymbol{e}_0 = \boldsymbol{r}_0; \tag{34}$$

  else apply a direct solver to (34).
  (2) Determine an approximate solution $\boldsymbol{e}_1'$ of the complement system

$$\mathbf{A}_{1,1}\boldsymbol{e}_1 = \boldsymbol{r}_1 - \mathbf{A}_{1,0}\boldsymbol{e}_0' \tag{35}$$

  and compute $\boldsymbol{u}_0' := \boldsymbol{u}_0 + \boldsymbol{e}_0'$, $\boldsymbol{u}_1' = \boldsymbol{u}_1 + \boldsymbol{e}_1'$.
  (3) Set

$$\boldsymbol{u}_j^{\mu+1} := (\boldsymbol{P}_{j,0}, \boldsymbol{P}_{j,1}) \begin{pmatrix} \boldsymbol{u}_0' \\ \boldsymbol{u}_1' \end{pmatrix}. \tag{36}$$

A few comments on the above scheme are in order. Of course, the rationale behind the above scheme is that it is essentially a perturbation of the ideal scheme from section 3.5. In fact, since the matrix $\hat{\boldsymbol{L}}$ produced by **DEC** does not satisfy (25) exactly the upper right block $\hat{\boldsymbol{A}}_{0,1}$ does not vanish. Moreover, (34) and (35) mean that the operators $\boldsymbol{M}, \boldsymbol{Q}$ in the smoother (31) are only approximations to the ideal choices in (32). Since these ideal choices would, according to (33), annihilate the high frequency part of the error completely the damping of this high frequency part and hence the convergence properties of the iterative scheme will depend on the accuracy of the approximations to $\hat{\boldsymbol{L}}$, $\boldsymbol{M}$, and $\boldsymbol{Q}$. By recursion this affects also the solution in (34) and will produce a better reduction in (33). Thus the number of multigrid cycles can be kept small at the expense of investing more computational effort, e.g., in the computation of $\hat{\boldsymbol{L}}$ or the execution of $\boldsymbol{M}$ and $\boldsymbol{Q}$. In this context note that the above multigrid iteration has V-cycle structure. It would be straightforward to formulate smoothing and coarse grid correction in terms of W-cycles. However, our numerical tests have shown that the number of cycles in either mode depends essentially on the quality of the approximate decoupling and the smoother. So far this is done by adapting the accuracy of **DEC** (**A**). Of course, it would be highly desirable to have a robust automatic procedure for adapting the involved tuning parameters, which is still subject to further research; see the discussion in section 5.

4. Adaptive full multigrid.

The objective is to design an adaptive refinement strategy which, as long as layers are to be *resolved*, in combination with the above multigrid scheme not only reduces computational cost and storage but also *stabilizes* the numerical solution without the need of an a priori stabilization technique. We proceed now describing the main ingredients.

4.1. Locally refined trial spaces.

Instead of working with an a priori fixed sequence of *uniformly refined* trial spaces we will seek for approximate solutions of (1) in a suitably adapted hierarchy of *subspaces* $S_j^\circ \subset S_j$. To describe the underlying setting let j_0 denote the coarsest level, i.e., S_{j_0} is the usual finite element space with respect to a (quasi-) uniform triangulation of Ω. As before each of the (full) spaces S_j is spanned by a basis $\Phi_j = \{\phi_{j,k} : k \in \mathcal{I}_j\}$ of Lagrange type, i.e., identifying $\mathcal{I}_j$ with the nodes, we have

$$\phi_{j,k}(k')/\|\phi_{j,k}\|_{L_\infty(\Omega)} = \delta_{k,k'}, \quad k,k' \in \mathcal{I}_j. \tag{37}$$

We will describe now how to generate larger spaces that are not necessarily spanned by *all* basis functions of a given level. Instead we make use of the refinement equation (8) which for each individual coarse basis function reads

$$\phi_{j,k} = \sum_{l\in\mathcal{I}_{j+1,k}} (\boldsymbol{P}_{j,0})_{l,k}\phi_{j+1,l}, \tag{38}$$

where $\mathcal{I}_{j+1,k}$ is the support of the kth column of $\boldsymbol{P}_{j,0}$, i.e., $\mathcal{I}_{j+1,k} = \{l \in \mathcal{I}_{j+1} : (\boldsymbol{P}_{j,0})_{l,k} \neq 0\}$.

It will be convenient to characterize the locally refined trial spaces by *grids* or better yet by collections of nodes that will eventually be nonuniformly spaced. These collection of nodes are generated as follows.

- **Initialization:** Let $\mathcal{G} := \{(j_0,k) : k \in \mathcal{I}_{j_0}\}$.
- For a given $\mathcal{G}$ a refinement step consists of selecting some set $\mathcal{R} \subseteq \mathcal{G}$ and setting

  $$(\mathcal{G}\setminus\mathcal{R}) \cup \mathcal{G}_\mathcal{R} \to \mathcal{G} \quad \text{where} \quad \mathcal{G}_\mathcal{R} := \{(l+1,m) : (l,k) \in \mathcal{R},\ m \in \mathcal{I}_{l+1,k}\}. \tag{39}$$

> Thus one marks a set of nodes in the current set and replaces them by those nodes on the next finer level which correspond to the fine grid basis functions needed to represent the marked coarse grid basis functions through (38).

Note that at a given stage the set $\mathcal{R}$ could contain pairs (l,k) from *different* levels l. Of course, associating with each $(l,k)\in\mathcal{G}$ the basis function $\phi_{l,k}$ the corresponding piecewise polynomial structure induces a *mesh* which is generally nonuniform and contains *hanging nodes.* This is illustrated in Figures 3 and 4. The solid circles (●) in Figure 3 form a typical set $\mathcal{R}$ of coarse grid nodes that are to be refined. The coarse grid basis functions marked by (○) remain unchanged. Figure 4 displays the refined nodes (•) replacing the elements of $\mathcal{R}$.

Defining $S_{\mathcal{G}} := \operatorname{span}\{\phi_{j,k} : (j,k)\in\mathcal{G}\}$, and given any refinement $\mathcal{G}'$ of $\mathcal{G}$ in the above sense, it is clear then that $S_{\mathcal{G}}\subset S_{\mathcal{G}'}$ and $\Phi_{\mathcal{G}} := \{\phi_{j,k} : (j,k)\in\mathcal{G}\}$ is a basis of $S_{\mathcal{G}}$.

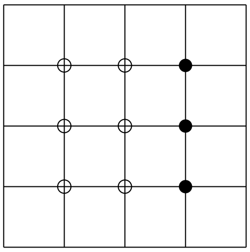


FIG. 3. *Nodes to be refined.*

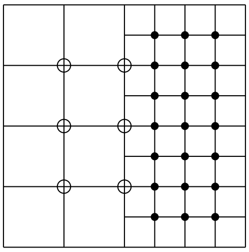


FIG. 4. *Refined grid.*

It will be useful to decompose $\mathcal{G}$ in its contributions to different levels. Let $J := \max\{j : (j,k)\in\mathcal{G}\}$ and define

$$\mathcal{I}_j^- := \{k : (j,k)\in\mathcal{G}\} \quad \text{for } j = J,\dots,j_0.$$

Obviously one has $\mathcal{G} = \bigcup_{j=j_0}^J\{(j,k) : k\in\mathcal{I}_j^-\}$ and $\mathcal{I}_j^-\cap\mathcal{I}_l^- = \emptyset$ for $j\neq l$. Every element $u_J\in S_{\mathcal{G}}$ has a unique representation

$$u_J = \sum_{j=j_0}^{J}\sum_{k\in\mathcal{I}_j^-} u_{j,k}^{\circ}\phi_{j,k}. \tag{40}$$

This representation involves the standard shape functions on the *locally* highest level of current resolution. The coarse grid matrix will be represented with respect to this basis because it allows us best to exploit sparseness.

4.2. Local multiscale transformations. In addition to the nodal basis representation of trial functions we will have to make use of two-level or even multilevel representations in terms of complement basis functions of the form (9). They are determined by stable completions represented by matrix pairs $\boldsymbol{P}^\circ, \boldsymbol{R}^\circ$. Here $\boldsymbol{P}^\circ$ will stand for either hierarchical complements $\boldsymbol{P}^\circ = \boldsymbol{P}$ or for stable bases in the sense of (19) $\boldsymbol{P}^\circ = \check{\boldsymbol{P}}$. Recall that in all cases both $\boldsymbol{P}^\circ$ and $\boldsymbol{R}^\circ$ are sparse.

In principle, the structure of transformations between representations in terms of nodal functions and complement functions is well known for uniform refinements. The only issue here is to realize such transformations for the locally refined case without unnecessarily increasing the complexity. To explain the essence of the matter, recall from section 3.1 that $\Phi_h^T\boldsymbol{u}_h = \Phi_H^T(\boldsymbol{R}_0^\circ\boldsymbol{u}_h) + \Psi^T(\boldsymbol{R}_1^\circ\boldsymbol{u}_h)$, which splits a fine scale representation into a coarse scale representation and a complement part determined by the given stable completion. Now suppose that J is the highest level occurring in a locally refined space as described above. Thus the support $\mathcal{I}_J^-$ could be much smaller than the full set of nodes on level J. By the above splitting, the coarse coefficients and complement coefficients are given by

$$\boldsymbol{u}_J^\circ \rightarrow \left\{ \begin{array}{lcl} \boldsymbol{R}_0^\circ \boldsymbol{u}_J^\circ & =: & \hat{\boldsymbol{u}}_{J-1}, \\ \boldsymbol{R}_1^\circ \boldsymbol{u}_J^\circ & =: & \boldsymbol{d}_J^\circ, \end{array} \right. \tag{41}$$

where these matrix/vector multiplications are carried out in a sparse format. That is, only the active coefficients in $\boldsymbol{u}_J^\circ$ select the corresponding active columns of $\boldsymbol{R}^\circ$ so that, due to the sparseness of $\boldsymbol{R}^\circ$, the computational work stays proportional to $\#\mathcal{I}_J^-$. The array $\boldsymbol{d}_J^\circ$ with support $\mathcal{J}_J^\circ$ are the *wavelet coefficients* on the highest level and will remain untouched in subsequent steps. The coarse scale coefficients $\hat{\boldsymbol{u}}_{J-1}$ will be added (if applicable) to the coefficients $\boldsymbol{u}_{J-1}^\circ$ to form the resulting contribution $\bar{\boldsymbol{u}}_{J-1} := \hat{\boldsymbol{u}}_{J-1} + \boldsymbol{u}_{J-1}^\circ$ with support $\bar{\mathcal{I}}_{J-1}$ on level $J-1$.

In the above multigrid scheme this two-level splitting already suffices. Below we will also need a full transformation involving all lower levels. This means the above procedure will be successively applied to the arrays $\bar{\boldsymbol{u}}_j$ for $j < J$, each time splitting off the wavelet coefficients $\boldsymbol{d}_j^\circ$ with supports $\mathcal{J}_j^\circ$. This results in the alternative representation

$$u_J = \sum_{k \in \bar{\mathcal{I}}_{j_0}} \bar{u}_{j_0,k} \phi_{j_0,k} + \sum_{j=j_0+1}^{J} \sum_{k \in \mathcal{J}_j^\circ} d_{j,k}^\circ \psi_{j,k}^\circ \tag{42}$$

of the function u_J in (40). Hence, denoting by

$$\boldsymbol{u}_{\mathcal{G}}^J := ((\boldsymbol{u}_{j_0}^\circ)^T, (\boldsymbol{u}_{j_0+1}^\circ)^T, \ldots, (\boldsymbol{u}_J^\circ)^T)^T, \quad \boldsymbol{u}_{ms}^J := ((\bar{\boldsymbol{u}}_{j_0})^T, (\boldsymbol{d}_{j_0+1}^\circ)^T, \ldots, (\boldsymbol{d}_J^\circ)^T)^T$$

the coefficient arrays in (40) and (42), respectively, the transformation $\boldsymbol{T}_{\mathcal{G}}^\circ : \boldsymbol{u}^J \rightarrow \boldsymbol{u}_{ms}^J$ can be carried out by successively applying (41). It is clear that the number of operations required by this transformation remains proportional to $\#\mathcal{G}$.

Likewise, an analogous local successive application of the steps

$$\boldsymbol{P}_{j-1,0}^\circ \bar{\boldsymbol{u}}_{j-1} + \boldsymbol{P}_{j-1,1}^\circ \boldsymbol{d}_j^\circ = \bar{\boldsymbol{u}}_j \tag{43}$$

realizes the transformation $(\boldsymbol{T}_{\mathcal{G}}^\circ)^{-1} : \boldsymbol{u}_{ms}^J \rightarrow \boldsymbol{u}_{\mathcal{G}}^J$ also at an expense of $\mathcal{O}(\#\mathcal{G})$ flops. The reader is referred to [15] for the details of these transformations.

The full multiscale transformation will be needed only for the adaptive mesh refinement and thus involves $\boldsymbol{P}^\circ = \check{\boldsymbol{P}}$. However, the corresponding transformations $\check{\boldsymbol{T}}_{\mathcal{G}}, \check{\boldsymbol{T}}_{\mathcal{G}}^{-1}$ can be efficiently generated through the hierarchical basis transformations and (15). Therefore we briefly indicate the simple interrelation between the sets $\mathcal{G}$ and $\{\bar{\mathcal{I}}_{j_0}, \mathcal{J}_{j_0+1}^\circ, \ldots, \mathcal{J}_J^\circ\}$ in the special case $\boldsymbol{P}^\circ = \boldsymbol{P}$. To describe this let $\mathcal{I}_J^+ := \mathcal{I}_J^-$ and for $j_0 < j \leq J$

$$\mathcal{J}_j^\circ := \mathcal{I}_j^+ \setminus \mathcal{I}_{j-1}, \quad \mathcal{I}_{j-1}^+ := \mathcal{I}_{j-1}^- \cup \left(\mathcal{I}_j^+ \cap \mathcal{I}_{j-1}\right). \tag{44}$$

Thus there is a simple one-to-one correspondence between the set $\mathcal{G}$ and the set $\mathcal{J} := \{(j_0, k) : k \in \mathcal{I}_{j_0}\} \cup \bigcup_{j=j_0+1}^{J} \{(j,k) : k \in \mathcal{J}_j^\circ\}$. It is important to bear in mind that both sets $\mathcal{G}$ and $\mathcal{J}$ contain index *pairs* (j,k) where only the second index k refers to a spatial location as a node. The same node k may have different level indices j,l depending on the data structure context $(l,k)\in\mathcal{G}$ or $(j,k)\in\mathcal{J}$.

4.3. Refinement strategy. The norm equivalence (19) offers new ways of estimating errors. To see this suppose $u_J = \sum_{k\in\mathcal{I}_{j_0}} u_{j_0,k}\phi_{j_0,k} + \sum_{j=j_0+1}^{J}\sum_{k\in\mathcal{J}_j}\check{d}_{j,k}\check{\psi}_{j,k}$ is a current approximation to u. Among the many possible examples, consider the following thresholding scheme for coarsening a given approximation:

(i) Fix a tolerance $\eta > 0$.

(ii) For $j = j_0+1,\dots,J$, find the largest $\tilde{\mathcal{J}}_j \subset \mathcal{J}_j$ such that for some fixed $\alpha > 0$ one has $\sum_{k\in\tilde{\mathcal{J}}_j} |\check{d}_{j,k}|^2 \le 2^{\alpha(j-J)}\eta^2$ and set

$$\tilde{d}_{j,k} := \begin{cases} \check{d}_{j,k} & \text{if} \quad k\in\mathcal{J}_j\setminus\tilde{\mathcal{J}}_j; \\ 0 & \text{if} \quad k\in\tilde{\mathcal{J}}_j. \end{cases}$$

Clearly, by (19), $\tilde{u}_J := \sum_{(j,k)\in\mathcal{J}\setminus\tilde{\mathcal{J}}}\check{d}_{j,k}\check{\psi}_{j,k}$ then satisfies $\|u_J - \tilde{u}_J\|_{L_2(\Omega)} \le c\eta$, where c depends only on the constants in (19) for $s=0$. Applying an analogous procedure to $2^j\check{d}_{j,k}$ yields estimates in H^1. This strategy imposes higher accuracy for low-level coefficients. Alternatively one could determine the largest $\tilde{\mathcal{J}} \subset \mathcal{J}$ such that $\sum_{(j,k)\in\tilde{cJ}} |\check{d}_{j,k}|^2 \le \eta^2$ to arrive at the same conclusion. A detailed account of related strategies for elliptic problems is given in [11].

Conversely, denoting by $\partial\mathcal{J}$ those indices reflecting a locally highest level, elements in $\partial\mathcal{J}\cap(\mathcal{J}\setminus\tilde{\mathcal{J}})$ may be taken to determine the next refinement (again variants are possible). The corresponding modifications of the meshes $\mathcal{G}$ will be described in the routines below.

We will now describe how the collections $\mathcal{G}$ and corresponding trial spaces $S_{\mathcal{G}}$ emerge through a refinement strategy by monitoring the multiscale coefficients $\boldsymbol{u}^J_{ms}$ based on the above considerations. Since the coefficients of the complement functions $\check{\psi}_{j,k}$ represent high scale fluctuations, whenever for some $(j,k)\in\partial\mathcal{G}$ the coefficient $|d_{j,k}|$ is large, this indicates that u^J exhibits strong oscillations of locally high frequency. Such oscillations may be caused by (locally) large grid-Péclet numbers or by the fact that the current approximation is locally not yet accurate enough. In *both* cases a refinement is needed. In particular, the approximate solutions of the complement problem (35) can be used to control local refinements. Hence a natural refinement strategy is to refine the trial spaces at locations where even *after smoothing* such high oscillations are still exhibited by the current approximate solution. It will be seen later that due to the nature of the full multigrid scheme below, it will be necessary to coarsen a space. Moreover, it will simplify data management significantly when working only with *graded* sets $\mathcal{G}$. This means that each edge in the mesh contains at most one hanging node, i.e., the levels of neighboring mesh points differ at most by one. It turns out that this is a very mild restriction which is satisfied in most cases automatically. We describe next a scheme that combines all three tasks: coarsening, refining, and grading. Its input are the current grid $\mathcal{G}$, the set $\mathcal{R}$ marking the last refinement in a prior step (which is needed to organize the coarsening), the array $\boldsymbol{u}^J_{ms}$ of wavelet coefficients with highest level J of the current solution, a vector $\boldsymbol{\tau} = (\tau_{j_0+1},\dots,\tau_J)$ of tolerances for refinement, and a corresponding vector $\boldsymbol{\tau}'$ for coarsening. The latter ones should be somewhat smaller. A typical choice is $\tau'_j = \tau_j/2$. The choice of the τ_j will be discussed later in connection with applications; see section 4.4.

REF-COARSE ($\mathcal{G}$, $\mathcal{R}$, $\boldsymbol{u}^J_{ms}$ $\boldsymbol{\tau}$, $\boldsymbol{\tau}'$) → ($\mathcal{R}$, $\mathcal{G}$).

```
• $\mathcal{R}' := \mathcal{R}$
• for $j = J, J-1, \ldots, j_0+1$
      $\mathcal{K}_{j+1} := \{k \in \mathcal{J}_{j+1} \ : \ \check{d}_{j+1,k} \neq 0\}$
      for $(j,k) \in \mathcal{R}'$
          if $(j+1,l) \notin \mathcal{R}' \ \forall\, l \in \mathcal{I}_{j+1,k}$
             $\mathcal{K}^G_l := \{l' \in \mathcal{J}_{j+1} \ : \ g^{j,1}_{l',l} \neq 0\}$
             if $|\check{d}_{j+1,l'}| < \tau'_{j+1} \ \forall\, l' \in \mathcal{K}_{j+1} \cap \bigcup_{l \in \mathcal{I}_{j+1,k}} \mathcal{K}^G_l$
                $\mathcal{R} := \mathcal{R} \setminus \{(j,k)\}$
      for $k \in \mathcal{I}^-_j \cap \mathcal{J}_j$
          if $|\check{d}_{j,k}| > \tau_j$
             $\mathcal{N}_j(k)$ := neighbors of $k$ and their neighbors on level $j$
             $\mathcal{R} := \mathcal{R} \cup \{(j,l) : \ l \in \mathcal{N}_j(k)\}$
• for $j = J, J-1, \ldots, j_0+2$
      for $(j,k) \in \mathcal{R}$
          $\mathcal{R} := \mathcal{R} \cup \{(j-1,l) : \text{supp } \phi_{j-1,l} \cap \text{supp } \phi_{j,k} \neq \emptyset\}$
• $\mathcal{I}^+_{j_0+1} = \mathcal{I}_{j_0+1}$
• $\mathcal{G} := \emptyset$
• for $j = j_0+1, j_0+2, \ldots, J$
      $\mathcal{I}^+_{j+1} := \emptyset$
      for $k \in \mathcal{I}^+_j$
          if    $(j,k) \in \mathcal{R}$
                $\mathcal{I}^+_{j+1} := \mathcal{I}^+_{j+1} \cup \mathcal{I}_{j+1,k}$
          else $\mathcal{G} := \mathcal{G} \cup \{(j,k)\}$
• $\mathcal{G} := \mathcal{G} \cup \{(j+1,k) : k \in \mathcal{I}^+_{j+1}\}$
```

4.4. The main algorithm. We have by now collected the main ingredients that will enter the adaptive full multigrid scheme. The basic idea may be sketched as follows. We begin on a coarse mesh and raise the viscosity so that the grid-Péclet number is approximately one. This means that the solution to the modified problem starts to exhibit oscillations while the above multigrid scheme is still efficient. A (possibly local) refinement based on the multiscale representation of the current solution is then made according to the above refinement strategy. The viscosity is reduced by a factor two and the (prolonged) current solution is used as an initial guess for the problem with lowered viscosity on the refined space. Note that the viscosity is halved *globally* even when the refinement is only *local.* This procedure is repeated until one either arrives at a prescribed highest level J_{max} of highest local resolution or until the original viscosity is restored. The multigrid iteration on this final mesh proceeds then until a stopping criterion is met. Thus in the course of nested iteration the original problem is approached through a sequence of auxiliary problems.

Before describing the algorithmic details a few preparatory remarks are necessary. By $\|\cdot\|_X$ we denote the norm in which the accuracy is to be measured. We will consider here only $X = L_2(\Omega)$ and $X = H^1_0(\Omega)$, but other choices are conceivable. The threshold parameters $\boldsymbol{\tau}, \boldsymbol{\tau}'$ appearing in **REF-COARSE** have the form

$$\tau_j = \kappa \, \|u_J\|_\infty \, 2^{(j-J)d/2}, \quad \tau'_j \leq \tau_j, \tag{45}$$

where κ is a constant controlling the overall accuracy of the approximate solutions. Note that the tolerances τ_j require higher relative accuracy on lower levels. In fact, in our experiments level-independent thresholds always produced worse results. We use a fixed choice for the matrices $\check{\boldsymbol{L}}$ controlling the splittings of the trial spaces according either to (Ia) or to (Ib) described in section 3.4. The multiscale transformation $\boldsymbol{T}_{\mathcal{G}}$ appearing below corresponds to the wavelet basis used in **REF-COARSE**; see sections 3.2 and 4.3. Finally, tol represents the target accuracy for the discrete systems and μ is the number of iterations in **MG**.

```
FMG $(f, \mu, \mathrm{tol}) \to \boldsymbol{u}$.
(1) Choose a coarsest level $j_0$, set $J = j_0 + 1$, choose the smallest $L$ such that
    $P_J(2^L\nu) < 1$, and set $\nu_J := 2^L\nu$. Let $\mathcal{G} = \mathcal{I}_{j_0+1}$, $\mathcal{R} = \emptyset$. Fix $J_{max} > L$ and
    compute
    $\mathbf{A}_J := a_{\nu_J}(\Phi_J, \Phi_J), \quad \boldsymbol{f}_J := (\Phi_J, f), \quad \mathcal{S} = \{\mathrm{span}\,\Phi_{j_0}, \mathrm{span}\,\Phi_{j_0+1}\}, \; \boldsymbol{u}_J = \mathbf{0}.$
(2) Set $\boldsymbol{u}_J^{(0)} = \boldsymbol{u}_J$.
    For $i = 1, \dots, \mu$ apply MG $(J, \mathcal{S}, \mathbf{A}_J, \boldsymbol{f}_J, \boldsymbol{u}_J^{(i-1)}) \to \boldsymbol{u}_J^{(i)}$.
    Set $\boldsymbol{u}'_J = \boldsymbol{u}_J^{(\mu)}$, $\boldsymbol{u}_J = \boldsymbol{u}_J^{(\mu-1)}$.
(3) If $J = J_{max}$
        compute $e_J := \|\boldsymbol{u}_J - \boldsymbol{u}'_J\|_X$.
        If $e_J \le$ tol accept $\boldsymbol{u}'_J$ as solution, set $\boldsymbol{u} := \boldsymbol{u}'_J$ and STOP. Else
        Set $\mu = 1$.
        Replace $\boldsymbol{u}_J$ by $\boldsymbol{u}'_J$ and go to (2).
(4) Compute $\boldsymbol{u}'_{ms} := \boldsymbol{T}_{\mathcal{G}}\boldsymbol{u}'_J$ and $\boldsymbol{\tau}, \boldsymbol{\tau}'$ based on $\boldsymbol{u}'_J$ according to (45).
(5) Apply REF-COARSE $(\mathcal{G}, \mathcal{R}, \boldsymbol{u}'_{ms}, \boldsymbol{\tau}, \boldsymbol{\tau}',) \to (\mathcal{R}, \mathcal{G}')$;
    If $\mathcal{G}' = \mathcal{G}$, $\nu_J = \nu$ and $J < J_{max} - 1$, set $J = J + 1$ and go to (5).
    Else
      • Set $\nu_{J+1} := \max\{\nu, \nu_J/2\}$;
      • Replace $J$ by $J + 1$;
      • Compute initial guess for next iteration:
            if $(j,k) \in \mathcal{G} \cap \mathcal{G}'$ then $u_{j,k} = u'_{j,k}$
            else if $(j,k) \in \mathcal{G}' \backslash \mathcal{G}$ then $u_{j,k} = \sum_{l \in \mathcal{I}^-_{j-1}} m^{j-1,0}_{k,l} u'_{j-1,l}$
            else if $(j,k) \in \mathcal{G} \backslash \mathcal{G}'$ then $u_{j,k} = \sum_{l \in \mathcal{I}^-_{j+1}} g^{j,0}_{k,l} u'_{j+1,l}$.
      • Replace $\mathcal{G}$ by $\mathcal{G}'$;
      • Determine the sets $\mathcal{I}^-_j$, $\mathcal{I}^+_j$, $j = j_0, \dots, J$ and set $\Phi^\circ_j := \{\phi_{l,k} : k \in$
        $\mathcal{I}^+_j, k \in \mathcal{I}^-_l, l = j_0, \dots, j-1\}$;
      • Set $\mathcal{S} = \{\mathrm{span}\,\Phi_{j_0}, \mathrm{span}\,\Phi^\circ_{j_0+1}, \dots, \mathrm{span}\,\Phi^\circ_J\}$;
      • Compute the new stiffness matrix $\mathbf{A}_J := a_{\nu_J}(\Phi_J, \Phi_J)$ and right-hand
        side $\boldsymbol{f}_J = (\Phi_J, f)$;
      • Go to (2).
```

One should note that the approximate solutions u_J are given in the form (40). Hence the stiffness matrices are always computed in the sparsest format. A multiscale stiffness matrix is never assembled. Only one step of the multiscale transformation is used to produce the block matrices. Otherwise it is used to produce the representation (42) for analysis purposes.

We close this section with some comments on the coarsening step in step (5). This step is necessary because the variation of the viscosity ν_j during the refinement process may cause layers to *migrate*. This is illustrated by the following simple univariate example where $\nu = 2^{-11}$, $f = 1$, $\gamma = 0$, and $\beta = \beta(x)$ is the ninth-degree truncated Taylor expansion of $-1-\sin(4\pi x)$. With the above parameters the algorithm produces 25 refinement steps while $J = 17$ is the highest level of local refinements. As shown by

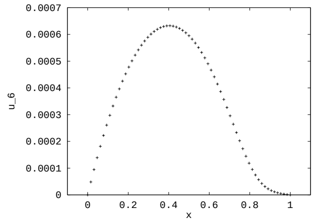


FIG. 5. $\nu_J = 2^{18}\nu,\ J = 6$.

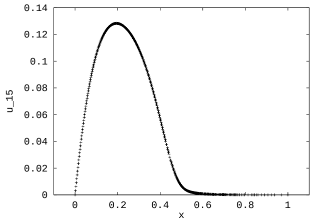


FIG. 6. $\nu_J = 2^{9}\nu,\ J = 15$.

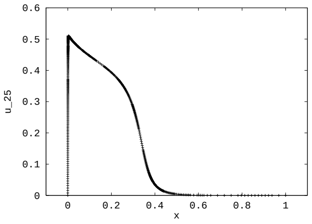


FIG. 7. $\nu_J = \nu,\ J = 25$.

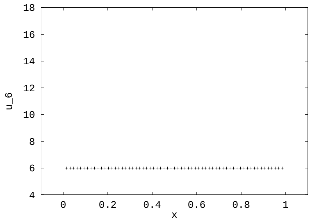


FIG. 8. $\nu_J = 2^{18}\nu,\ J = 6$.

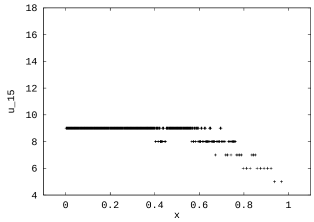


FIG. 9. $\nu_J = 2^{9}\nu,\ J = 15$.

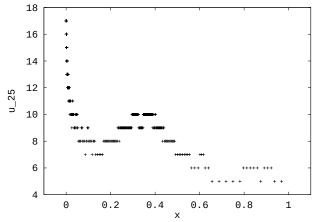


FIG. 10. $\nu_J = \nu,\ J = 25$.

Figures 5–10 the peak of intermediate approximate solutions migrates to the left when proceeding to higher levels and lowering viscosity. Without coarsening one would end up with unnecessary refinements throughout most of the domain.

Figures 5, 6, and 7 display approximate solutions for various current highest levels J and corresponding viscosity ν_J. The corresponding distributions of nodes are depicted in Figures 8, 9, and 10 to demonstrate how they evolve in the course of nested iteration.

5. Numerical results.

Algorithm **FMG** will be tested now on several examples posing different potential obstructions. Our main interest concerns (i) the reliability of the algorithm, i.e., all physical relevant effects should be appropriately resolved, (ii) the performance of the computation, e.g., in comparison with quasi-uniform discretizations, and (iii) the accuracy of the solution in terms of the involved degrees of freedom.

The test cases. We consider the following examples: (A) a constant direction of convection which results in a typical boundary layer, (B) variable convection causing a "near singularity" at a point in the interior, and (C) the interaction of convection and reaction. The corresponding convection and reaction coefficients are given by

(A) constant convection:
$\beta_1(x,y) = 1,\ \beta_2(x,y) = 1,\ \gamma = 0;$
(B) variable convection:
$\beta_1(x,y) = \frac{1}{2} - x,\ \beta_2(x,y) = \frac{1}{2} - y,\ \gamma = 0;$
(C) interaction of convection and reaction:
$\beta_1(x,y) = 1 + x^2,\ \beta_2(x,y) = x\,y,\ \gamma(x,y) = 10\,(\frac{1}{2} - x);$

In all cases we have employed finite element spaces spanned by standard bilinear shape functions on the domain $\Omega = (0,1)^2$. The coarsest level is always $j_0 = 2$, and $f = 1$ is always chosen as the right-hand side. We have deliberately dispensed with employing predetermined solutions because we wanted the singular behavior of the solution to truly reflect the particular choice of parameters in the variational problem.

The highest level of (local) refinement is always bounded by some integer J_{max}. Depending on the accuracy requirements and the coefficients in (1) this will sometimes be attained and sometimes a lower level $J = J_c$ turns out to suffice. All remaining parameters that have to be specified for the different computations are listed in Table 1. In particular, the threshold parameters $\tau_j = \kappa \, \|u_J\|_\infty \, 2^{j-J}$ and $\tau_j' = \kappa' \, \|u_J\|_\infty \, 2^{j-J}$ are characterized by the parameters κ and κ', respectively.

Table 1
Parameters for the computations.

Case	ν	κ	κ'	μ	J
A1	2^{-13}	2×10^{-6}	1×10^{-6}	5	14
A2	2^{-7}	1×10^{-5}	1×10^{-6}	3	9
B1	2^{-15}	5×10^{-5}	2.5×10^{-5}	5	14
B2	2^{-8}	2×10^{-4}	2×10^{-5}	3	9
C	2^{-7}	5×10^{-5}	5×10^{-5}	2	8

All computations have been carried out on a PC with a Pentium III processor (600 MHz).

Reliability. Some figures illustrating the numerical results for Cases A1, B1, and C follow. In Figures 11–16, we display the graph of the approximate solution and the corresponding adaptive mesh in order to see whether the automatic refinement actually captures the essential features of the solution and avoids unnecessary refinements.

Case A1. The refinement pattern in Figure 12 looks very appropriate—medium refinement around the diagonal ridge, strong local refinements at the corners and along the boundary layers. Of course, anisotropic refinements would be even better, but the present scheme does not require any a priori knowledge of the structure of the solution, and whenever alignment with meshlines is violated, isotropic refinements would evolve anyway.

Case B1. In Figure 14 we see a perfect symmetry and the refinement pattern accurately reflects the structure of the solution; see Figure 13.

Case C. Overall the test confirms that the algorithm deals with convection and reaction in essentially the same way. Although the reduction of unknowns is not as strong as before, one obtains a stable meaningful solution; see Figures 15 and 16.

In all cases we notice that the adaptation process works reliable in the sense that higher resolution levels occur only near steep gradients, whereas for moderate variation of the solution a much coarser resolution is recognized to suffice.

In particular, we emphasize that in Cases A1 and B1 we have chosen a fairly small diffusion without any a priori stabilization in the Galerkin discretization in order to see how this affects the various algorithmic ingredients. Progressing to extremely fine local resolutions turns out to put more and more burden on the approximate decoupling routine **DEC** (**A**), and a proper choice of the involved control parameters becomes essential when ν is of the order of 2^{-13}. So far these parameters are still set based on experience. A robust automatic adaptation is still subject to further research.

Performance. In order to demonstrate the benefits of the adaptive algorithm we compare the results with those for uniform meshes. Here we consider only the Cases A2 and B2, since the number of refinement levels has to be moderate due to the high memory requirements of the full grid.

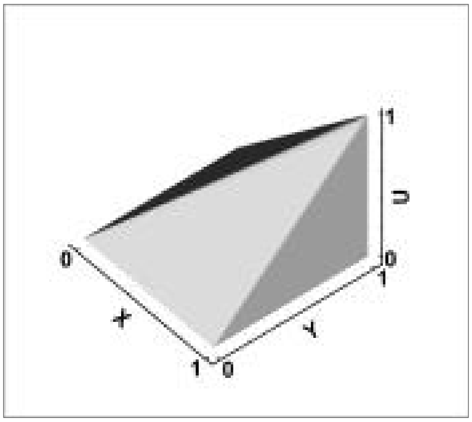


Fig. 11. *Case* A1.

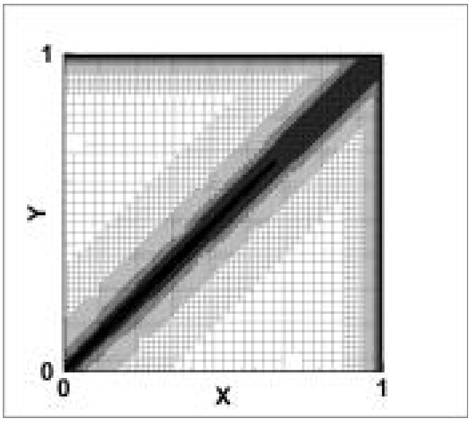


Fig. 12. *Case* A1.

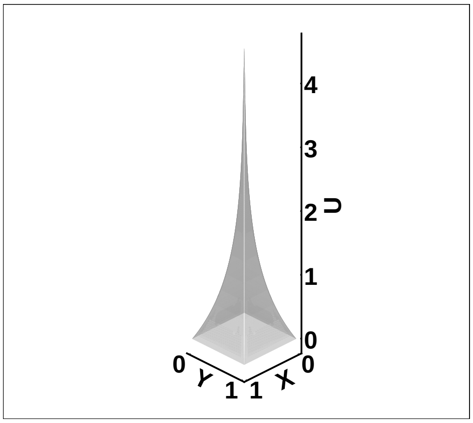


Fig. 13. *Case* B1.

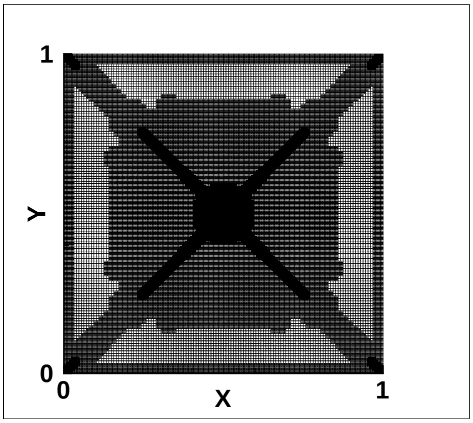


Fig. 14. *Case* B1.

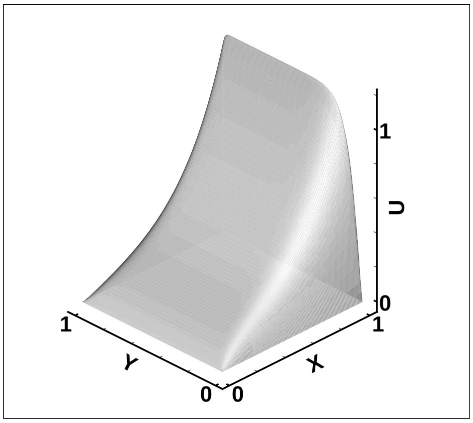


Fig. 15. *Case* C.

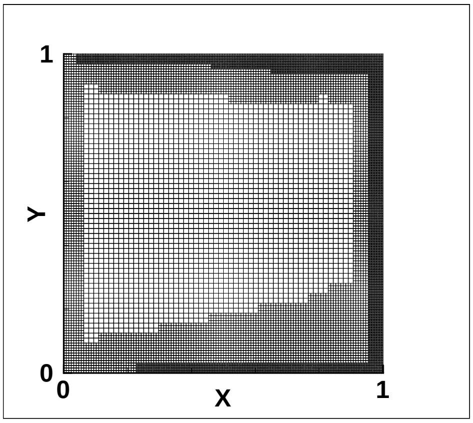


Fig. 16. *Case* C.

The results of these computations are collected in the Tables 2 and 3. The parameter j counts the number of refinements during nested iteration while j_c and j_f represent the *coarsest*, respectively, *finest* resolution that occurs locally at that stage. $\#\Phi_j^{adapt}$ stands for the actual number of degrees of freedom at the jth stage compared with the dimension $\#\Phi_j$ of the corresponding fully refined trial space. ν_j is the current viscosity. Employing highly nonuniform grids, we find that the meaning of grid-Péclet numbers is less clear. Of course, one can define them locally. P_f will stand for the local grid-Péclet number at the currently highest discretization level, while P_c refers to the locally coarsest level at stage j. The numerical results will show that their interpretation is no longer straightforward, particularly since variable convection will be considered. For instance, it will be seen that the spaces are *not* always refined at locations where the local grid-Péclet numbers are large.

Table 2
Case A2.

j	j_f	j_c	$\#\Phi_j^{adapt}$	$\#\Phi_j$	ν_j	P_f	P_c
3	3	3	49	49	2^{-3}	0.50	0.5
4	4	4	225	225	2^{-4}	0.50	0.5
5	5	5	961	961	2^{-5}	0.50	0.5
6	6	5	3829	3969	2^{-6}	0.50	1.0
7	7	5	7585	16129	2^{-7}	0.50	2.0
8	8	5	15059	65025	2^{-7}	0.25	2.0
9	9	5	29448	261121	2^{-7}	0.125	2.0

Table 3
Case B2.

j	j_f	j_c	$\#\Phi_j^{adapt}$	$\#\Phi_j$	ν_j	P_f	P_c
3	3	3	49	49	2^{-5}	0.708	0.708
4	4	4	225	225	2^{-5}	0.427	0.427
5	5	5	961	961	2^{-6}	0.464	0.464
6	6	5	2717	3969	2^{-7}	0.482	0.964
7	6	6	3969	16129	2^{-8}	0.482	0.482
8	7	6	4705	65025	2^{-8}	0.245	0.490
9	7	6	5045	261121	2^{-8}	0.245	0.490

According to Table 2 the number of unknowns is reduced by almost a factor 1/9. The Galerkin discretization is stable although $P_c = 2$ is encountered. While the number of degrees of freedom has been reduced by a factor of 8.9, the total CPU time has reduced by a factor of 4.6 in the comparison to the uniform grid.

Table 3 confirms that the savings in terms of grid compression are better than in Case A2 (a factor 1/52). Note that in this case also P_c remains moderate. The reduction rates of degrees of freedom and CPU time are 51.8, respectively, 27.4.

We like to emphasize that in Cases A1 and B1 the compression rates are even much higher. In Case A1 we have $\#\Phi_{14}^{adapt} = 461943$ and $\#\Phi_{14} = 268,402,689$ and for Case B1 $\#\Phi_{14}^{adapt} = 65173$ and $\#\Phi_{14} = 268,402,689$, respectively.

Accuracy. To assess the accuracy of an approximate solution $u_{\mathcal{G}}$ with coefficient array $\boldsymbol{u} =: \boldsymbol{u}_{\mathcal{G}}$ on an adaptive mesh $\mathcal{G}$, we will compare it with a highly accurate fixed reference solution u_L that will play the role of the exact solution. The difference $\|u_{\mathcal{G}} - u_L\|_X$ for $X \in \{L_2, H^1\}$ can be evaluated with the aid of the norm equivalences

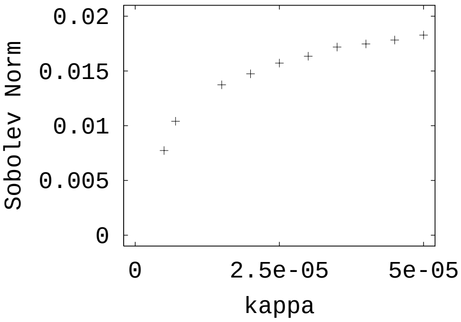


FIG. 17. *Case* A2: H^1*-norm.*

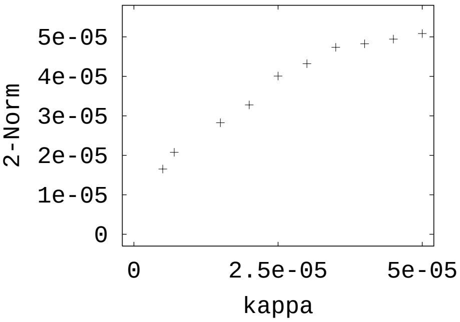


FIG. 18. *Case* A2: L_2*-norm.*

(19), which requires only expanding $u_{\mathcal{G}}$ and u_L in the stable multiscale basis described earlier.

To interpret the results correctly recall that on uniform grids with meshsize h piecewise linear approximands can produce at best errors in $\|\cdot\|_{H^s(\Omega)}$ of order h^{2-s}, provided that the approximated function belongs at least to $H^2(\Omega)$. In terms of degrees of freedom N the error can therefore decay at best with order $N^{-(2-s)/d}$ in d spatial dimensions. This remains the best possible order also when using adaptive approximations. The gain of adaptive schemes is that this best possible order can be realized under weaker regularity assumptions. The relevant (weaker) regularity measure is known to be based on a certain scale of Besov spaces; see, e.g., [11].

Therefore we wish to determine how $\|u_{\mathcal{G}} - u_L\|_X$ relates to the number N of adaptively generated unknowns. Since for small viscosity H^1 is not the right energy space we are primarily interested in the L_2-error, i.e., $X = L_2$. Nevertheless, we monitor also the H^1-error, which is expected to be much larger due to the large gradients in layer regions.

Recall that the threshold parameter κ is to control the accuracy of the approximate solutions. It is therefore also of interest to study the dependence of $\|u_{\mathcal{G}} - u_L\|_X$ on κ, again mainly for $X = L_2$.

Figures 17 and 18 display the dependence of the H^1-error, respectively, the L_2-error on the parameter κ. We see that in Case A2 the L_2-error stays essentially proportional to κ (in the range between $\kappa = 5 \cdot 10^{-6}$ and $5 \cdot 10^{-5}$) up to a factor of 2 so that κ is a good accuracy measure. As expected the H^1-error is much larger.

Figures 19 and 20 show the dependence of the H^1- and L_2-errors on the number N of degrees of freedom. In agreement with the dependence on κ the optimal rates $N^{-1/2}$, respectively, N^{-1} seem to be matched. Figures 21 and 22 confirm the same behavior for the L_2-norm in Case B2.

5.1. Concluding remarks. The above examples show that the algorithm **FMG** exhibits essentially the same performance in a variety of test cases with different characteristic features. The adaptive refinements reduce the computational complexity significantly in all cases and seem to stabilize the Galerkin approximations in a reliable way. In none of the examples have we used an a priori stabilization technique. The adaptive meshes appear to reflect the structure of the solutions in all cases very well. Nevertheless, our tests also show that a proper choice of the threshold parameters is important. The selection of these parameters is at this point still partly based on heuristics and experience. Likewise the role of appropriate approximate decoupling becomes more and more crucial when the convection term grows. The above variant

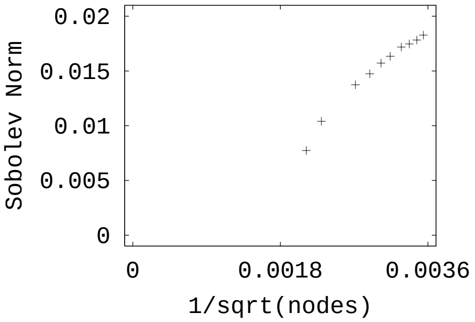


FIG. 19. *Case* A2: H^1*-norm.*

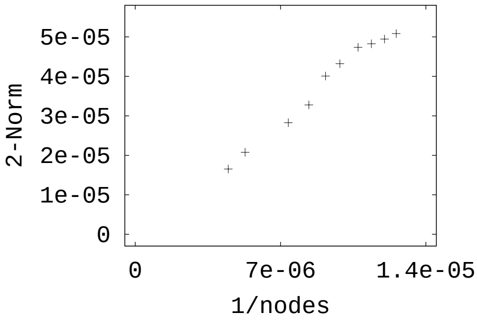


FIG. 20. *Case* A2: L_2*-norm.*

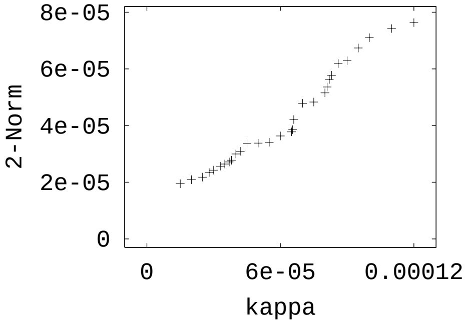


FIG. 21. *Case* B2: L_2*-norm.*

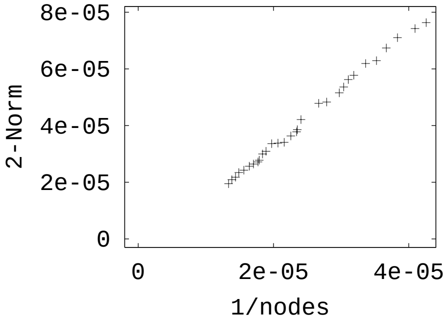


FIG. 22. *Case* B2: L_2*-norm.*

of SPAI techniques has so far been the only successful scheme that works in the viscosity range under consideration. However, further efforts are needed to increase its robustness.

So far we have used for simplicity a very crude way of modifying the original problem in **FMG** for a proper low-level treatment. Alternatively one could use more sophisticated schemes like *streamline diffusion* or any other stabilization scheme. Since the stabilization will gradually be removed when moving up to higher levels the precise tuning of parameters is irrelevant. Likewise, one could employ a weak a priori stabilization corresponding to the prescribed highest level J_{max} if it is not important to resolve all features of the solution with high accuracy. In fact, it should be clear that without *any* a priori stabilization it would be much harder to progress to significantly higher Reynolds numbers than with conventional stabilized versions. In this sense the present investigation is to explore the potential of such aspects.

REFERENCES

[1] O. Axelsson and V. Barker, *Finite element solution of boundary value problems*, Computer Science and Applied Mathematics, Academic Press, Orlando, FL, 1984.

[2] O. Axelsson and M. Nikolova, *Adaptive refinement for convection-diffusion problems based on a defect-correction technique and finite difference method*, Computing, 58 (1997), pp. 1–30.

[3] O. Axelsson and M. Nikolova, *Avoiding slave points in adaptive refinement procedure for convection-diffusion problems in* 2*D*, Computing, 61 (1998), pp. 331–357.

[4] O. Axelsson and P. Vassilevski, *Algebraic multilevel preconditioning methods* I, Numer. Math., 56 (1989), pp. 157–177.

[5] R. Bank, T. Dupont, and H. Yserentant, *The hierarchical basis multigrid method*, Numer. Math., 52 (1988), pp. 427–458.

[6] R. Bank and S. Gutsch, *The generalized hierarchical basis two-level method for the convection-diffusion equation on a regular grid*, in Multigrid Methods V, Proceedings of the Fifth European Multigrid Conference, W. Hackbusch et al., eds., Lect. Notes Comput. Sci. Eng. 3, Springer-Verlag, Berlin, 1998, pp. 1–20.

[7] R. Bank and S. Gutsch, *Hierarchical basis for the convection-diffusion equation on unstructured meshes*, in Proceedings of the Ninth International Symposium on Domain Decomposition Methods for Partial Differential Equations, P. Bjørstad, M. Espedal, and D. Keyes, eds., J. Wiley and Sons, New York, 1998, pp. 251–265.

[8] J. Bey and G. Wittum, *Downwind numbering: Robust multigrid for convection-diffusion problems*, Appl. Numer. Math., 23 (1997), pp. 177–192.

[9] J. Carnicer, W. Dahmen, and J. Peña, *Local decomposition of refinable spaces and wavelets*, Appl. Comput. Harmon. Anal., 3 (1996), pp. 127–153.

[10] I. Christie, D. Griffiths, A. Mitchell, and O. Zienkiewicz, *Finite element method for second order differential equation with significant first derivatives*, Internat. J. Numer. Methods Engrg., 10 (1976), pp. 1389–1396.

[11] A. Cohen, W. Dahmen, and R. DeVore, *Adaptive Wavelet Methods for Elliptic Operator Equations—Convergence Rates*, IGPM–Report 165, RWTH Aachen, Germany, 1998.

[12] W. Dahmen, *Wavelet and multiscale methods for operator equations*, Acta Numer., 6 (1997), pp. 55–228.

[13] W. Dahmen, S. Müller, and T. Schlinkmann, *Multigrid and multiscale decompositions*, in Large-Scale Scientific Computations of Engineering and Environmental Problems, M. Griebel, O. Iliev, S. Margenov, and P. Vassilevski, eds., Vieweg, Braunschweig, Germany, 1998, pp. 18–41.

[14] W. Dahmen, S. Müller, and T. Schlinkmann, *On a Robust Adaptive Multigrid Solver for Convection-Dominated Problems*, IGPM–Report 171, RWTH Aachen, Germany, 1999.

[15] W. Dahmen, R. Schneider, and Y. Xu, *Nonlinear Functionals of Wavelet Expansions—Adaptive Reconstruction and Fast Evaluation*, IGPM–Report 160, RWTH Aachen, Germany, 1998.

[16] P. DeZeeuw, *Matrix-dependent prolongations and restrictions in a blackbox multigrid solver*, J. Comput. Appl. Math., 33 (1990), pp. 1–27.

[17] M. J. Grote and T. Huckle, *Parallel preconditioning with sparse approximate inverses*, SIAM J. Sci. Comput., 18 (1997), pp. 838–853.

[18] W. Hackbusch, *On the feedback vertex set problem for a planar graph*, Computing, 58 (1997), pp. 129–155.

[19] W. Hackbusch and T. Probst, *Downwind Gauss–Seidel smoothing for convection dominated problems*, Numer. Linear Algebra Appl., 4 (1997), pp. 85–102.

[20] J. Heinrich, P. Huyakorn, O. Zienkiewicz, and A. Mitchell, *An upwind finite element scheme for two-dimensional convective transport equation*, Internat. J. Numer. Methods Engrg., 11 (1977), pp. 131–143.

[21] T. Hughes and A. Brooks, *A multidimensional upwind scheme with no crosswind diffusion*, in Finite Element Methods for Convection Dominated Flows, AMD 34, Amer. Soc. Mech. Engrs., New York, 1979.

[22] C. Johnson, *Numerical Solution of Partial Differential Equations by the Finite Element Method*, Cambridge University Press, Cambridge, UK, 1987.

[23] R. Lorentz and P. Oswald, *Multilevel finite element Riesz bases in Sobolev spaces*, in DD9 Proceedings, P. Bjørstad, M. Espedal, and D. Keyes, eds., 1997.

[24] A. Reusken, *Multigrid with matrix-dependent transfer operators for a singular perturbation problem*, Computing, 50 (1993), pp. 199–211.

[25] A. Reusken, *Fourier analysis of a robust multigrid method for convection-diffusion equations*, Numer. Math., 71 (1995), pp. 365–397.

[26] A. Reusken, *On a robust multigrid solver*, Computing, 56 (1996), pp. 303–322.

[27] H. Roos, *Layer-adapted grids for singular perturbation problems*, ZAMM Z. Angew. Math. Mech., 78 (1998), pp. 291–309.

[28] H. Roos, M. Stynes, and L. Tobiska, *Numerical Methods for Singularly Perturbed Differential Equations*, Springer Ser. in Comput. Math. 24, Springer-Verlag, Berlin, 1996.

[29] J. W. Ruge and K. Stüben, *Algebraic multigrid*, in Multigrid Methods, Frontiers Appl. Math. 3, S. McCormick, ed., SIAM, Philadelphia, 1987, pp. 73–130.

[30] T. Schlinkmann, *Ein adaptives Mehrgitterverfahren für konvektionsdominante Konvektions–Diffusions–Probleme*, Ph.D. thesis, RWTH Aachen, Shaker Verlag, Aachen, Germany, 2001.

[31] P. Vassilevski and J. Wang, *Stabilizing the hierarchical basis by approximate wavelets* I: *Theory*, Numer. Linear Algebra Appl., 4 (1997), pp. 103–126.

[32] P. Wesseling, *A robust and efficient multigrid method*, in Multigrid Methods, W. Hackbusch and U. Trottenberg, eds., Lecture Notes in Math. 960, Springer-Verlag, New York, 1982, pp. 614–630.

[33] G. Wittum, *On the robustness of ILU smoothing*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 699–717.

[34] J. Witzel, *Numerical Solution of Linear Systems arising from a New Petrov–Galerkin Discretization of Convection–Diffusion Problems*, Ph.D. thesis, TH Darmstadt, Shaker Verlag, Aachen, Germany, 1997.

[35] H. Yserentant, *On the multi-level splitting of finite element spaces*, Numer. Math., 49 (1986), pp. 379–412.

SIAM J. SCI. COMPUT.
Vol. 23, No. 3, pp. 805–823

POINT VALUE MULTISCALE ALGORITHMS FOR 2D COMPRESSIBLE FLOWS*

GUILLAUME CHIAVASSA† AND ROSA DONAT†

Abstract. The numerical simulation of physical problems modeled by systems of conservation laws is difficult due to the presence of discontinuities in the solution. High-order shock capturing schemes combine sharp numerical profiles at discontinuities with a highly accurate approximation in smooth regions, but usually their computational cost is quite large.

Following the idea of A. Harten [*Comm. Pure Appl. Math.*, 48 (1995), pp. 1305–1342] and Bihari and Harten [*SIAM J. Sci. Comput.*, 18 (1997), pp. 315–354], we present in this paper a method to reduce the execution time of such simulations. It is based on a point value multiresolution transform that is used to detect regions with singularities. In these regions, an expensive high-resolution shock capturing scheme is applied to compute the numerical flux at cell interfaces. In smooth regions a cheap polynomial interpolation is used to deduce the value of the numerical divergence from values previously obtained on lower resolution scales.

This method is applied to solve the two-dimensional compressible Euler equations for two classical configurations. The results are analyzed in terms of quality and efficiency.

Key words. conservation laws, multiresolution, shock capturing schemes

AMS subject classifications. 65M06, 65D05, 35L65

PII. S1064827599363988

1. Introduction.

The computation of solutions to hyperbolic systems of conservation laws has been a very active field of research for the last 20 to 30 years and, as a result, there are now a variety of methods that are able to compute accurate numerical approximations to the physically relevant solution. The latest addition to the pool of numerical methods for hyperbolic conservation laws are the modern high-resolution shock capturing (HRSC) schemes. These schemes succeed in computing highly accurate numerical solutions, typically second- or third-order in smooth regions, while maintaining sharp, oscillation-free numerical profiles at discontinuities.

State-of-the-art shock capturing schemes usually perform a "delicate art-craft" on the computation of the numerical flux functions. A typical computation involves at least one eigenvalue-eigenvector decomposition of the Jacobian matrix of the system, as well as the approximation of the values of the numerical solution at both sides of each cell interface, obtained via some appropriately chosen approximating functions. The numerical result is very often spectacular in terms of resolution power, but the computational effort also tends to be quite spectacular.

Without doubt, the computational speed of the latest personal computers and workstations has made it possible for an increasing number of researchers to become interested in HRSC methods and, as a result, HRSC methods are now being tested in a variety of physical scenarios that involve hyperbolic systems of conservation laws (see, e.g., [7, 10, 19] and references therein).

When the underlying grid is uniform, the implementation of most of these shock capturing schemes is quite straightforward and numerical simulations on uniform grids

*Received by the editors November 16, 1999; accepted for publication (in revised form) October 30, 2000; published electronically August 15, 2001. This work was supported in part by TMR Research Network contract FMRX-CT98-0184 and by Spanish DGICYT grant PB97-1402.

http://www.siam.org/journals/sisc/23-3/36398.html

†Departamento de Matematica Aplicada, Universidad de Valencia, 46100 Burjassot (Valencia), Spain (guillaume.chiavassa@uv.es, donat@uv.es).

are routinely used to investigate the behavior of the different HRSC schemes in use, and also their limitations. It is known that some HRSC schemes can produce an anomalous behavior in certain situations; a catalog of numerical pathologies encountered in gas dynamics simulations can be found in [20], where it is observed that some of these pathologies appear only when very fine meshes are used.

When using very fine uniform grids, in which the basic code structure of an HRSC scheme is relatively simple, we find that the computational time becomes the main drawback in the numerical simulation. For some HRSC schemes, fine mesh simulations in two dimensions are out of reach simply because they cost too much. The numerical flux evaluations are too expensive, and the computational time is measured by days or months on a personal computer. As an example we notice that a typical computation of a two-dimensional (2D) jet configuration in [19] is 10 to 50 days on an HP710 or 1 to 5 days on an Origin 2000 with 64 processors.

It is well known, however, that the heavy-duty flux computations are needed only because nonsmooth structures may develop spontaneously in the solution of a hyperbolic system of conservation laws and evolve in time, and this basic observation has lead researchers to the development of a number of techniques that aim at reducing the computational effort associated to these simulations. Among these, shock tracking and adaptive mesh refinement (AMR) techniques (often combined with one another) are very effective at obtaining high-resolution numerical approximations, but the computational effort is transferred to the programming and the data structure of the code.

Starting with the pioneering work of Harten [14], a different multilevel strategy aiming to reduce the computational effort associated to high-cost HRSC methods entered the scene. The key observation is that the information contained in a multiscale decomposition of the numerical approximation can be used to determine its local regularity (smoothness). At discontinuities or steep gradients, it is imperative to use a numerical flux function that models correctly the physics of the problem, but in smoothness regions the costly exact value of an HRSC numerical flux can be replaced by an equally accurate approximation obtained by much less expensive means. The multiscale decomposition of the numerical solution can then be used as a tool to decide *in which regions* a sophisticated evaluation of the numerical flux function is truly needed. In smoothness regions, Harten proposes [14] to evaluate the numerical flux function of the HRSC scheme only on a coarse grid and then use these values to compute the fluxes on the finest grid using an inexpensive polynomial interpolation process in a multilevel fashion.

Harten's approach can be viewed, in a way, as an AMR procedure, in which grids of different resolutions are considered in the numerical simulation, but in reality it is far from being an AMR technique. The different grids are used *only* to analyze the smoothness of the numerical solution. The numerical values on the highest-resolution grid need to be *always* available, because the computation of the numerical fluxes with the HRSC scheme, when needed, use directly the finest-grid values. This is clearly a disadvantage with respect to the memory savings that an AMR technique can offer in certain situations. On the other hand, using the values of the numerical solution in the direct computation has some nice features. First, it avoids the use of complicated data structures, which is very useful when one is trying to incorporate the algorithm into an existing code. Second, the availability of the numerical solution on the finest grid guarantees that the "delicate art-craft" involved in the direct evaluation of the numerical fluxes (via a sophisticated HRSC scheme) is performed adequately.

When memory requirements do not impose a severe restriction (as it often happens in many 2D, as well as in some three-dimensional (3D), computations), the techniques proposed in [14, 5, 22, 1] and in this paper can help to reduce the large running times associated to numerical simulations with HRSC schemes. We view Harten's approach as an acceleration tool, which can be incorporated in a straightforward manner into an existing code.

The novelty of our approach with respect to the multilevel strategies described in [14, 5, 22, 1] lies in the multiresolution transform used to analyze the smoothness of the numerical solution. We use the interpolatory framework, while in the references mentioned above the cell-average framework is used. In addition, our implementation incorporates several features that improve the efficiency of the algorithm (see section 3.3), while maintaining the quality of the numerical approximation.

The rest of the paper is organized as follows: In section 2, we briefly describe the essential features of the HRSC schemes we shall employ in our simulations. In section 3 we describe the interpolatory framework for multiresolution and its role in our multilevel strategy, as well as some implementation details. Section 4 examines the accumulation of error in the multilevel simulation. In section 5 we perform a series of numerical experiments and analyze the results in terms of *quality*, i.e., closeness to the reference simulation, and *efficiency*, i.e., time savings of the multilevel simulations with respect to the reference simulation. Finally, some conclusions are drawn in section 6.

2. Shock capturing schemes for 2D systems of conservation laws. Let us consider a 2D system of hyperbolic conservation laws:

$$\partial_t \vec{U} + \vec{f}(\vec{U})_x + \vec{g}(\vec{U})_y = \vec{0}, \tag{2.1}$$

where $\vec{U}$ is the vector of conserved quantities. We shall consider discretizations of this system on a Cartesian grid $\mathcal{G}^0 = \{(x_i = i\delta x, y_j = j\delta y), \quad i = 0, \dots, Nx \ j = 0, \dots, Ny\}$ that follow a semidiscrete formulation,

$$\frac{d\vec{U}_{ij}}{dt} + \mathcal{D}(\vec{U})_{ij} = 0, \tag{2.2}$$

with the numerical divergence $\mathcal{D}(\vec{U})_{ij}$ in *conservation form*, i.e.,

$$\mathcal{D}(\vec{U})_{ij} = \frac{\vec{F}_{i+1/2,j} - \vec{F}_{i-1/2,j}}{\delta x} + \frac{\vec{G}_{i,j+1/2} - \vec{G}_{i,j-1/2}}{\delta y}. \tag{2.3}$$

One typically has $\vec{F}_{i+1/2,j} = \vec{F}(\vec{U}_{i-k,j}, \dots, \vec{U}_{i+m,j})$, $\vec{G}_{i,j+1/2} = \vec{G}(\vec{U}_{i,j-k}, \dots, \vec{U}_{i,j+m})$, where $\vec{F}(\vec{w}_1, \dots, \vec{w}_{k+m})$ and $\vec{G}(\vec{w}_1, \dots, \vec{w}_{k+m})$ are consistent numerical flux functions, which are the trademark of the scheme.

In this paper we shall use two numerical flux formulae, which are significantly different in terms of computational effort:

- The essentially nonoscillatory (ENO) method of order 3 (ENO-3 henceforth) from [21], which uses the nonlinear piecewise parabolic ENO reconstruction procedure to achieve high accuracy in space.
- Marquina's scheme from [8] together with the piecewise hyperbolic method (PHM) [18] to obtain high accuracy in space (M-PHM henceforth).

In both cases, the reconstruction procedure (piecewise parabolic ENO or piecewise hyperbolic) is applied directly on the fluxes, as specified by Shu and Osher in [21].

The ENO-3 scheme uses Roe's linearization and involves one Jacobian evaluation per cell interface, while M-PHM uses a flux-splitting technique in the flux computation that requires two Jacobian evaluations per cell interface. Although M-PHM is more expensive than ENO-3, it has been shown in [8, 12, 9] that it is a pretty robust scheme that, in addition, avoids (or reduces) certain numerical pathologies associated to the Roe solver.

In both cases, a third-order fully discrete scheme is obtained by applying a TVD Runge–Kutta method for the time evolution as proposed in [21].

3. The multilevel algorithm. As explained in the introduction, the goal of the multilevel method is to *decrease the cpu time* associated to the underlying scheme by reducing the number of expensive flux evaluations. To understand the basic mechanism by which this goal is achieved, let us consider, for the sake of simplicity, Euler's method applied to (2.2), i.e.,

$$\vec{U}_{ij}^{n+1} = \vec{U}_{ij}^{n} - \delta t\, \mathcal{D}(\vec{U})_{ij}^{n}. \tag{3.1}$$

If *both* U^n and U^{n+1} are smooth around (x_i, y_j) at time t^n, then (3.1) implies that the numerical divergence is also smooth at that location; thus we can, in principle, avoid using the numerical flux functions of the HRSC scheme in its computation. On the other hand, if a discontinuity appears during the time evolution (or when a steep gradient makes it imminent), the Riemann solver of the HRSC scheme has to be called necessarily to compute the numerical divergence if the high-resolution properties of the underlying scheme are to be maintained.

Consequently, the most important steps in the multilevel algorithm concern the smoothness analysis of U^n *and* U^{n+1} (observe that the latter is unknown at time n) and how this information is used in the computation of $\mathcal{D}(\vec{U})$.

3.1. Interpolatory multiresolution. Finite volume schemes for (2.1) produce numerical values that can be naturally interpreted as approximations to the mean values of the solution in each computational cell (the *cell averages*). Because of this, all applications of Harten's idea known to us [14, 5, 22, 6, 1, 2, 17] have invariably used the cell-average multiresolution framework (see [14] for definitions and details) to analyze the smoothness of the numerical approximation.

In Shu and Osher's framework, the numerical values can be interpreted as approximations to the point values of the solution. In a point value framework for multiresolution, the numerical data to be analyzed are interpreted as the values of a function on an underlying grid; consequently, in our multilevel strategy the *point value multiresolution framework* is used to analyze the smoothness properties of the numerical approximation.

Multiscale decompositions within the point value framework were initially introduced by Harten [13] (and also independently developed by Sweldens [23]) and have been extensively analyzed in a series of papers [15, 3]. Here we present only a brief summary to clarify the notation in the remainder of the paper.

One first defines a set of nested grids $\{\mathcal{G}^l,\ l = 1, \dots, L\}$ by

$$(x_i, y_j) \ \in\ \mathcal{G}^l \Longleftrightarrow (x_{2^l i}, y_{2^l j}) \ \in\ \mathcal{G}^0. \tag{3.2}$$

The values of a function v on $\mathcal{G}^0$ (which is considered the finest resolution level), $(v_{ij}^0)_{i,j}$, are the input data. Due to the embedding of the grids, the representation of the function on the coarser grid $\mathcal{G}^l$, its point values on $\mathcal{G}^l$, is

$$v_{ij}^l = v_{2^l i\ 2^l j}^0, \quad i = 0, \dots, Nx/2^l, \quad j = 0, \dots, Ny/2^l. \tag{3.3}$$

To recover the representation of v on $\mathcal{G}^{l-1}$ from the representation on $\mathcal{G}^l$ (the next coarser grid), the following procedure is used:

- A set of predicted values is first computed:

$$\begin{aligned} &\tilde{v}_{ij}^{l-1} = v_{i/2\ j/2}^l \quad \text{if } (x_i, y_j) \in\ \mathcal{G}^l, \\ &\tilde{v}_{ij}^{l-1} = I[(x_i, y_j); v^l] \quad \text{if } (x_i, y_j) \in \mathcal{G}^{l-1} \backslash\ \mathcal{G}^l, \end{aligned} \tag{3.4}$$

  where $I(.;.)$ denotes an rth-order polynomial interpolation.
- The difference between the exact values (3.3), v_{ij}^{l-1}, and $\tilde{v}_{ij}^{l-1}$ is then represented by the details, or wavelet coefficients:

$$d_{ij}^l = v_{ij}^{l-1} - \tilde{v}_{ij}^{l-1}, \quad (x_i, y_j) \in \mathcal{G}^{l-1}. \tag{3.5}$$

  Observe that $d_{2p,2q}^l = 0$ because of the interpolation property. Thus even-even detail coefficients are never computed (or stored).
- Relations (3.4) and (3.5) lead immediately to

$$\begin{aligned} &v_{ij}^{l-1} = v_{i/2\ j/2}^l \quad \text{if } (x_i, y_j) \in \mathcal{G}^l, \\ &v_{ij}^{l-1} = I[(x_i, y_j); v^l] + d_{ij}^l \quad \text{if } (x_i, y_j) \in \mathcal{G}^{l-1} \backslash\ \mathcal{G}^l. \end{aligned} \tag{3.6}$$

Applying this procedure from $l = 1$ to L gives an equivalence between the discrete set v^0 and its multiresolution representation: $Mv^0 = (v^L, d^L, \ldots, d^1)$.

Remark 3.1. In our numerical experiments we use a tensor-product interpolation procedure of order 4 ($r = 4$). The corresponding formulae come from standard 2D polynomial interpolation; explicit details can be found, for example, in [5, section 3].

The point value framework for multiresolution is probably the simplest one, because the detail coefficients are simply interpolation errors. When the grid is uniform and the interpolation technique is constructed using a tensor product approach, it is very simple to analyze the smoothness information contained in the interpolation errors, which can then be used directly as "regularity sensors" to localize the nonsmooth structures of the solution (compare with the derivation of the regularity sensors in [5] in the 2D cell-average framework for multiresolution).

3.2. The basic strategy. As observed in [5], the original idea of a multilevel computation of the numerical flux function (in one dimension) described by Harten in [14] cannot be used in a robust and general manner in two dimensions. The key point is then to observe that it is the *numerical divergence* the quantity that should be adapted to the multilevel computations. For the sake of simplicity, let us consider again the simplest ODE solver: Euler's method. When applying it to the semidiscrete formulation (2.2), we get

$$\vec{U}_{ij}^{n+1} - \vec{U}_{ij}^n = -\delta t\ \mathcal{D}(\vec{U})_{ij}^n, \tag{3.7}$$

and this relation shows that a multilevel computation of the numerical divergence must be carried out within the same framework as the sets $\vec{U}_{ij}^{n,n+1}$. The idea to use the numerical divergence instead of the flux for the multilevel computation was a key step in the development of multilevel strategies in multidimensions in [5, 1]. Once this fact is recognized, the choice of the particular framework used to analyze the smoothness information contained in the numerical approximation is not crucial. We propose to use the point value framework because of its simplicity.

As in [5], the computation of the numerical divergence $\mathcal{D}(\vec{U}^n)$ on the finest grid is carried out in a sequence of steps. First the numerical divergence is evaluated at all points on the coarsest grid $\mathcal{G}^L$ using the numerical flux function of the prescribed HRSC scheme. Then, for the finer grids, $\mathcal{D}$ is evaluated recursively, either by the same procedure *or* with a cheap interpolation procedure using the values obtained on the coarser grids. The choice depends on the regularity analysis of the approximate solution, made with the help of its multiresolution representation.

Thus, the main ingredients of the algorithm are the following:

- The *multiresolution transform* described in section 3.1 to obtain the wavelet coefficients of $\vec{U}^n$.
- A *thresholding algorithm* which associates to each wavelet coefficient a boolean flag, b^l_{ij}, whose value (0 or 1) will determine the choice of the procedure to evaluate $\mathcal{D}(U)$. The goal is to use this flag to mark out the nonsmooth regions of *both* $\vec{U}^n$ and $\vec{U}^{n+1}$. This is done as follows:

  For a given tolerance parameter ϵ, the tolerance at level l is defined as $\epsilon_l = \epsilon/2^l$. Starting from a zero value for all b^l_{ij}, one applies for each detail coefficient the following two tests:

  $$\begin{aligned}&\text{if } |d^l_{ij}| \geq \epsilon_l \implies b^l_{i-k\ j-m} = 1, \quad k,m = -2,\dots,2,\\ &\text{if } |d^l_{ij}| \geq 2^{r+1}\epsilon_l \quad \text{and} \quad l > 1 \implies b^{l-1}_{2i-k\ 2j-m} = 1, \quad k,m = -1,0,1.\end{aligned}$$

  The first test takes into account the propagation of information (recall that the propagation of "real" information is limited by the CFL condition). The second one aims at detecting shock formation. In a smooth region the local rate of decay of the detail coefficients is determined by the accuracy of the interpolation and the local regularity of the function. The second test measures whether the decay rate is that of a smooth function; if this is not the case, compression leading to shock formation might be taking place and the location is also flagged (see [5] for specific details on both tests).
- The *multilevel evaluation* of the numerical divergence.

  For all points $(x_i, y_j) \in \mathcal{G}^L$, $\mathcal{D}^L(\vec{U})_{ij}$ is computed with the prescribed HRSC scheme. Once the divergence is known on $\mathcal{G}^l$, its value on $\mathcal{G}^{l-1}$, $\mathcal{D}^{l-1}(\vec{U})$ is evaluated using the boolean flag:

  $$\text{If } b^l_{ij} = 1, \quad \text{compute } \mathcal{D}^{l-1}(\vec{U})_{ij} \text{ directly (with the HRSC method).}$$

  $$\text{If } b^l_{ij} = 0, \quad \mathcal{D}^{l-1}(\vec{U})_{ij} = I[(x_i, y_j); \mathcal{D}^l(\vec{U})].$$

Letting l go from L to 1 gives us the values of $\mathcal{D}(\vec{U}^n)$ on the finest grid $\mathcal{G}^0$.

Remark 3.2. Recall that $\mathcal{D}^{l-1}(\vec{U})_{ij} = \mathcal{D}^0(\vec{U})_{2^{l-1}i,2^{l-1}j}$, and thus the direct evaluation of $\mathcal{D}^{l-1}(\vec{U})_{ij}$ is performed by computing the numerical flux functions using the values of $\vec{U}$ on $\mathcal{G}^0$: $\vec{F}(\vec{U}_{2^{l-1}i-k,j}, \dots, \vec{U}_{2^{l-1}i+m,j})$ and $\vec{G}(\vec{U}_{i,2^{l-1}j-k}, \dots, \vec{U}_{i,2^{l-1}j+m})$. As a consequence, the finest grid, $\mathcal{G}^0$, is always needed and no memory savings is obtained in comparison to the direct method (without multiresolution).

3.3. Some implementation details. In the original work of Harten [14], the flag coefficients are obtained using a multiresolution transform for each component of the vector $\vec{U}$. The thresholding algorithm is then applied to the largest resulting wavelet coefficient, i.e., $\tilde{d}^l_{ij} = \max(|d_{ij}(\rho)|, |d_{ij}(m_x)|, |d_{ij}(m_y)|, |d_{ij}(E)|)$.

For the Euler equations of gas dynamics, the density retains all the possible nonsmooth structures of the flow (shocks, contact discontinuities, and corners of rarefaction waves); thus it seems appropriate to derive the flag only from the multiresolution representation of the density, a modification that has also been implemented by

Table 3.1

Cpu time in seconds for the overhead steps of the multilevel algorithm. First column for all the components of $\vec{U}$, second one only for the density.

	$\vec{U}$ multiresolution	ρ multiresolution
MR transform	0.09	0.024
Maximum	0.014	–
Thresholding	0.026	0.027
Total	0.13	0.051

Sjögreen [22]. In our experiments, no significant differences are noted in the quality of the numerical results obtained by computing the flag from the density only; thus in the numerical tests we report, the boolean flag is computed using only the multiscale information of the density. This option saves time in the overhead associated to the multiscale algorithm. In Table 3.1, we present the cpu time measured for both methods for an initial grid $\mathcal{G}^0$ of 512×128 points and 5 levels for the multiresolution transform.

We observe a reduction of the computational time by a factor of 2 in that case.

In [22], Sjögreen presents numerical simulations for 2D systems of conservation laws using a "dimension-by-dimension" cell-average multilevel algorithm and uniform meshes. This means that for the fluxes in the x-direction, $\vec{F}_{i+1/2,j}$, a one-dimensional (1D) multilevel algorithm is applied to each grid line $j = j_0$. Then, the same procedure is applied to each line $i = i_0$ to compute the y-fluxes. The major advantage of Sjögreen's implementation lies in its simplicity: only 1D procedures are used for the multiresolution transform, the thresholding algorithm, and the interpolation process.

We have also implemented Sjögreen's version in the point value context and have compared it with our algorithm, in which a fully bidimensional multiresolution transform is used. Qualitatively speaking, the results are very similar and the percentage of fluxes computed by the solver is the same in both cases. Nevertheless, Sjögreen's version turns out to be less efficient than the 2D one and, in our implementation, a factor of 1.6 is observed between the corresponding cpu times. The difference could be explained by the fact that each point of the domain is visited two times by the multiresolution transform and thresholding algorithm (for the x- and y-flux computations) and that this algorithm requires more memory access.

A Runge–Kutta ODE solver is applied to the semidiscrete scheme (2.2)–(2.3) to obtain a fully discrete scheme. In [5, 1, 22], a flag vector is computed at the beginning of each Runge–Kutta step between t^n and t^{n+1}, but it is possible to avoid this computation for the last step. The third-order TVD Runge–Kutta method of [21] is defined as follows:

$$\begin{aligned} \vec{U}^* &= \vec{U}^n - \delta t\ \mathcal{D}(\vec{U}^n), \\ \vec{U}^{**} &= (3\vec{U}^n + \vec{U}^* - \delta t\ \mathcal{D}(\vec{U}^*))/4, \\ \vec{U}^{n+1} &= (\vec{U}^n + 2\vec{U}^{**} - 2\delta t\ \mathcal{D}(\vec{U}^{**}))/3, \end{aligned} \tag{3.8}$$

and the intermediate steps can be represented on the time axis as in Figure 3.1. Clearly, $\vec{U}^*$ is an order-1 approximation of $\vec{U}^{n+1}$; thus it contains similar nonsmooth structures at the same places, and the flag coefficients obtained from its multiresolution transform could be used to compute $\vec{U}^{n+1}$ from $\vec{U}^{**}$, instead of deriving them from the $\vec{U}^{**}$ multiresolution transform. This modification reduces the computational cost of the multilevel algorithm while keeping the same quality in the numerical results.

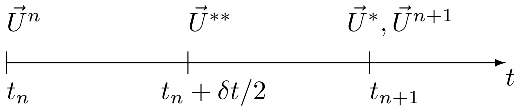


FIG. 3.1. *Representation on the time axis of the intermediate steps of the third-order Runge–Kutta method.*

Remark 3.3. The implementation of the multilevel strategy into an existing code would then amount to the following:

– Define two additional matrices, one to store the scale coefficients of the multiresolution representation of the density, the other to store the flag coefficients.
– Include the multiresolution transform routine. Apply it to the density values according to the guidelines in this section.
– Use the flag to modify the computation of the numerical divergence. Use the numerical flux function of the scheme only when the flag value is 1.

4. Error analysis. In [14] Harten performs a study of the accumulation of the error in the multilevel strategy. When the underlying shock capturing scheme is monotone, Harten shows that the global accumulation error, i.e., the difference between the true solution and the numerical approximation obtained with the multilevel algorithm, can be bounded in terms of the thresholding parameters and the local truncation error of the underlying shock capturing scheme. In addition, if the tolerance for thresholding is of the order of the local truncation error of the scheme, then the multilevel scheme is of the same order as the underlying shock capturing scheme (see [14] for details). The main ingredients in his proof are the stability of the multiresolution transform and the monotonicity of the shock capturing scheme.

The schemes we consider in this paper are not monotone, and an estimate on the global error cannot be obtained. Keeping in mind that we view the multilevel scheme as an acceleration tool, and that our target is to lower the cost that is needed to obtain the numerical solution on the finest grid, we seek only to control the global error between the multilevel and the reference solution. The nonlinearity of the schemes we are considering prevents us from carrying out a rigorous analysis similar to that of [14]; we conjecture that this error can be controlled due to the stability of the multiresolution transform. In section 5.2, we perform several numerical experiments that seem to indicate that

$$||v^{ref} - v^{mult}||_1 \leq C\epsilon^{\alpha} \tag{4.1}$$

for some real number $\alpha > 1$.

5. Numerical experiments. This section is devoted to the presentation and analysis of the results obtained with our multilevel algorithm. We focus on two classical configurations for numerical simulations involving the Euler equations in two dimensions. A detailed description of the flow structure, for both test cases, can be found in [24].

Test A: Double mach reflection of a strong shock. The problem involves a Mach 10 shock in air ($\gamma = 1.4$) which makes a 60^o angle with a reflecting wall. The computational domain is a tunnel 4 units long and 1 unit high, starting at $x = 0$, $y = 0$.

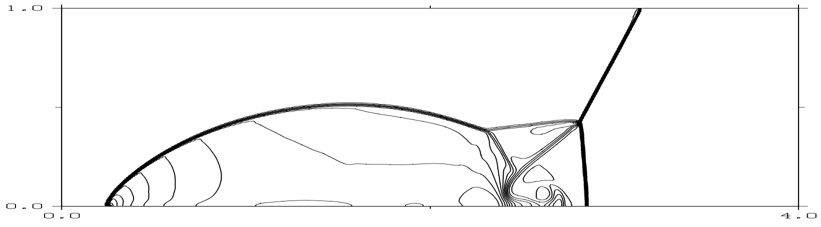


FIG. 5.1. *Density reference solution for Test A at time* $t = 0.2$*, obtained with* 512×128 *points and M-PHM scheme without multiresolution.*

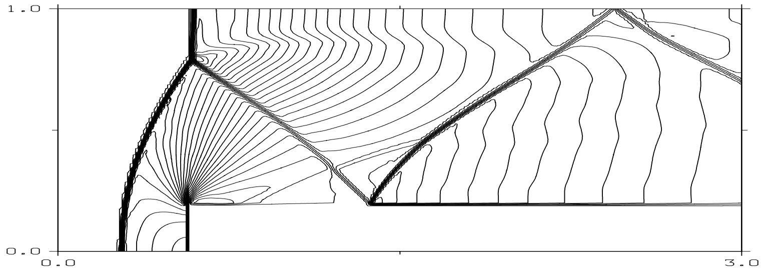


FIG. 5.2. *Density reference solution for Test B at time* $t = 4$*, obtained with* 256×80 *points and M-PHM scheme without multiresolution.*

Initially the shock extends from the point $x = \frac{1}{4}$ at the bottom of the computational domain to the top boundary. The reflecting wall begins at $x = \frac{1}{4}$ on this bottom wall. Postshock conditions, $\vec{U}_{left} = (8., 57.1597, -33.0012, 563.544)$, are assigned at the boundaries located to the left of the shock; the air ahead of the shock is left undisturbed and has density 1.4 and pressure 1. Outflow conditions are applied at the right end of the domain, and the values on the top boundary to the right of the shock are those of undisturbed air.

The finest resolution grid, $\mathcal{G}^0$, that we shall consider for this test problem has 512×128 points. The density obtained at time $t = 0.2$ using M-PHM on $\mathcal{G}^0$ is displayed in Figure 5.1. We see that all the features of the flow are appropriately represented, including the jet-like structure near the reflecting wall. This is our *reference* simulation. We shall apply the multilevel algorithm to this test case with $L = 5$ and $\epsilon = 5 \times 10^{-3}$.

Test B: Mach 3 *wind tunnel with a step.* The problem begins with a uniform Mach 3 flow in a tunnel containing a step. The tunnel is 3 units long and 1 unit wide, and the step is located 0.6 units from the left-hand end of the tunnel and is 0.2 units high. Inflow boundary conditions are applied at the left of the domain and outflow conditions occur at the right end. Along all the walls of the tunnel, reflecting boundary conditions are applied. Initially the tunnel is filled with a gamma-law gas with $\gamma = 1.4$, which has density 1.4, pressure 1.0, and velocity 3.

At time $t = 4$, the flow has a rich and interesting structure that can be accurately described using M-PHM on a grid with 256×80 points, which is then considered as our finest grid, $\mathcal{G}^0$, for this test case. In Figure 5.2, we display the density distribution at time $t = 4$ obtained with M-PHM on $\mathcal{G}^0$. This is our *reference* simulation. We shall apply the multilevel algorithm to this test case with $L = 4$ and $\epsilon = 5 \times 10^{-3}$.

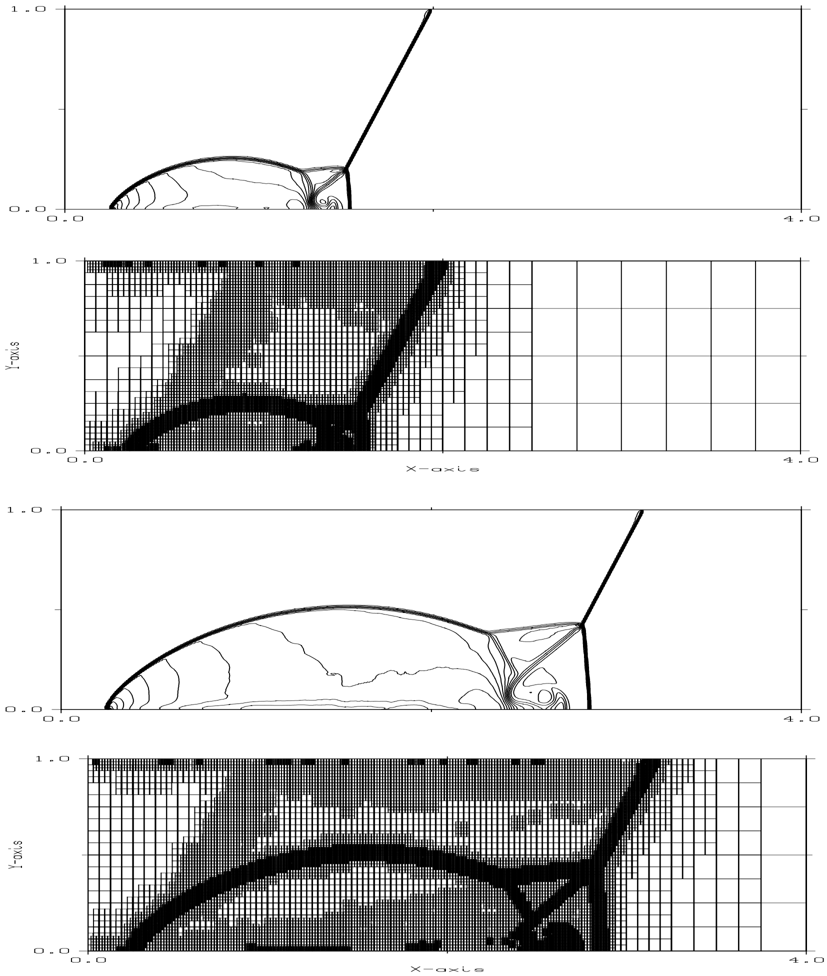


FIG. 5.3. *Density field of the multilevel solution and adaptive grids for Test A at time* $t = 0.1$ *(top) and* $t = 0.2$ *(bottom). Initial grid* $\mathcal{G}^0$ *contains* 512×128 *points,* $L = 5$ *levels, and* $\epsilon = 5 \times 10^{-3}$.

5.1. Test results: Marquina's scheme. In this first set of experiments we apply the multilevel algorithm to the M-PHM scheme in order to compute the solution to the previous test problems.

In Figures 5.3 (Test A), 5.4, and 5.5 (Test B) we display the level curves of the numerical solution obtained with the multilevel algorithm at different times of the flow evolution. For each simulation, we also present a second plot displaying only the points of $\mathcal{G}^0$ where the numerical divergence is computed directly with the HRSC scheme. The graphical display is arranged so that it looks like a structure of *adaptive grids*, similar to those used in numerical simulations involving AMR techniques. The plots of the *adaptive grids* give a very good indication of the amount of work saved by the strategy. It must be pointed out that these plots do not represent, as in AMR, the various grids involved in the computation. We must remember that the multilevel strategy uses the data on the finest grid for the direct flux evaluations. There is only one CFL number, dictated by the finest grid, and the memory requirements correspond to those of the finest grid (in fact they are slightly larger, since we need

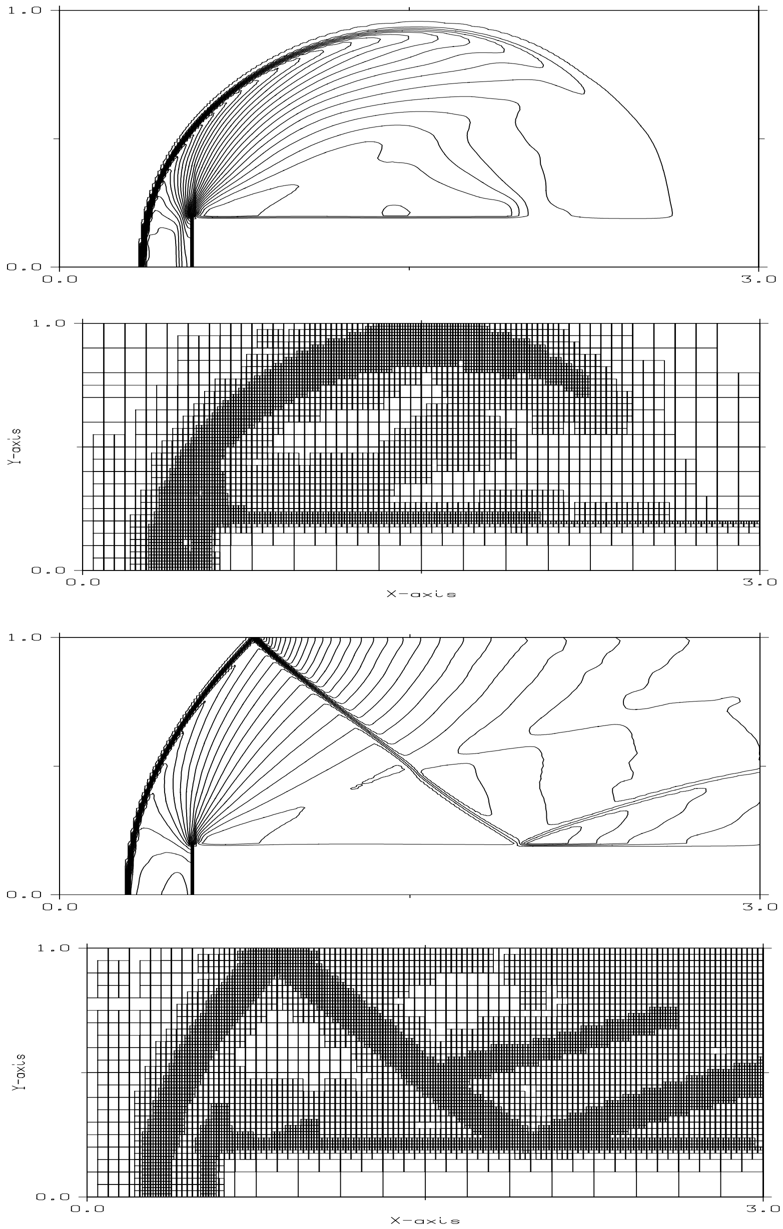


Fig. 5.4. *Density field of multilevel solution and adaptive grids for Test B at time* $t = 0.5$ *(top) and* $t = 1.5$ *(bottom). Initial grid* $\mathcal{G}^0$ *contains* 256×80 *points,* $L = 4$ *levels, and* $\epsilon = 5 \times 10^{-3}$.

two more matrices).

In looking at the plots of level curves, we readily observe that the numerical simulation is of the same "quality" as the reference simulation. The plots of the adaptive grids show that the smoothness analysis performed on the wavelet coefficients is able to localize correctly the nonsmooth structures of the flow. A direct evaluation of the numerical fluxes is being performed in the neighborhood of all singularities, as well as in the shock formation process, and, as a result, the numerical solution presents

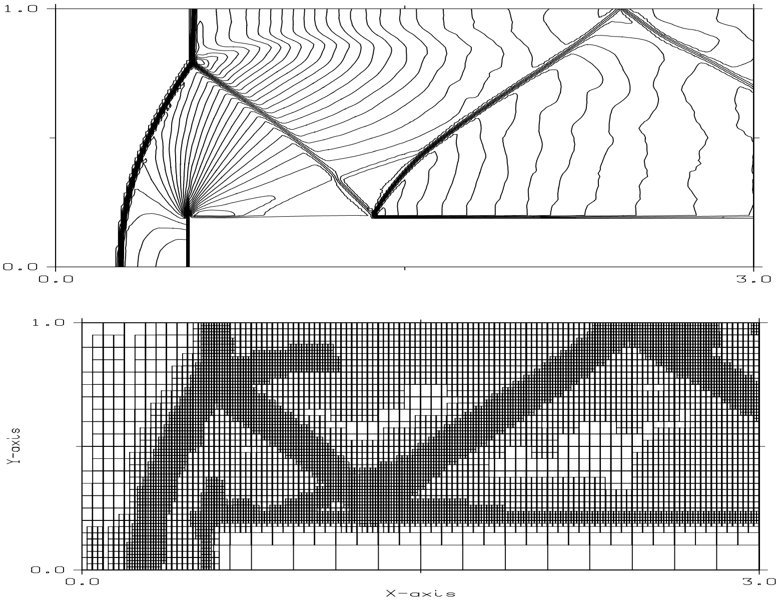


FIG. 5.5. *Same as Figure* 5.4 *for the time* $t = 4$.

the sharp shock profiles that are typical of a third-order scheme such as M-PHM.

The plots of the adaptive grids give additional information also. As observed in [24], the numerical results for Test A are marred by small errors that are due to the description of the initial data and to the fact that the boundary conditions on the top boundary are set to describe the exact motion of the initial Mach 10 shock. These errors are identified as nonsmooth behavior by the multiresolution-based smoothness analysis and, as a consequence, there is some unnecessary refinement in smooth regions, since no shock formation or evolution is taking place there. It is important to notice that this phenomenon occurs for both the reference and multilevel simulations. Through the plots of the *adaptive grid* structure, the occurrence and relative importance of these errors can be clearly appreciated.

Notice also the refinement appearing at reflecting walls in both tests. The problem of dealing with reflecting boundary conditions in high-resolution simulations has been addressed by various authors in recent papers (see, e.g., [11] and references therein), and here the multilevel algorithm can also help to detect which areas of the computational domain are displaying a numerical behavior susceptible to improvement. In addition, it is clear that any improvement with respect to lowering the level of numerical noise close to boundaries will produce in turn an increase in the efficiency of the multilevel algorithm, since the unnecessary refinement will be eliminated.

5.2. Quality and efficiency. As discussed in section 4, the question of quality will be analyzed by measuring the difference between the multilevel solution $\vec{U}^n$ and the reference one, $\vec{U}^n_{ref}$. Our objective is to examine the relation between the tolerance parameter ϵ and the difference $||\vec{U}^n - U^n_{ref}||$, measured in some appropriate norm, which in our case we choose to be the (discrete) l_1-norm. To examine the relation between the tolerance ϵ and the difference $||\vec{U}^n - U^n_{ref}||_1$, we consider the density, for

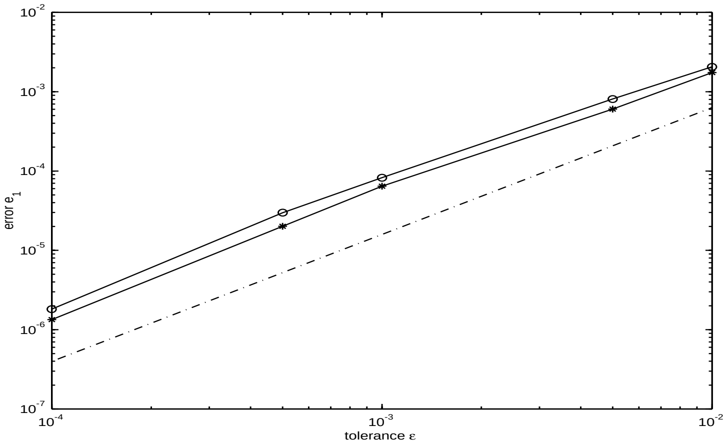


FIG. 5.6. *Error* e_1*, for the density* (∗) *and the pressure* (o)*, between the multilevel algorithm and the reference one versus the tolerance* ϵ*. The dotted line represents the curve of equation:* $\epsilon^{1.6}$.

example, as a representative variable and compute

$$e_1^\rho = \frac{1}{N_p}\sum_{i=0}^{N_x}\sum_{j=0}^{N_y} |\rho_{ij}^n - \rho_{ref_{ij}}^n| / \|\rho_{ref}^n\|_{l_1}, \tag{5.1}$$

where $N_p = (N_x + 1) \times (N_y + 1)$ is the total number of points on the finest grid $\mathcal{G}^0$.

We apply the multilevel algorithm to Test A with $N_p = 128 \times 32$ and $L = 3$ for different values of the tolerance ϵ. The error is measured, for the density ρ and pressure p, at time $t = 0.2$; results are presented on Figure 5.6. It is readily observed that both e_1^ρ and e_1^P decrease with ϵ according to (4.1), with $\alpha = 1.6$. Numerical experimentation indicates that this exponent is solution-dependent, but the behavior is similar in all test cases we have considered (i.e., $\alpha > 1$).

The results of Figure 5.6 imply that the *quality* of the numerical solution obtained with the multilevel scheme, i.e., the closeness to the reference simulation, can be controlled by adjusting the tolerance suitably.

The goal of the multilevel algorithm is to save time in the evaluation of costly numerical flux functions; thus an important quantity is the percentage of numerical divergences computed directly per time step, $\%f$. Table 5.1 (for Test A) and Table 5.2 (for Test B) show the maximum and minimum values for $\%f$ in the simulation. Observe that, for a given test, the finer the grid, the smaller the percentage of direct flux evaluations, since the direct evaluation of the numerical divergence is carried out in a neighborhood of the nonsmooth structures of the flow, and the percentage of computational grid cells involved in these regions decreases when increasing the grid resolution.

A more concrete measure of the efficiency of the multilevel algorithm with respect to the reference simulation is given by θ_{iter}, the cpu gain for a given iteration, and θ, the gain for the global simulation. Introducing t_{ref}^{iter} and t_{mr}^{iter} as the cpu times at iteration $iter$ for the reference and the multilevel algorithm, respectively, θ_{iter} and θ are defined as

$$\theta_{iter} = \frac{t_{ref}^{iter}}{t_{mr}^{iter}} \quad \text{and} \quad \theta = \frac{\sum t_{ref}^{iter}}{\sum t_{mr}^{iter}}. \tag{5.2}$$

Table 5.1
Percentage of resolved flux and cpu gain for Test A at time $t = 0.2$.

Grid size $\mathcal{G}^0$	$\%f_{\min}$ – $\%f_{\max}$	cpu gain θ
128×32	17.6 – 52.7	1.7
256×64	8.9 – 33.2	2.45
512×128	4.5 – 23.2	3.8

Table 5.2
Same as Table 5.1 *for Test B at time $t = 4$.*

Grid size $\mathcal{G}^0$	$\%f_{\min}$ – $\%f_{\max}$	cpu gain θ
128×40	7 – 69.5	0.9
256×80	2.8 – 45	1.4

Table 5.1 (for Test A) and Table 5.2 (for Test B) show the global gain for each simulation. It is obvious that the global gain, θ, is problem dependent. In Figure 5.7, we represent $\theta(t)$. In the early stages of the computation, when there are very few nonsmooth structures in the flow, the gain is quite large; as expected, $\theta(t)$ is a decreasing function, and the gain is larger when we compute on finer grids. The bottom part of Figure 5.7 displays $\%f(t)$ for these simulations. It can be observed that the behavior of θ is roughly inversely proportional to that of $\%f$.

There is an overhead associated to the multilevel computation. In Table 5.3 we show the cpu time for one step of the multilevel algorithm and one stage of the Runge–Kutta method. These results have been obtained with Test A and 512×128 points in $\mathcal{G}^0$ when $\%f$ has its maximum value, 23%. It is worth noting that the overhead caused by the multiresolution transform and the threshold represents only a small part of the total cpu time, $\approx 2\%$, and that most of the time is spent in the numerical divergence evaluation, $\approx 96\%$.

To end this section, we apply the multilevel method, with the same underlying HRSC scheme, to Test A with a very fine grid of 2560×640 points. We set $L = 7$ and $\epsilon = 3.10^{-4}$. In Figure 5.8 we show a zoom of the double-Mach reflection region displaying the level curves of the computed density. The small mesh-size of the underlying grid $\mathcal{G}_0$ used for the simulation reduces the numerical viscosity of the shock capturing scheme and, as a result, we can observe the development of Kelvin–Helmholtz-type instabilities at the contact discontinuities. Such phenomena are not observable for lower-resolution grids, but in fact they correspond to physical effects that have been reported in numerical tests in [16], where a fifth-order shock capturing scheme is being used, and also observed in real experiments [4].

In this case, the percentage of numerical divergences computed directly with M-PHM grows from $\%f = 1\%$ to $\%f = 10\%$, which leads to an estimated global gain $\theta = 7.5$. From the practical point of view, it is important to notice that the estimated computing time for the reference simulation, i.e., full M-PHM, on this fine grid is approximately one month, while the actual time for the multilevel computation was 3–4 days.[1]

5.3. Test results: ENO schemes. As observed by Sjögreen in [22], a multilevel strategy like the one described in this paper should lead to a considerable gain in efficiency with respect to the reference simulation under the following conditions:

1. Large number of grid points.
2. Computationally expensive underlying shock capturing scheme.

[1] All simulations were done with a 350-MHz PC.

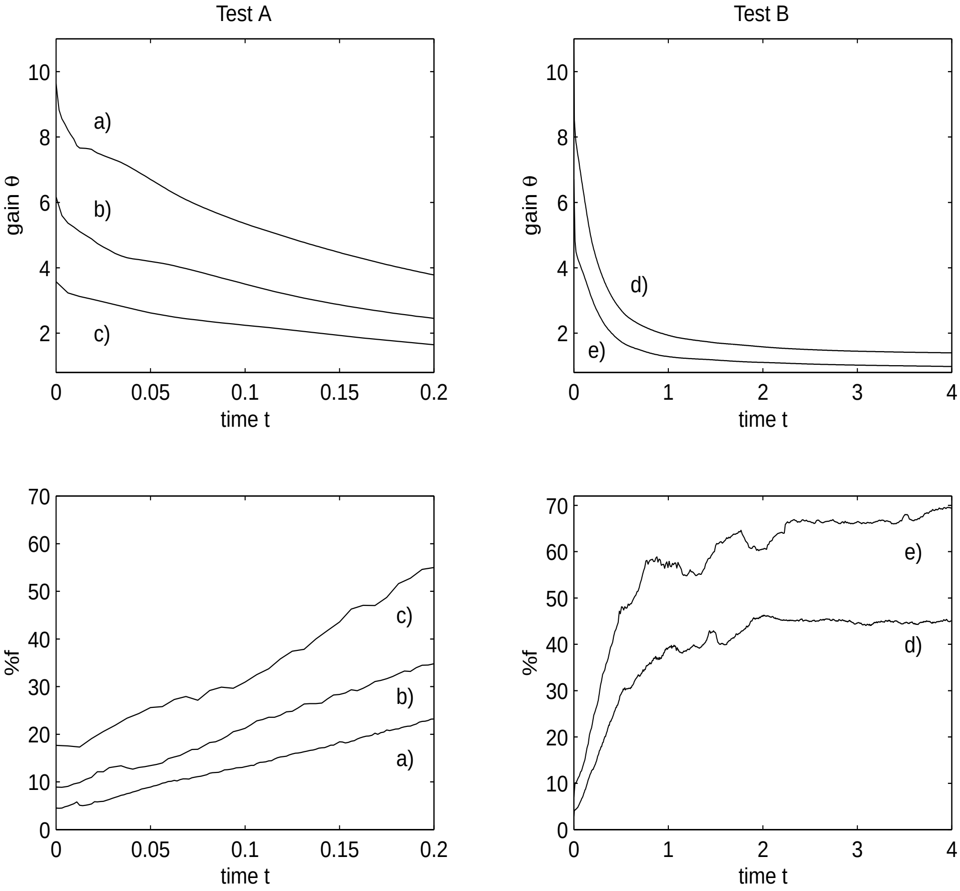


FIG. 5.7. *Time evolution of θ (top) and $\%f$ (bottom) for Test A (left) and Test B (right) and for different initial grid $\mathcal{G}^0$.* (a) 512×128, (b) 256×64, (c) 128×32, (d) 256×80, (e) 128×40.

TABLE 5.3
Cpu time in seconds for the different steps of the multilevel and reference algorithms for one Runge–Kutta stage. These values are obtained with Test A and the largest grid 512×128 *and with* $\%f = 23$.

	Multilevel algorithm	Reference algorithm
Transform	0.06	–
Thresholding	0.08	–
Divergence Evaluation	6.9	13.8
Other	0.15	0.15
Total	7.2	13.95

We have seen this to be the case in the previous section. In this section we would like to compare the computational gain of the multilevel strategy when applied to the M-PHM scheme and to the ENO-3 scheme.

Remark 5.1. It should be mentioned that some entropy corrections, as proposed in [11], are needed near the reflecting wall when using the ENO-3 scheme to avoid the occurrence of a carbuncle phenomenon in the case of the finest grid for Test A; these

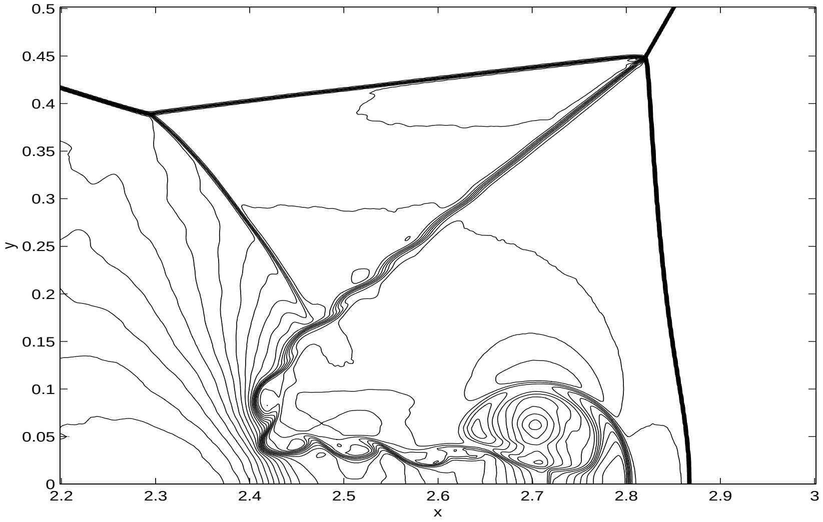


FIG. 5.8. *Zoom of the double Mach reflection region for Test A at* $t = 0.2$ *obtained with* 2560×640 *grid points.*

TABLE 5.4
*Percentage of resolved fluxes and cpu gain for Test A with ENO-*3 *fluxes at time* $t = 0.2$.

Grid size $\mathcal{G}^0$	$\%f_{\min}$ – $\%f_{\max}$	cpu gain θ
128×32	17.6 – 54.2	1.54
256×64	8.9 – 38.4	2.2
512×128	4.5 – 25.7	2.9

corrections are unnecessary for Marquina's scheme. For Test B, a Roe-matrix-related numerical instability develops for grids of size 256×64 or larger, which leads to a crash of the code [9]. These instabilities can be avoided by using appropriate entropy corrections on the bottom wall of the wind tunnel as specified in [11], but we will not pursue this here.

Table 5.4 reports the minimum and maximum percentage of ENO-computed numerical divergences and the global gain θ for the simulation with Test A. Comparing with the results of Table 5.1, we observe that the gain is not as large as in the case of the M-PHM-based multilevel scheme but remains significant. This fact is consistent with Sjögreen's observations, since the cost of a direct evaluation of the numerical divergence by the M-PHM scheme is higher than that of the ENO-3 scheme (by a factor of 2 in our implementation).

It is interesting to display also the gain per iteration θ_{iter} as a function of $\%f$. In Figure 5.9 we represent $\theta_{iter}(\%f)$ for the M-PHM-based and ENO-3-based multilevel strategies. Notice that this representation is more or less independent of the considered test case since the time evolution is not taken into account.

We observe that the gain is much more important for the M-PHM multilevel scheme and small values of $\%f$. Observe also that the difference is reduced when this percentage increases, a fact that could be easily understood considering the following (crude) estimate of θ_{iter}:

$$\theta_{iter} = \frac{N_p t_f}{t_{mr} + t_{thres} + N_f t_f + (N_p - N_f) t_I}$$

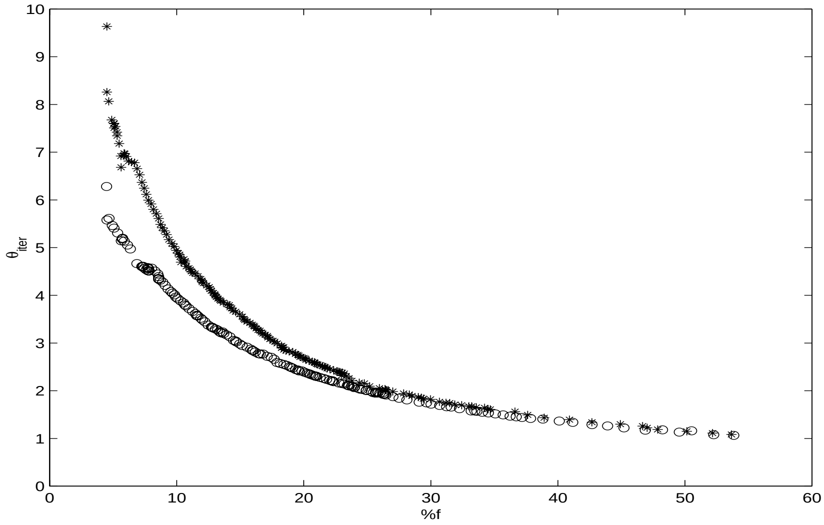


FIG. 5.9. *Gain per iteration θ_{iter} versus the percentage of resolved fluxes $\%f$ for Marquina $(*)$ and ENO (o) schemes.*

$$= \frac{t_f}{\frac{1}{N_p}(t_{mr} + t_{thres}) + \lambda t_f + (1-\lambda)t_I}, \tag{5.3}$$

where t_f is the cpu time to compute one value of the numerical divergence with the HRSC scheme, t_I is the cpu time for one interpolation, and t_{mr} and t_{thres} denote, respectively, the multiresolution transform and thresholding cpu times (which are essentially negligible, as shown in Table 5.3). N_p is the total number of grid points, N_f represents the number of points where the numerical divergence is evaluated directly, and $\lambda = N_f/N_p$.

Considering the same percentage of resolved fluxes for both schemes, i.e., $\%f^M = \%f^E (= 100 * \lambda)$, we can write

$$\frac{\theta^M_{iter}}{\theta^E_{iter}} = \frac{t^M_f}{t^E_f}\,\frac{\frac{1}{N_p}(t_{trans} + t_{thres}) + \lambda t^E_f + (1-\lambda)t_I}{\frac{1}{N_p}(t_{trans} + t_{thres}) + \lambda t^M_f + (1-\lambda)t_I} \sim 2\frac{\lambda + (1-\lambda)\frac{t_I}{t^E_f}}{2\lambda + (1-\lambda)\frac{t_I}{t^E_f}}, \tag{5.4}$$

since in our implementation $t^M_f / t^E_f \sim 2$.

The function

$$g(\lambda) = 2\frac{\lambda + (1-\lambda)\beta}{2\lambda + (1-\lambda)\beta} \tag{5.5}$$

is monotonically decreasing and approaches 1 when λ tends to 1. Moreover, the smaller the ratio β, the faster the convergence to the limit value. In our computations, the ratio $\beta := t_I/t_f$ is approximately 1/56, which leads to $g(.4) = 1.01$ and explains the behavior observed on Figure 5.9. When $\%f \geq 60\%$ the multilevel algorithm is no longer computationally competitive with respect to the reference simulation (see also the first entry in Table 5.2).

6. Conclusions. We have presented a multilevel algorithm designed to reduce the high computational cost associated to HRSC schemes for hyperbolic systems of conservation laws, and we have investigated the application of this multilevel strategy to state-of-the-art HRSC schemes using standard tests for the 2D Euler equations.

The numerical results presented in this paper point out that there is a significant reduction of the computational time when using the multilevel algorithm and confirm Sjögreen's observations in [22]: the more expensive the flux computation, the better the efficiency of the multilevel computation with respect to the reference simulation.

Our multilevel strategy follows the basic design principle of Bihari and Harten in [5], but it is built upon the interpolatory multiresolution framework, instead of the cell-average framework, as in [1, 2, 5, 6, 22, 17]. Through a series of numerical experiments, we show that the strategy we propose offers the possibility of obtaining a high-resolution numerical solution on a very fine grid at the cost of the user's own numerical technique on a much coarser mesh. Its potential users might be researchers performing computational tests with state-of-the-art HRSC methods and using uniform grids.

As in [1, 2, 5, 22], our technique works on the discrete values at the highest resolution level, which need to be always available. There are no memory savings with respect to the reference simulation. In [6, 17], the authors concentrate on solving the evolution equations associated to the (cell-average) scale coefficients. While this option opens the door to what might be an alternative to AMR, a fully adaptive algorithm with selective refinement and real memory savings, it also suffers, in our opinion, from some of the drawbacks of AMR: the need of a special data structure which invariably leads to a very complicated coding structure.

On the other hand, our approach (due in part to the use of the interpolatory framework) is pretty transparent, even to the nonexpert in multiscale analysis, and its incorporation into an existing hydrodynamical code is, in principle, much easier.

REFERENCES

[1] R. Abgrall, *Multiresolution in unstructured meshes: Application to CFD*, in Numerical Methods for Fluid Dynamics, Vol. 5, K. W. Morton and M. J. Baines, eds., Oxford Sci. Publ., Oxford University Press, New York, 1995, pp. 271–276.

[2] R. Abgrall, S. Lantéri, and T. Sonar, *ENO schemes for compressible fluid dynamics*, Z. Angew. Math. Mech., 79 (1999), pp. 3–28.

[3] F. Aràndiga, R. Donat, and A. Harten, *Multiresolution based on weighted averages of the hat function* I: *Linear reconstruction techniques*, SIAM J. Numer. Anal., 36 (1998), pp. 160–203.

[4] G. Ben-Dor, *Shock Wave Reflection Phenomena*, Springer-Verlag, New York, 1992.

[5] B. Bihari and A. Harten, *Multiresolution schemes for the numerical solutions of* 2D *conservation laws*, SIAM J. Sci. Comput., 18 (1997), pp. 315–354.

[6] W. Dahmen, B. Gottschlich-Müller, and S. Müller, *Multiresolution Schemes for Conservation Laws*, Technical report, Bericht 159 IGMP, RWTH Aachen, Germany, 1998.

[7] A. Dolezal and S. Wong, *Relativistic hydrodynamics and essentially non-oscillatory shock capturing schemes*, J. Comput. Phys., 120 (1995), pp. 266–277.

[8] R. Donat and A. Marquina, *Capturing shock reflections: An improved flux formula*, J. Comput. Phys., 125 (1996), pp. 42–58.

[9] R. Donat and A. Marquina, *Computing Strong Shocks in Ultrarelativistic Flows: A Robust Alternative*, Internat. Ser. Numer. Math. 129, Birkhäuser, Basel, 1999.

[10] F. Eulderink, *Numerical Relativistic Hydrodynamics*, Ph.D. thesis, University of Leiden, Leiden, The Netherlands, 1993.

[11] R. Fedkiw, A. Marquina, and B. Merriman, *An isobaric fix for the overheating problem in multimaterial compressible flows*, J. Comput. Phys., 148 (1999), pp. 545–578.

[12] R. Fedkiw, B. Merriman, R. Donat, and S. Osher, *The Penultimate Scheme for Systems of Conservation Laws: Finite Difference ENO with Marquina's Flux Splitting*, UCLA CAM Report 96-18, UCLA, Los Angeles, CA, 1996.

[13] A. Harten, *Discrete multiresolution analysis and generalized wavelets*, J. Appl. Numer. Math., 12 (1993), pp. 153–192.

[14] A. Harten, *Multiresolution algorithms for the numerical solution of hyperbolic conservation laws*, Comm. Pure Appl. Math., 48 (1995), pp. 1305–1342.

[15] A. Harten, *Multiresolution representation of data: A general framework*, SIAM J. Numer. Anal., 33 (1996), pp. 1205–1256.

[16] C. Hu and C.-W. Shu, *Weighted essentially non-oscillatory schemes on triangular meshes*, J. Comput. Phys., 150 (1999), pp. 97–127.

[17] S. Kaber and M. Postel, *Finite volume schemes on triangles coupled with multiresolution analysis*, C. R. Acad. Sci. Paris Sér. I Math., 328 (1999), pp. 817–822.

[18] A. Marquina, *Local piecewise hyperbolic reconstruction of numerical fluxes for nonlinear scalar conservation laws*, SIAM J. Sci. Comput., 15 (1994), pp. 892–915.

[19] J. Marti, E. Muller, J. Font, J. Ibanez, and A. Marquina, *Morphology and dynamics of relativistic jets*, Astrophys. J., 479 (1997), pp. 151–163.

[20] J. Quirk, *A contribution of the great Riemann solver debate*, Internat. J. Numer. Methods Fluids, 18 (1994), pp. 555–574.

[21] C.-W. Shu and S. J. Osher, *Efficient implementation of essentially nonoscillatory shock-capturing schemes.* II, J. Comput. Phys., 83 (1989), pp. 32–78.

[22] B. Sjögreen, *Numerical experiments with the multiresolution scheme for the compressible euler equations*, J. Comput. Phys., 117 (1995), pp. 251–261.

[23] W. Sweldens, *The lifting scheme: A construction of second generation wavelets*, SIAM J. Math. Anal., 29 (1998), pp. 511–546.

[24] P. R. Woodward and P. Colella, *The numerical simulation of two-dimensional fluid flow with strong shocks*, J. Comput. Phys., 54 (1984), pp. 115–173.

SIAM J. SCI. COMPUT.
Vol. 23, No. 3, pp. 824–840

A MODEL FOR THE UNSTABLE MANIFOLD OF THE BURSTING BEHAVIOR IN THE 2D NAVIER–STOKES FLOW[*]

NEJIB SMAOUI[†]

Abstract. Quasi-periodic and bursting behaviors of the two-dimensional (2D) Navier–Stokes flow are analyzed. The tools used are the proper orthogonal decomposition (POD) method and the artificial neural network (ANN) method. The POD is used to extract coherent structures and prominent features from PDE simulations of a quasi-periodic regime and a bursting regime. Eigenfunctions of the two regimes were related by the symmetries of the 2D Navier–Stokes equations. Three eigenfunctions that represent the dynamics of the quasi-periodic regime and two eigenfunctions associated with the unstable manifold of the bursting regime were derived. Calculations of the POD eigenfunctions are performed on the Fourier amplitudes in a comoving frame. Inverse Fourier transform is applied to represent the POD eigenfunctions in both streamfunction and vorticity formulations so that the number of relevant eigenfunctions for streamfunction and vorticity data is the same. Projection onto the two eigenfunctions associated with the unstable manifold reduces the data to two time series. Processing these time series through an ANN results in a low-dimensional model describing the unstable manifold of the bursting regime that can be used to predict the onset of a burst.

Key words. 2D Navier–Stokes equations, proper orthogonal decomposition, symmetry, neural networks

AMS subject classifications. 35K55, 58F39, 65M70

PII. S1064827599355013

1. Introduction.

The use of the artificial neural network (ANN) in modelling various real-world problems has shown a remarkable success in diverse areas such as speech recognition [1], combinatorial optimization [2], image processing, artificial intelligence, control systems [3], and pattern recognition [4]. Recently, ANN has been applied to process nonlinear time series particularly for the prediction of temporally complicated dynamics [5, 6, 7] and the identification of long-term dynamic behavior [8] and bifurcation [9]. A neural network is a logical structure in which multiple nodes, or neurons, operating in parallel, communicate with each other through connecting synapses. The greatest advantage of a neural network is its ability to model complex nonlinear relationships without any a priori assumptions about the nature of the relationships. ANN was also combined with proper orthogonal decomposition to estimate rock permeability [10], to model fluid flow in porous media [11], and to model cellular flames [12].

The proper orthogonal decomposition (POD), also known as Karhunen–Loève decomposition, principal component analysis, and singular value decomposition, has wide applications in scientific problems for both data compression and feature identification [13, 14, 15, 16]. Recently, most applications of the POD method have concentrated on modelling PDE simulations with optimal POD eigenfunctions to generate Galerkin systems that behave like the large-scale numerical simulations of the PDE

[*] Received by the editors April 26, 1999; accepted for publication (in revised form) February 19, 2001; published electronically August 15, 2001. This work was partially supported by Kuwait University (SM182).

http://www.siam.org/journals/sisc/23-3/35501.html

[†] Department of Mathematics and Computer Science, Kuwait University, P.O. Box 5969, Safat 13060, Kuwait (smaoui@mcs.sci.kuniv.edu.kw).

[17, 18, 19, 20]. The system of ordinary differential equations (ODEs), thus obtained, is valid most of the time for a finite range of the bifurcation parameter. However, if the POD method is used to analyze experimental data when there is no known mathematical model for the data, then there is no straight way of reducing the data via a Galerkin projection. Therefore, the use of ANN combined with POD has been shown to be a feasible approach [12].

The objective of this research is to develop a model for the unstable manifold of the bursting behavior in the two-dimensional (2D) Navier–Stokes flow known as Kolmogorov flow. The model will then be used to predict the onset of a burst during the bursting regime; thus, controlling the chaotic regime or delaying its instability. To achieve the above objectives, POD in conjunction with ANN is used. The POD eigenfunctions are obtained from numerical Kolmogorov simulations for a regime characterized by a quasi-periodic behavior at a Reynolds number, $Re = 25.7$. Those eigenfunctions are related to the POD eigenfunctions of the laminar part of the bursting regime at $Re = 25.77$ by the symmetries of the 2D Navier–Stokes equations. The idea of symmetry was first introduced by Sirovich [21]. He suggested the use of symmetry operations to enlarge the available dataset in order to get better averaging behavior. Later on, Aubry, Lian, and Titi [22] have shown that by using a symmetrized dataset, the resulting POD Galerkin system is equivariant with respect to the symmetry which is a necessary condition to achieve a representation of the global phase space. Berkooz and Titi [23] showed that there is no need to apply a symmetry group on an actual realization since all information gained is contained in the eigenfunctions and eigenvalues of the original experiment. Therefore, the symmetry group should be applied on the eigenfunctions. They suggested possible computational savings. Smaoui and Armbruster [17] demonstrated that the computational savings promised in [23] can only be realized if the eigenfunctions are obtained using the snapshot method.

Using the snapshot method, the difference between the data field at $Re = 25.77$ and the symmetrized data at $Re = 25.7$ describes the unstable manifold that leads toward bursting. A POD analysis of this data reveals that 97% of the energy in the unstable manifold is contained in two eigenfunctions. Instead of projecting the 2D Navier–Stokes equations onto these two eigenfunctions, and getting a nonautonomous 2D system of ODEs, we project the unstable data onto those two eigenfunctions to reduce the data to a small number of time series. Finally, the time series is processed through an ANN which results in a low-dimensional, nonlinear dynamical model describing the unstable manifold of the bursting regime.

The remainder of this paper is organized as follows: Section 2 discusses the bursting behavior of the Kolmogorov flow. Section 3 describes the POD analysis and how it is adapted to Kolmogorov flow. Section 4 presents the ANN model for the unstable manifold, and we conclude in section 5.

2. The bursting regime of the Kolmogorov flow. The 2D Kolmogorov flow is the solution of the 2D Navier–Stokes equation with force $\vec{f} = (\frac{1}{R_e}k^3 \cos ky, 0)$ assumed stationary and spatially biperiodic. It was introduced by Kolmogorov in the late 1950s as an example with which to study transition to turbulence. This 2D Kolmogorov flow is studied because of the rich symmetries and the lack of boundary layer associated with it as compared to the Navier–Stokes equations. This will help us gain insight in the understanding of the behaviors of the three-dimensional Navier–Stokes equations. For very low macroscopic Reynolds number R_e, the only stable flow is the plane parallel periodic shear flow $\vec{u}_0 = (k \cos ky, 0)$, usually called the "basic Kolmogorov flow." In a 2 π-periodic square box the equations are

$$\tag{1} \begin{gathered} \frac{\partial \vec{u}}{\partial t} + \vec{u}.\nabla\vec{u} + \nabla p = \frac{1}{R_e}\nabla^2\vec{u} + \vec{f}, \qquad \nabla.\vec{u} = 0, \\ \vec{f} = \left(\tfrac{1}{R_e}k^3\cos ky, 0\right), \qquad 0 \le x, y \le 2\pi. \end{gathered}$$

In the usual stream function representation of the flow, (1) becomes

$$\tag{2} \frac{\partial \Delta\phi'}{\partial t} + \frac{\partial}{\partial x}\left(\Delta\phi'\frac{\partial\phi'}{\partial y}\right) - \frac{\partial}{\partial y}\left(\Delta\phi'\frac{\partial\phi'}{\partial x}\right) = \frac{1}{R_e}\Delta^2\phi' + \frac{\partial f_1}{\partial y} - \frac{\partial f_2}{\partial x}.$$

The stability of the basic Kolmogorov flow, $\sin ky$, is analyzed by perturbing it as $\phi' = \phi + \sin ky$. Thus, the equations for a perturbation of the Kolmogorov flow reduce to

$$\tag{3} \frac{\partial\Delta\phi}{\partial t} = \Delta^2\phi - kR_e\frac{\partial}{\partial x}[\Delta\phi + k^2\phi]\cos ky - R_e\left[\frac{\partial}{\partial x}\left(\Delta\phi\frac{\partial\phi}{\partial y}\right) - \frac{\partial}{\partial y}\left(\Delta\phi\frac{\partial\phi}{\partial x}\right)\right].$$

The symmetries of (3) follow from the Euclidean invariance of the Navier–Stokes equations restricted by the form of the force. All the symmetries are generated by the following transformations:

$$\tag{4} \begin{array}{cccc} T_\xi & : & x & \rightarrow & x+\xi, \\ r & : & (x,y) & \rightarrow & (-x, -y+\frac{\pi}{k}), \\ s & : & (y,\phi) & \rightarrow & (-y,-\phi), \\ t & : & y & \rightarrow & y + \frac{2\pi}{k}. \end{array}$$

Since $(rs)^2 = t$ and $(rs)^{2k} = id$, the two symmetry operations s and rs generate the symmetry group of the regular $2k$-gon D_{2k}. The complete symmetry group of (3) can be written as the semidirect product $D_{2k}\dot{+}SO(2)$.

In the simplest case, $k = 1$, the Kolmogorov flow with 2π-periodic boundary conditions is not unstable for any value of R_e [24, 25, 26], and one is forced to consider $k > 1$. Bifurcations of the 2D Navier–Stokes equation have been investigated in [27, 28, 29, 30, 31, 32] and transitions that occur at higher Reynolds number when $k = 8$ have been studied in [33, 34, 35]. Recently, symmetries and dynamical information for all attractors at low Reynolds number when $k = 2$ have been investigated in [30, 36]. In [30] most bifurcations that occur at low Reynolds number were analyzed and it has been shown that the low-dimensional attractor for the 2D Navier–Stokes flow has a crucial component that lies in the stable eigenspace of the trivial solution. In this study, we focus on a weakly chaotic regime that occurs at $Re = 25.77$ for $k = 2$ (the laminar states are still stable at $Re = 25.70$). The most striking feature of this regime is that the dynamics follows a long laminar regime, then undergoes a burst and settles down to a laminar regime at the same level as before; then other explosions follow. Intervals between bursts are not constant and fluctuate randomly. A study of the Fourier modes [30, 36] shows that

- the laminar regime can be described by a modulated travelling wave with well-defined symmetries;
- successive laminar intervals may not correspond to identical dynamical states but rather to a sequence of states mapped onto each other under some group action.

This kind of behavior was discussed in terms of heteroclinic and homoclinic connections in phase space involving an analysis of the symmetry group of the system [30, 36]. Information is obtained on the unstable manifold of the hyperbolic tori where

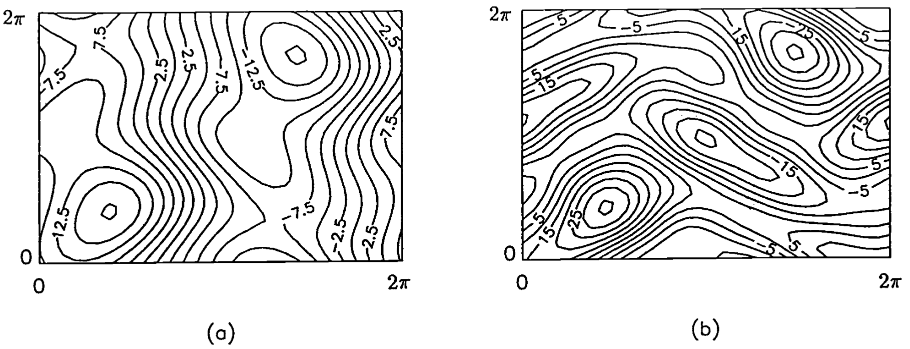


FIG. 1. (a) *Level set for the streamfunction for* $R_e = 25.7$. (b) *Level set for the vorticity for* $R_e = 25.7$.

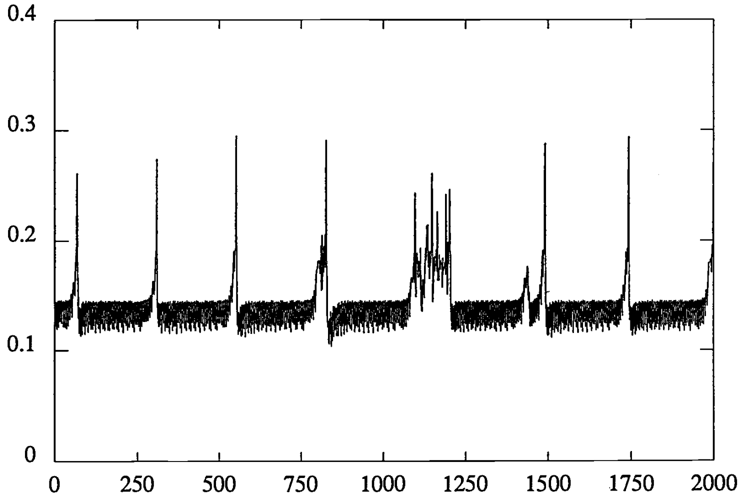


FIG. 2. *Time evolution of the maximum vorticity at* $R_e = 25.77$.

the dynamics is still weakly chaotic and remains for a very long time in a neighborhood of the tori. Figure 1 shows a typical plot of the streamfunction, ϕ, and the vorticity, $w = -\Delta\phi$ at $R_e = 25.7$. Figure 2 depicts the maximal vorticity against time in typical time series at $Re = 25.77$. Figure 2 reveals a long laminar sequence with “microbursts” spaced far apart. We will use POD analysis on these “microbursts” to be able to reveal the trigger mechanism for the bursting behavior.

3. POD. POD is a well-known procedure. It has been used in various fields of applications with different names [13, 14, 15, 16]. We summarize the main idea only briefly. One begins with a given inhomogeneous, chaotic and/or random realization of some state variable $u(x)$, which could be laboratory experimental data or data created by a large-scale simulation as in [21]. We assume that the average of $u(x, t)$, defined as

$$\langle u(x,t) \rangle = \frac{1}{M} \sum_{i=1}^{M} u_i, \tag{5}$$

is equal to zero. The data set $u\left(x,t\right)$ is defined over a finite spatial domain R and finite interval M. A scalar product is given as

$$(u,u')=\int_{R}u\left(x,t\right)u^{'}\left(x,t\right)dx. \tag{6}$$

We choose a function $\psi_1(x)$ such that the projection of the dataset onto all possible functions $\psi(x)$, given by

$$\lambda_1=\frac{1}{M}\sum_{i=1}^{M}\left(\psi_1(x),u_i(x)\right)^2, \tag{7}$$

is maximal, where we can normalize $(\psi_1,\psi_1)=1$. Proceeding inductively, we can find ψ_2 with $(\psi_2,\psi_1)=0$, $(\psi_2,\psi_2)=1$ such that

$$\lambda_2=\frac{1}{M}\sum_{i=1}^{M}\left(\psi_2(x),u_i(x)\right)^2 \tag{8}$$

is maximal. In this way, we can find a unique orthonormal set of functions ψ_n which are the eigenfunctions of the Fredholm-type integral equation

$$\int_{R}K\left(x,y\right)\psi\left(y\right)dy=\lambda\psi\left(x\right), \tag{9}$$

where $K\left(x,y\right)$ is the time-averaged correlation function

$$K\left(x,y\right)=\left\langle u\left(x,t\right)u\left(y,t\right)\right\rangle=\lim_{M\to\infty}\frac{1}{M}\sum_{i=1}^{M}u_i(x)u_i(y), \tag{10}$$

approximated by

$$K\left(x,y\right)=\frac{1}{M}\sum_{i=1}^{M}u_i(x)u_i(y). \tag{11}$$

The ψ_n's are called the empirical eigenfunctions, the coherent structures, etc. It can be shown that any projection of the data $u\left(x,t\right)$ onto a finite set of ψ_k, given by

$$u_N=P_Nu=\sum_{k=1}^{N}a_k\left(t\right)\psi_k\left(x\right), \tag{12}$$

yields amplitudes a_k that are uncorrelated with respect to the averaging process; that is,

$$\left\langle a_j\left(t\right)a_k\left(t\right)\right\rangle=\lambda_j\delta_{jk}, \tag{13}$$

where λ_j is the variance of the data in the direction of the kth eigenfunction. Furthermore, the error given by

$$\epsilon_N=\|u-u_N\|^2 \tag{14}$$

is a minimum over all possible sets of orthonormal functions for any given N, where $\|.\|$ is the L^2-norm. The "energy" of the data is defined as the sum of eigenvalues of the correlation function, $E = \sum_i \lambda_i$. We assign an energy percentage to each eigenfunction based on its associated eigenvalues, i.e., $E_k = \frac{\lambda_k}{E}$.

Since we hope to identify the building blocks of low-dimensional attractors in spatio-temporally complex data, we assume that we have a high resolution in space and a comparatively low-dimensional attractor. That is, we assume $N \gg M$, since we can sample the attractor with only a few snapshots. Two approaches were known in the literature [21] for calculating the correlation function. The first is the direct method in which the covariance matrix is approximated by

$$C = \frac{1}{M} \sum_{i=1}^{M} u_i(x) u_i(x)^T. \tag{15}$$

This covariance matrix is an $N \times N$ matrix. For large N, this matrix can become too large for practical computation. The second method, known as the snapshot method, makes the computation more tractable [17]. In this case, the practical approach for calculating the correlation matrix is not to determine the $N \times N$ correlation matrix but to use the dual approach on the M snapshots, also known as the sample space setting [21]. We take the snapshot vector $u_k(x)$, $k = 1, \ldots, M$, and determine the empirical eigenfunctions $\psi_k(x)$ as an admixture of the given snapshots, i.e.,

$$\psi_k\left(x\right) = \sum_{j=1}^{M} \phi_j^{(k)} u_j\left(x\right), \tag{16}$$

where $\phi_j^{(k)}$ is the jth component of the kth eigenvector and such that the projection of the data is maximal on the $\psi_k\left(x\right)$, as in the previous process. The corresponding eigenvalue problem is then to find the eigenvalues and eigenfunctions of the symmetric $M \times M$ matrix C with

$$C_{ij} = \frac{1}{M}\left(u_i\left(x\right), u_j\left(x\right)\right) = \frac{1}{M} \sum_{k=1}^{N} u_i\left(x_k\right) u_j\left(x_k\right). \tag{17}$$

First, we concentrate on a laminar regime characterized by travelling waves that consist of a travelling structure plus additional time-dependent behavior at $R_e = 25.70$. We perform POD analysis on the Fourier amplitudes in a comoving frame of the travelling wave and find three eigenfunctions that capture 99.8% of the energy of the datafield. The inverse Fourier transformation of those three eigenfunctions is represented in terms of the streamfunction and vorticity formulations in Figures 3 and 4, respectively. The first eigenfunction captures 96.6% of the energy and lies in the invariant subspace Fix(sT_π). This is shown in Figure 5 by first operating with the T_π symmetry on the first streamfunction eigenfunction and by the s symmetry on the resulting eigenfunction. Figure 6 is the same as Figure 5 except for the first vorticity eigenfunction. The difference between the original eigenfunction and the symmetrized one in both streamfunction and vorticity formulations is shown in Figure 5(d) and Figure 6(d), respectively. The second and third eigenfunctions capture 2.6% and 0.62% of the energy, respectively. Inspection shows that those two eigenfunctions span a plane in phase space orthogonal to Fix(sT_π). These findings confirm that the laminar flow can be described as a modulated travelling wave which in phase space is represented by a torus. If we act with the whole symmetry group on the torus, we determine the group orbit of the torus. In particular, we can act with t and r on the solution to obtain different solutions; two solutions in Fix(sT_π) and two in Fix(tsT_π).

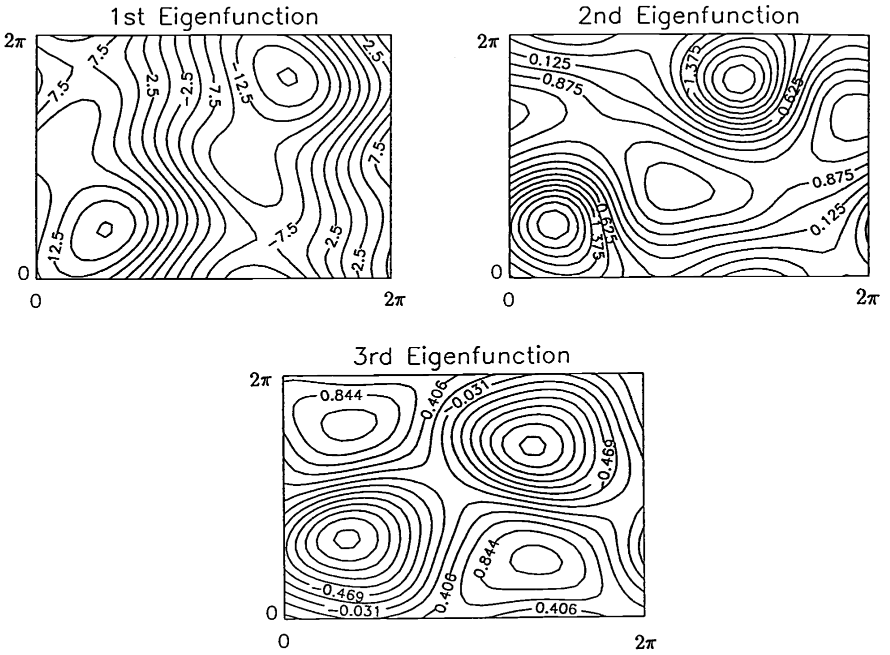


FIG. 3. *The first three POD eigenfunctions given in terms of streamfunction.*

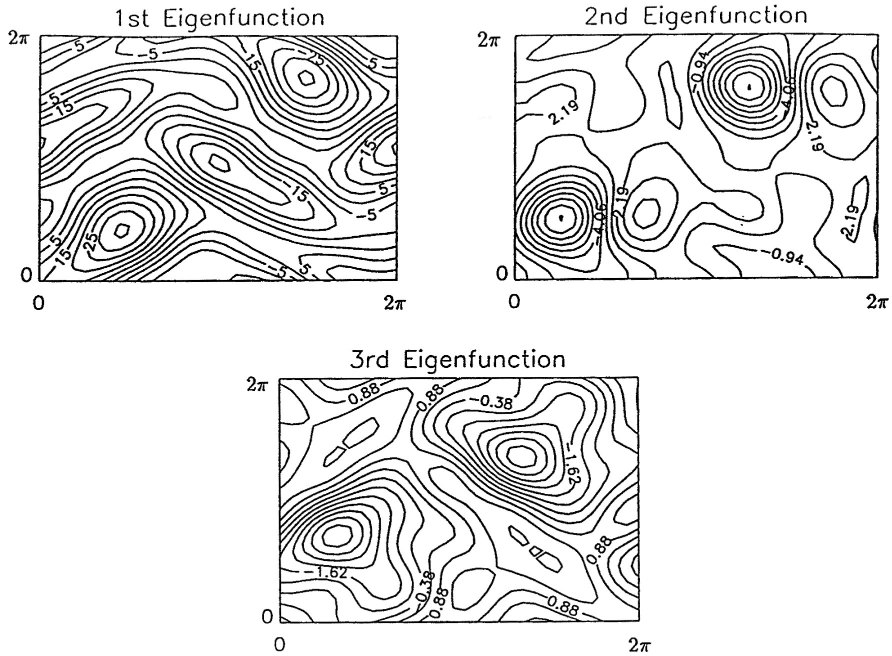


FIG. 4. *The first three POD eigenfunctions given in terms of vorticity.*

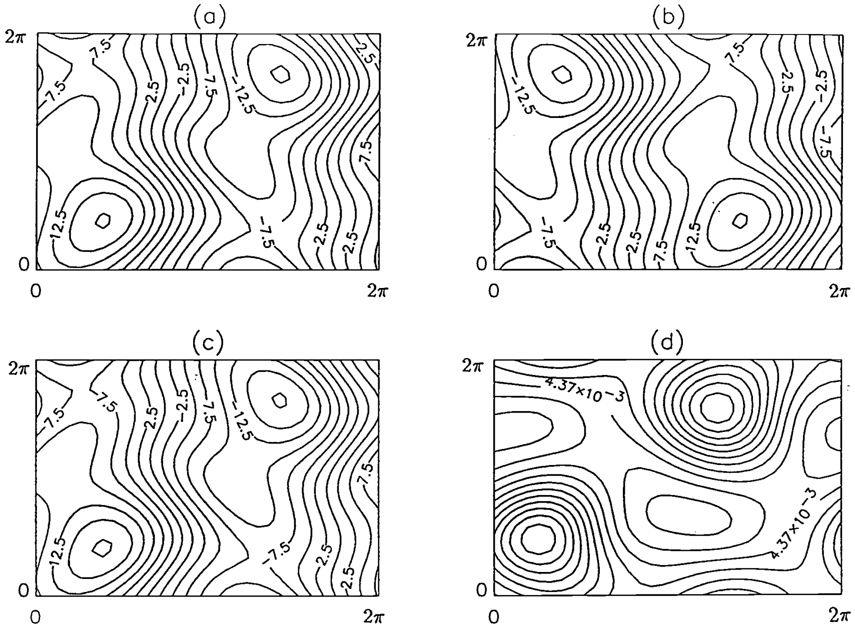


FIG. 5. *The first POD eigenfunction for the streamfunction* (a) *for $R_e = 25.7$,* (b) *for $R_e = 25.7$ after the transformation by T_π, and* (c) *for $R_e = 25.7$ after the transformation by sT_π.* (d) *The difference between the original POD eigenfunction* (a) *and the transformed one* (c).

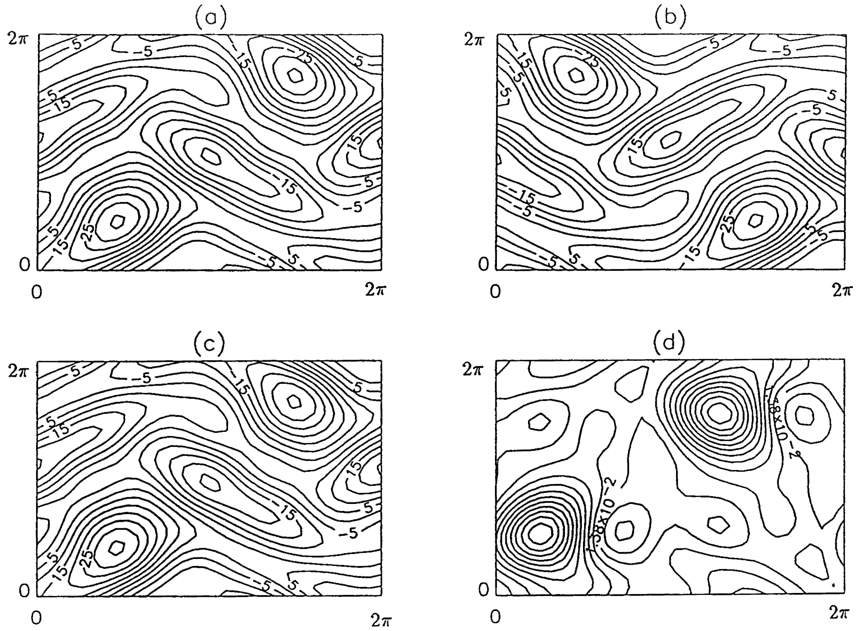


FIG. 6. *The first POD eigenfunction for the vorticity* (a) *for $R_e = 25.7$,* (b) *for $R_e = 25.7$ after the transformation by T_π, and* (c) *for $R_e = 25.7$ after the transformation by sT_π.* (d) *The difference between the original POD eigenfunction* (a) *and the transformed one* (c).

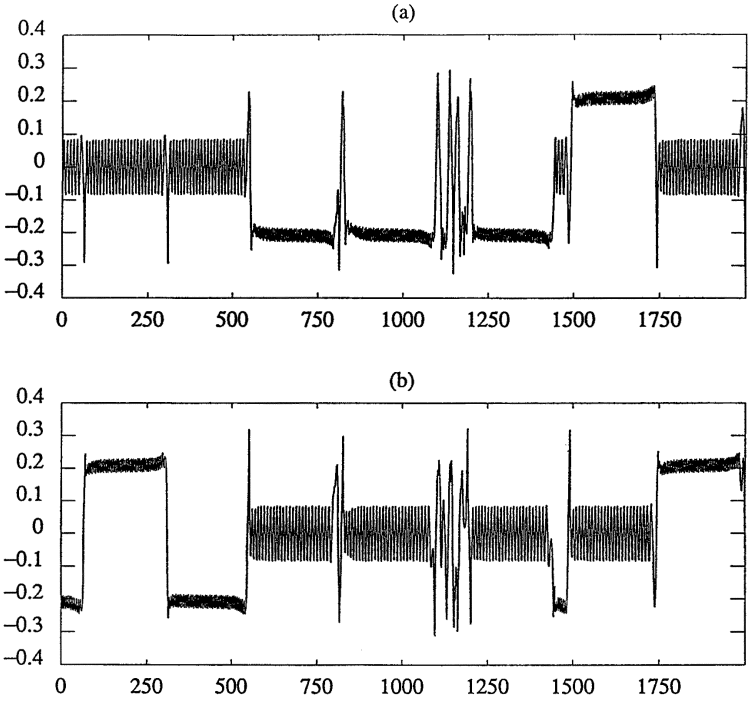


FIG. 7. *Real and imaginary part of the Fourier amplitude e^{iy} for $R_e = 25.77$.*

Next, we consider a slightly higher Reynolds number, namely, $R_e = 25.77$, that leads to a bursting behavior (see Figure 2). This Reynolds number varies very little as compared to the Reynolds number of the laminar regime. If we look closely enough at the Fourier amplitudes of the mode e^{iy} (see Figure 7), we notice that the laminar phases in between bursts look very much like the laminar phase at $Re = 25.7$. In particular, if we plot the imaginary part versus the real part of the Fourier amplitudes of the mode e^{iy}, we observe the four modulated travelling waves discussed earlier (see Figure 8). Therefore, it is reasonable to assume that there exists a close connection between the stable laminar flow at $R_e = 25.7$ and the laminar phase in between bursts at $R_e = 25.77$. Figure 8 also shows that the bursting behavior is not associated with structurally stable heteroclinic cycles. To extract the unstable manifold of the modulated travelling wave that leads toward bursting, we consider a particular laminar window of the dataset in Figure 7. This is obtained by considering the data that look perfectly laminar. These data are contaminated with the dataset associated with the unstable manifold of the bursting regime, and therefore the eigenfunctions extracted from it will not correspond to the pure laminar state. As a result, we use the dataset of the modulated travelling wave at $R_e = 25.7$ to extract the eigenfunctions responsible for the laminar regime. We then relate these eigenfunctions by the symmetry of the Navier–Stokes equations in order to obtain the eigenfunctions responsible for the laminar state between two bursts. This is done by showing that the most energetic eigenfunction of the data in between two bursts is almost identical up to a phase shift to one of the four possible laminar phases that appears at $R_e = 25.70$ (see Figure 9). This phase shift is determined by matching the amplitudes and phase of e^{ix} mode.

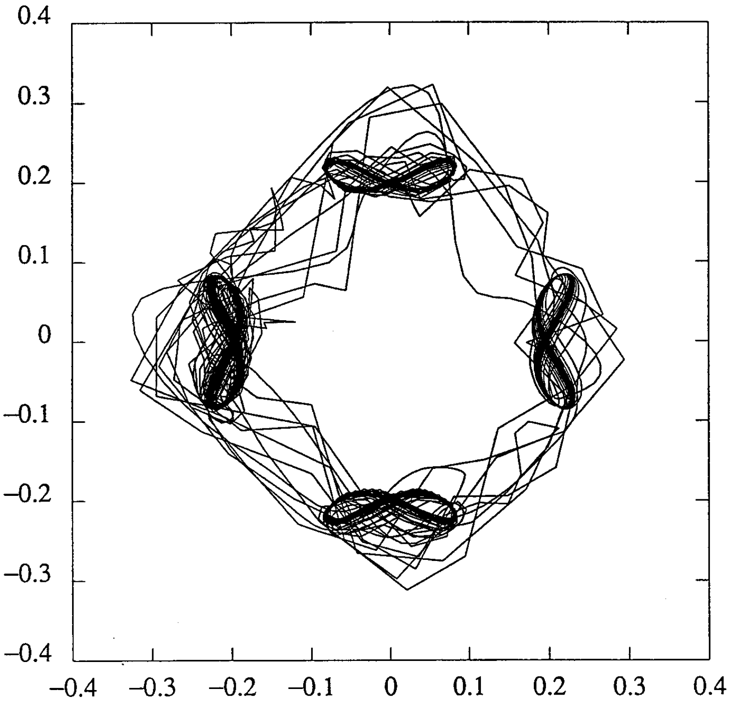


FIG. 8. *Phase plot of the simulation at $R_e = 25.77$. The (x, y) coordinates are the real and imaginary parts of the Fourier amplitude e^{iy}.*

We then shift the first three eigenfunctions of the laminar case by the phase shift thus determined and project the data in between two bursts onto these modes. The difference between this projection and the datafield describes the unstable manifold that leads toward bursting. A POD analysis of this data reveals that 97% of the energy in the unstable manifold is contained in two eigenfunctions (see Figure 10). The time series of the amplitudes of those eigenfunctions can be displayed by projecting the unstable manifold data onto those two POD eigenfunctions (see Figure 11). To model the unstable manifold for this particular datafield, we will use ANN on the time series.

4. ANN. ANN can be considered as a nonlinear dynamic system consisting of a large number of highly interconnected processing nodes. These nodes were inspired by studies of biological nervous systems and the internal operation of the human brain. They operate in parallel, communicating with each other through connecting synapses. Each node receives input, computes an activation, and transmits that activation to other processing nodes. The connection between two nodes is characterized by a weight. Learning occurs while modification of a weight matrix is undertaken. Therefore, what the network computes is highly dependent on how the nodes are interconnected and on the strengths of the connections or weights between them. The architecture of a neural network depends on the node characteristics, network topology, and learning algorithm. A sensitivity study on different neural network architectures led to the choice of an optimal network topology that consists of a six-node input layer, two ten-node hidden layers, both with a nonlinear sigmoid

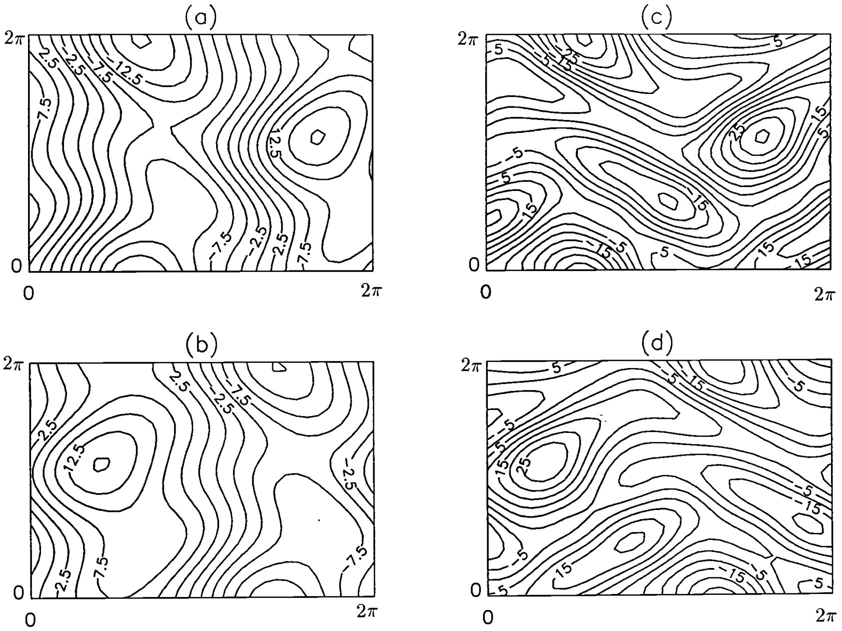


FIG. 9. (a) *The first most energetic eigenfunction for $R_e = 25.7$ given in streamfunction level set.* (b) *The first most energetic eigenfunction for $R_e = 25.77$ given in streamfunction level set for data between two bursts.* (c) *and* (d) *are for the first most energetic vorticity eigenfunction. Notice the shift in x, $T_{3.95}$.*

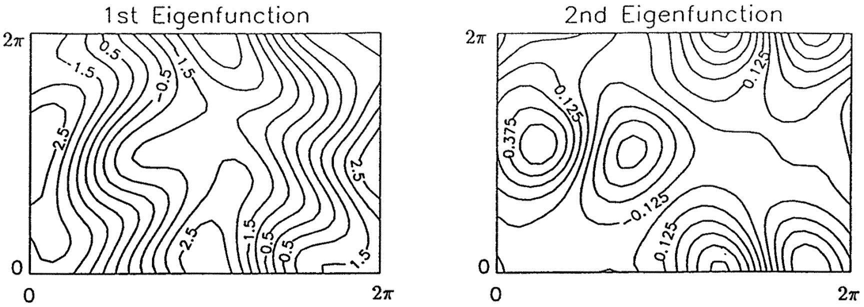


FIG. 10. *The first two most energetic eigenfunction of the unstable manifold given in terms of streamfunction.*

function $\sigma(x) = \tanh x$, and a two-node output layer with linear transfer function is used (see Figure 12). The input layer consists of the time series $a_1(t)$, a suitable number of delays $[a_1(t-\Delta), a_1(t-2\Delta)]$, and the time series $a_2(t)$ with two delays $[a_2(t-\Delta), a_2(t-2\Delta)]$. The output is the predicted values $a_1(t+P)$ and $a_2(t+P)$ of the time series at time $t+P$, that is, we have the following mapping:

$$a_i(t+P) = f(a_j(t), a_j(t-\Delta), a_j(t-2\Delta)); \quad i,j = 1,2, \tag{18}$$

where f is a set of nonlinear functions representing the neural network model, Δ is a time delay, and P is the prediction time into the future.

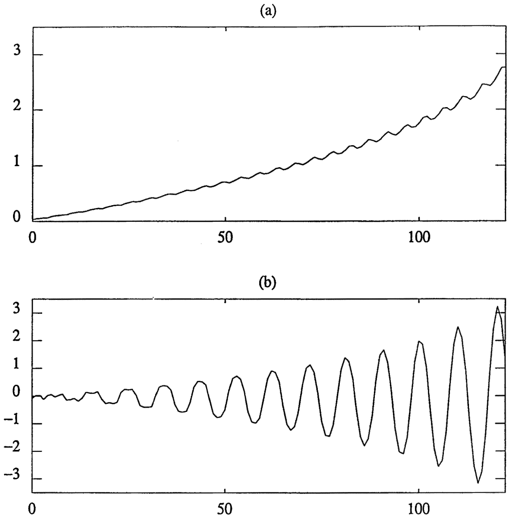


FIG. 11. *The data coefficients of the first two eigenfunctions associated with the unstable manifold.*

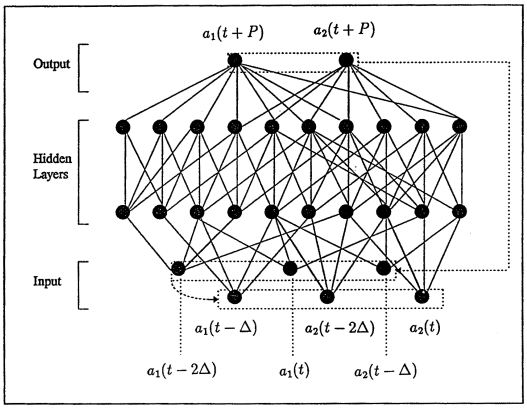


FIG. 12. *ANN paradigm used for the time series of the data coefficients of the unstable manifold.*

Once an architecture has been chosen, it remains to select a learning algorithm. The standard backpropagation learning algorithm has been the most widely used. The backpropagation algorithm is a gradient descent or steepest descent that uses the mean square error of the system over a given set of input-output pairs [37]. The overall error is usually defined as

$$e = \frac{1}{2} \sum_p \sum_k (z_k - y_k)_p^2 \tag{19}$$

over all output nodes k and input vector patterns p. In this notation, z_k is the target vector of the kth output node and y_k is the actual output vector of the kth output node. During the training phase, the weights are successively modified in order to reduce e. The backpropagation algorithm has been demonstrated to be effective in learning but is known to have a slow learning convergence rate because the generalized delta rule is basically a gradient descent scheme with constant step length. Gradient descent is simply the technique where parameters, such as weights, are moved in the opposite direction to the error gradient. Each step down the gradient results in a smaller error until an error minima is reached.

Recently, many learning algorithms and procedures were proposed to improve the modelling capability and learning convergence rate of the backpropagation algorithm [38, 39, 40]. Among the procedures proposed is the momentum procedure. It is basically used to avoid the search process getting stuck in a local minimum rather than a global one, a detailed description of which can be found in [10]. In this study, the training algorithm uses the Levenberg–Marquardt variation of Newton's method to modify the neural network weights. Newton's method can be written as

$$\Delta w = (J^T J)^{-1} J^T e, \tag{20}$$

where J is the Jacobian matrix of derivatives of each error to each weight, and e is an error vector. The problem with the above equation is that $J^T J$ may not be invertible. Thus, the Levenberg–Marquardt algorithm guarantees invertibility when

$$\Delta w = (J^T J + \mu I)^{-1} J^T e, \tag{21}$$

where μ is a scalar. When μ is small the above expression approximates the Gauss–Newton method, and when it is very large it approximates gradient descent. The Gauss–Newton method is used because it is faster and more accurate near an error minimum. Thus, μ is decreased after each successful step and increased only when a step increases the error.

The optimal network architecture discussed earlier with $\Delta = 4$, and $P = 4$, has been trained using data coefficients obtained from the time series of the unstable manifold of one of the four possible modulated travelling waves (i.e., data from vectors corresponding to $420 \leq t \leq 530$; see Figure 7). The choice of Δ implies that a prediction made P time steps into the future past the last actual data coefficients $a_1(t)$, $a_2(t)$ will be made using actual dataset at times: $a_1(t)$, $a_1(t-4)$, $a_1(t-8)$, $a_2(t)$, $a_2(t-4)$, and $a_2(t-8)$. The training was considered successful when the mean square error $e \leq 10^{-4}$. Figure 13 shows both original and predicted time series during the training procedure for the first and second data coefficient $a_1(t)$ and $a_2(t)$. The neural network model is then tested on a different time series associated with another unstable manifold (i.e., data from vectors corresponding to $1640 \leq t \leq 1720$; see Figure 7). Figure 14 shows the excellent agreement between the original time series and the one estimated using the neural network model. Different tests were done on other generated unstable manifolds and excellent prediction has been achieved in each case.

5. Conclusions. We have shown that the use of ANN in conjunction with POD for predicting the data coefficients associated with the unstable manifold of the bursting behavior is very successful. The POD analysis was used to analyze and extract coherent structures from a bursting behavior and a quasi-periodic behavior. Two types of dynamics were observed during the bursting regime: a large-scale, low-dimensional one similar to the one observed in the quasi-periodic regime which can be represented by three coherent structures. Riding on top of that dynamics is a small-scale dynamics characterized by two coherent structures. Those two coherent structures play a role in triggering the burst event and hence in the randomness of the time between

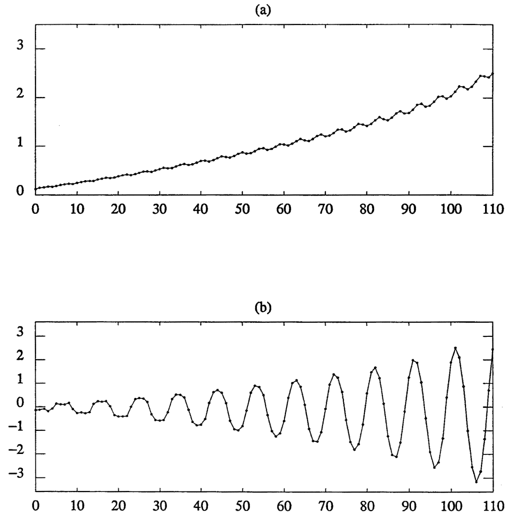


FIG. 13. *Time series plot for the training data set of the two data coefficients $a_1(t)$ and $a_2(t)$. Original (solid line) and predicted (circle) data set values.*

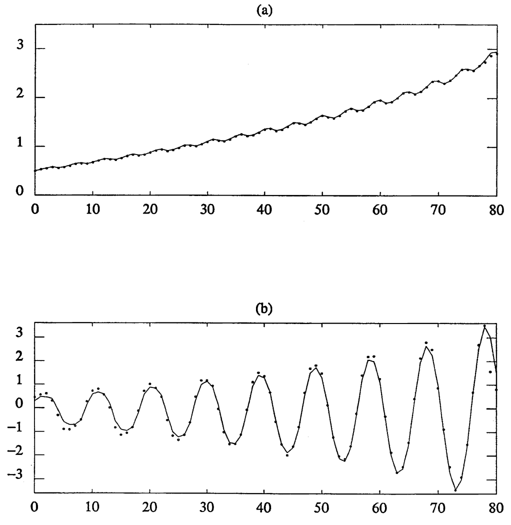


FIG. 14. *Time series plot for the testing data set of the two data coefficients $a_1(t)$ and $a_2(t)$. Original (solid line) and predicted (circle) data set values.*

bursts. Modelling the small-scale dynamics using ANN helps predict the time needed for the occurrence of a burst. The combination of the two approaches has shown to be a powerful method capable of accurately producing a model that can be used to predict the onset of a burst, thus delaying the bursting regime.

This work establishes a foundation for using neural network to predict the small-scale dynamics of more complicated regimes (e.g., $k = 4$ and $k = 8$) which will be the subject of future studies.

Acknowledgments. The author wishes to thank Dieter Armbruster for helpful conversations. The financial support is gratefully acknowledged. Also, the author would like to thank the anonymous referees for their valuable comments, which improved the paper significantly.

REFERENCES

[1] Y. Ephrain and L. Rabiner, *On the relations between modelling approaches for speech recognition*, IEEE Trans. Acoust. Speech Signal Process., 36 (1990), pp. 372–379.

[2] J. J. Hopfield and D. W. Tank, *Neural computation of decisions optimization problems*, Biol. Cybern., 52 (1985), p. 141.

[3] P. Antsaklis, *Neural networks in control systems*, IEEE Contr. Syst. Mag., 10 (1990), pp. 3–5.

[4] J. H. Pao, *Adaptive Pattern Recognition and Neural Networks*, Addison–Wesley, Reading, MA, 1989.

[5] R. Rico-Martinez, K. Krisher, I. G. Kevrekidis, M. C. Kube, and J. L. Hudson, *Discrete vs. continuous-time nonlinear signal processing of Cu electrodissolution data*, Chem. Engrg. Comm., 118 (1992), pp. 25–48.

[6] A. S. Lapedes and R. F. Farber, *Nonlinear Signal Processing Using Neural Networks: Prediction and System Modeling*, Los Alamos Report LA-UR 87-2662, Los Alamos National Laboratory, Los Alamos, NM, 1987.

[7] N. Smaoui, *An artificial neural network noise reduction method for chaotic attractors*, Int. J. Comput. Math., 73 (2000), pp. 417–431.

[8] K. Krisher, R. Rico-Martinez, I. G. Kevrekidis, H. H. Rotermund, G. Ertl, and J. L. Hudson, *Model identification of a spatiotemporally varying catalytic reaction*, AIChE J., 39 (1993), pp. 89–98.

[9] J. L. Hudson, M. Kube, A. Adomaitis, I. G. Kevrekidis, A. S. Lapedes, and R. F. Farber, *Nonlinear signal processing and system identification: Applications to time series from electrochemical reactions*, Chem. Engrg. Sci., 45 (1990), pp. 2075–2081.

[10] N. Smaoui and A. A. Garrouch, *A new approach combining Karhunen-Loéve analysis and artificial neural network for estimating gas sand permeability*, J. Petroleum Sci. Engrg., 18 (1997), pp. 101–112.

[11] N. Smaoui and R. Gharbi, *Using Karhunen-Loeve decomposition and artificial neural network to model fluid flow in porous media*, Appl. Math. Modelling, 24 (2000), pp. 657–675.

[12] N. Smaoui, *Artificial neural network-based low-dimensional model for spatio-temporally varying cellular flames*, Appl. Math. Modelling, 21 (1997), pp. 739–748.

[13] J. L. Lumley, *The structure of inhomogeneous turbulent flows*, in Atmospheric Turbulence and Radio Wave Propagation, A. M. Yaglom and V.I. Tatarski, eds., Nauka, Moscow, 1967, pp. 166–178.

[14] I. T. Jollife, *Principal Component Analysis*, Springer-Verlag, New York, 1986.

[15] R. C. Gonzalez and P. Wintz, *Digital Image Processing*, 2nd ed., Addison–Wesley, Reading, MA, 1987, pp. 122–130.

[16] P. Holmes, J. L. Lumley, and G. Berkooz, *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*, Cambridge University Press, Cambridge, UK, 1996.

[17] N. Smaoui and D. Armbruster, *Symmetry and the Karhunen–Loève analysis*, SIAM J. Sci. Comput., 18 (1997), pp. 1526–1532.

[18] N. Aubry, P. Holmes, J. L. Lumley, and E. Stone, *The dynamics of coherent structures in the wall region of a turbulent boundary layer*, J. Fluid Mech., 192 (1988), pp. 115–173.

[19] G. Berkooz, P. Holmes, and J. L. Lumley, *The proper orthogonal decomposition in the analysis of turbulent flow*, Ann. Rev. Fluid Mech., 25 (1993), pp. 539–575.

[20] M. Kirby and D. Armbruster, *Reconstructing phase-space from PDE simulations*, Z. Angew. Math. Phys., 43 (1992), pp. 999–1022.

[21] L. Sirovich, *Turbulence and the dynamics of coherent structures.* I. *Coherent structures*, Quart. Appl. Math., 45 (1987), pp. 561–571.

[22] N. Aubry, W. Y. Lian, and E. S. Titi, *Preserving symmetries in the proper orthogonal decomposition*, SIAM J. Sci. Comput., 14 (1993), pp. 483–505.

[23] G. Berkooz and E. Titi, *Galerkin projections and the proper orthogonal decomposition for equivariant equations*, Phys. Lett. A, 174 (1993), pp. 539–575.

[24] V. I. Arnold and L. D. Meshalkin, *A. N. Kolmogorov's seminar on selected problems of analysis* (1958–1959), Uspekhi Mat. Nauk, 15 (1960), pp. 247–250.

[25] L. D. Meshalkin and Ia. G. Sinai, *Investigation of the stability of a stationary solution of a system of equations for the plane movement of an incompressible viscous fluid*, J. Appl. Math. Mech., 25 (1961), pp. 1700–1705.

[26] C. Marchioro, *An example of absence of turbulence for any Reynolds number*, Comm. Math. Phys., 105 (1986), pp. 99–106.

[27] Z. S. She, *Large-scale dynamics and transition to turbulence in the two-dimensional Kolmogorov flow*, in Current Trends in Turbulence Research, H. Branover, M. Mond, and Y. Unger, eds., AIAA, Washington, D.C., 1988, pp. 374–396.

[28] Z. S. She, *Metastability and vortex pairing in the Kolmogorov flow*, Phys. Lett. A, 124 (1987), pp. 161–164.

[29] N. Platt, L. Sirovich, and N. Fitzmaurice, *An investigation of chaotic Kolmogorov flow*, Phys. Fluids A, 3 (1991), pp. 681–696.

[30] D. Armbruster, B. Nicolaenko, N. Smaoui, and P. Chossat, *Symmetries and dynamics for* 2*-D Navier-Stokes flow*, Phys. D, 95 (1996), pp. 81–93.

[31] M. S. Jolly, *Bifurcation computations on an approximate inertial manifold for the* 2*-D Navier-Stokes equations*, Phys. D, 63 (1993), pp. 8–20.

[32] M. S. Jolly and C. Xiong, *On computing the long-time solution of the two-dimensional Navier-Stokes equations*, Theoret. Comp. Fluid Dyn., 7 (1995), pp. 261–278.

[33] B. Nicolaenko and Z. S. She, *Symmetry-breaking homoclinic chaos in the Kolmogorov flows*, in Nonlinear World, International Workshop on Nonlinear and Turbulent Processes in Physics, Kiev, 1989, V. G. Baryakhtar et al., eds., World Scientific, Singapore, 1990, pp. 602–618.

[34] B. Nicolaenko and Z. S. She, *Temporal intermittency and turbulence production in the Kolmogorov flow*, in Topological Fluid Mechanics, H. K. Moffatt, ed., Cambridge University Press, Cambridge, UK, 1990, pp. 256–277.

[35] B. Nicolaenko and Z. S. She, *Symmetry breaking homoclinic chaos and vorticity bursts in periodic Navier-Stokes flows*, Eur. J. Mech. B Fluids, 10 (1991), pp. 67–74.

[36] D. Armbruster, B. Nicolaenko, N. Smaoui, and P. Chossat, *Analyzing bifurcations in the Kolmogorov flow equations*, in Dynamics, Bifurcations and Symmetries, P. Chossat, ed., Kluwer, Dordrecht, The Netherlands, 1994, pp. 11–33.

[37] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Learning internal representations by error propagations*, in Parallel Distributed Processing 1: Foundations, D. E. Rumelhart and J. L. McClelland, eds., MIT Press, Cambridge, MA, 1986, pp. 318–362.

[38] T. Ash, *Dynamic Node Creation in Backpropagation Networks*, ICS report 8901, Institute of Cognitive Science, University of California, San Diego, La Jolla, CA, 1989.

[39] S. Becker and Y. Le Cun, *Improving the convergence of backpropagation learning with second order method*, in Proceedings of the 1988 Connectionist Models Summer School, D. S. Touretzky, G. Hinton, and T. Sejnowski, eds., Morgan-Kaufmann, San Mateo, CA, 1988, pp. 29–36.

[40] S. E. Fahlman, *Fast learning variations on backpropagation: An empirical study*, in Proceedings of the 1988 Connectionist Models Summer School, D. S. Touretzky, G. Hinton, and T. Sejnowski, eds., Morgan-Kaufmann, San Mateo, CA, 1988, pp. 38–51.

SIAM J. SCI. COMPUT.
Vol. 23, No. 3, pp. 840–865

A MINIMAX METHOD FOR FINDING MULTIPLE CRITICAL POINTS AND ITS APPLICATIONS TO SEMILINEAR PDES[*]

YONGXIN LI[†] AND JIANXIN ZHOU[‡]

Abstract. Most minimax theorems in critical point theory require one to solve a two-level *global* optimization problem and therefore are not for algorithm implementation. The objective of this research is to develop numerical algorithms and corresponding mathematical theory for finding multiple saddle points in a stable way. In this paper, inspired by the numerical works of Choi–McKenna and Ding–Costa–Chen, and the idea to define a solution submanifold, some local minimax theorems are established which require us to solve only a two-level *local* optimization problem. Based on the local theory, a new local numerical minimax method for finding multiple saddle points is developed. The local theory is applied, and the numerical method is implemented successfully to solve a class of semilinear elliptic boundary value problems for multiple solutions on some nonconvex, non star-shaped and multiconnected domains. Numerical solutions are illustrated by their graphics for visualization. In a subsequent paper [Y. Li and J. Zhou, *Convergence results of a minimax method for finding critical points*, in review], we establish some convergence results for the algorithm.

Key words. multiple saddle point, Morse index, local minimax, semilinear PDEs

AMS subject classifications. 58E05, 58E30, 35A40, 35A65

PII. S1064827599365641

1. Introduction.

Multiple solutions with different performance and instability indices exist in many nonlinear problems in natural and social sciences [33, 30, 24, 36, 23]. When cases are variational, the problems can be reduced to solving the Euler–Lagrange equation

$$J'(u) = 0, \tag{1.1}$$

where J, called *a generic energy functional*, is a C^1-functional on a Banach space H, and J' or ∇J is its Frechet derivative. A solution to the Euler–Lagrange equation (1.1) is called *a critical point* of J. The first candidates for critical points are the local maxima and minima to which the classical critical point theory was devoted in calculus of variation. Traditional numerical methods focus on finding such stable solutions. Critical points that are not local extrema are called *saddle points*, that is, critical points u^* of J for which any neighborhood of u^* in H contains points v, w such that (s.t.) $J(v) < J(u^*) < J(w)$. In physical systems, saddle points appear as *unstable* equilibria or transient excited states. Note that this definition is different from and much more general than the saddle point in optimization and game theory in which a splitting structure for the space H is required to be known in advance and which is therefore not used in critical point theory.

A number $c \in \mathbb{R}$ is a *critical value* of J if $J(\hat{u}) = c$ for some critical point $\hat{u}$. For a critical value c, the set $J^{-1}(c)$ is called a *critical level*. When the second Frechet derivative J'' exists at a critical point $\hat{u}$, then $\hat{u}$ is said to be *nondegenerate* if $J''(\hat{u})$ is invertible. Otherwise, $\hat{u}$ is said to be *degenerate*.

[*] Received by the editors January 19, 2000; accepted for publication (in revised form) December 12, 2000; published electronically August 29, 2001.
http://www.siam.org/journals/sisc/23-3/36564.html

[†] IBM T. J. Watson Research Center, Yorktown Hts, NY 10598.

[‡] Department of Mathematics, Texas A&M University, College Station, TX 77843 (Jianxin.Zhou@math.tamu.edu). This author was supported in part by NSF grant DMS 96-10076.

Stability is one of the main concerns in control and system design. On the other hand, in many applications, higher maneuverability and performance are desirable, in particular, in system design for emergency or combat machineries. Unstable solutions may have much higher maneuverability and performance indices.

Can one find a way to provide a choice or balance between instability and maneuverability or performance indices? Thus one needs to solve for multiple solutions and then study their individual properties.

Numerically finding such unstable solutions in a stable way is very challenging. So far, there is virtually no theory existing in the literature to devise such a feasible numerical algorithm. The objective of this research project is to systematically develop effective numerical algorithms and corresponding mathematical theory for finding multiple saddle points in a stable way. To do so, we need to know the local mathematical structure of a critical point and its connection to a critical point at the next critical level. We do not intend to establish *new* existence theorems.

Structure and behavior of critical points have attracted the attention of many researchers. In 1925, Morse proved that if $\hat{u}$ is a nondegenerate critical point of a real function J of n variables, then there exists a neighborhood $\mathcal{N}(\hat{u})$ of $\hat{u}$ and a local homeomorphism h from $\mathcal{N}(\hat{u})$ into H s.t.

$$J(h(u)) = J(\hat{u}) + \frac{1}{2}\langle J''(\hat{u})u, u\rangle \quad \forall u \in \mathcal{N}(\hat{u}).$$

That is, J behaves locally like a quadratic function around a nondegenerate critical point. This result, called the Morse lemma, has been extended to a real-valued infinite-dimensional functional [10]. Therefore, critical levels with a local minimum, if it exists, at the bottom can be imagined.

The *Morse index* (MI) of a critical point $\hat{u}$ of a real-valued functional J is the maximal dimension of a subspace of H on which the operator $J''(\hat{u})$ is negative definite; the *nullity* of a critical point $\hat{u}$ is the dimension of the null-space of $J''(\hat{u})$. Thus, for a nondegenerate critical point, if its MI $= 0$, then it is a local minimizer and a stable solution, and if its MI > 0, then it is a saddle point, an unstable solution.

DEFINITION 1.1. *A point $v \in H$ is called a* descent (*ascent*) *direction of J at a critical point $\hat{u}$ if there exists $\delta > 0$ s.t.*

$$J(\hat{u} + tv) < \ (>)\, J(\hat{u}) \quad \forall 0 < |t| < \delta.$$

Thus J has at least k linearly independent descent directions at a critical point with MI $= k$.

Many boundary value problems (BVPs) are equivalent to solving [33]

$$A(u) = 0 \tag{1.2}$$

for a solution $u \in H$ and an operator $A : H \to H^*$. When the problem is variational, there exists $J : H \to \mathbb{R}$ s.t.

$$\langle A(u), v\rangle = \langle J'(u), v\rangle = \lim_{t\to 0} \frac{J(u+tv) - J(u)}{t} \quad \forall v \in H, \tag{1.3}$$

or $A(u) = J'(u)$. Thus $\hat{u}$ is a (weak) solution to (1.2) if and only if $\hat{u}$ is a critical point of J.

The following semilinear BVP is our model problem in this paper; it is known that this model has originated from many applications in physics, engineering, biology, ecology, geometry, etc. Consider

$$\Delta u(x) - \ell u(x) + f(x, u(x)) = 0, \quad x \in \Omega, \tag{1.4}$$

for $u \in H$ with either the zero Dirichlet boundary condition (BC) or the zero Neumann BC, where Ω is a bounded open domain in $\mathbb{R}^N$, and f is a nonlinear function of $(x, u(x))$ with $u \in H$. For the zero Dirichlet BC, we let $H = H_0^1(\Omega)$ and $\ell \geq 0$; for the zero Neumann BC, we let $H = H^1(\Omega)$ and $\ell > 0$, where $H^1(\Omega)$ is the Sobolev space $W^{1,2}(\Omega)$ and $H_0^1(\Omega) = \{v \in H^1 : v(x) = 0, x \in \partial\Omega\}$ [1]. The associated variational functional is the energy

$$J(u) = \int_\Omega \left\{ \frac{1}{2}|\nabla u(x)|^2 + \frac{1}{2}\ell u^2(x) - F(x, u(x)) \right\} dx, \tag{1.5}$$

where $F(x, t) = \int_0^t f(x, \tau) d\tau$ satisfies the assumptions (h1)–(h5) as stated in section 4 and $u \in H$. Then a direct computation shows that a point $\hat{u} \in H$ is a critical point of J in H if and only if $\hat{u}$ is a weak solution to the BVP (1.4) and, therefore, a classical solution to (1.4) by a standard elliptic regularity argument.

Since Ljusternik–Shnirelman (1934), under a deformation assumption, proved the existence of a saddle point as a minimax solution, i.e., a solution to a two-level optimization problem

$$\min_{A \in \mathcal{A}} \max_{v \in A} J(v) \tag{1.6}$$

for some collection $\mathcal{A}$ of subsets A in H, the minimax principle has become the most popular approach in critical point theory. Note that there is another minimax approach in multilevel optimization and game theory, which is a two-level optimization of the form

$$\min_{x \in X} \max_{y \in Y} J(x, y),$$

where $H = X \times Y$ for some subspaces X and Y. Due to its splitting structure, this minimax approach prevents us from turning around in a search for a critical point. Although, it is known that if $J''(\hat{u})$ is self-adjoint and Fredholm, H has such a splitting structure around a nondegenerate critical point $\hat{u}$ according to the Morse theory. But such a splitting structure depends on $\hat{u}$ and is not known until one finds $\hat{u}$. Thus this minimax approach does not help in searching for a critical point and is not used in critical point theory.

It is the mountain pass lemma, proved in 1973 by Ambrosetti–Rabinowitz [3] by constructing a deformation, that sets a milestone in nonlinear analysis. Since then, minimax theorems have gained great popularity in the study of nonlinear PDEs and dynamic systems. Various minimax theorems, such as linking and saddle point theorems, have been successfully established to prove the existence of multiple solutions to various nonlinear PDEs and dynamic systems [5, 8, 10, 13, 14, 16, 22, 23, 24, 26, 27, 30, 31, 32, 33, 34, 36]. For example, for a semilinear elliptic equation $\Delta u + f(u) = 0$ with the zero Dirichlet BC, when $f(u)$ is superlinear, it is known that if f is odd, then infinitely many solutions exist; otherwise, a third sign-changing solution (MI = 2) exists [34, 5, 9] in addition to a positive and a negative solution.

When multiple solutions exist in a nonlinear system, some are stable and others are unstable. A stable solution (MI = 0) can be found through local minimization techniques or a monotone iterative scheme (see, e.g., [2, 7, 11, 15, 18, 19, 28, 29]). However, relatively little is known in the literature on constructing algorithms to compute such unstable saddle points in a numerically stable way. One might mention Newton's method. Note that zeros and critical points are different concepts. Without knowing or using the local structure of a (degenerate) critical point, the usual Newton method will not be effective or stable. When a local minimization is involved in a quasi-Newton method, it will lead to a local minimum, in the case of (1.4), the zero.

Most minimax theorems in the literature mainly focus on the existence issue. They require one to solve a two-level *global* optimization problem, i.e., (constrained) global maximizations at the first level and a global minimization at the second level, and therefore are not for algorithm implementation.

By studying the mountain pass lemma and using an idea from Aubin–Ekeland [4], in 1993 Choi–McKenna [12] proposed a numerical minimax algorithm, called a mountain pass method, to solve the model problem basically for a solution with MI = 1. The algorithm opens a brand new door to numerically compute unstable solutions. The algorithm has been modified in [17] and further revised in [11]. Since the function J in [12] has only one maximum along each direction, whether or not it is a local or global maximum at the first level is not a concern there. The merit of this algorithm is that (a) at the first level, a maximization is taken over an affine line starting from 0, and (b) a steepest descent direction is used to search for a local minimum at the second level. In contrast, the mountain pass lemma requires a maximization on every continuous path connecting 0 and a given point v at the first level and then a global minimization at the second level. Thus the method in [12] cannot be justified by the mountain pass lemma. An earlier result of Ding–Ni [27], which proved for the model problem the existence of a saddle point as a minimax solution that requires a unique maximization on each affine line starting from 0 at the first level and a global minimization at the second level, can only be viewed as a partial justification.

In [11], for a class of functionals, a very simple scheme called a scaling iterative method is designed to find a solution with MI = 1. That approach is not based on functional analysis. A partial justification of that algorithm is also given there.

A high linking theorem for the existence of a third solution (MI = 2) is proved in [34] by constructing a "local link" at a mountain pass solution. Motivated by this idea, a numerical high linking method is proposed in [17] by Ding–Costa–Chen to solve the model problem for a sign-changing solution (MI = 2). The basic idea is to assume a mountain pass solution w_1 has been found, and by using an ascent direction and a descent direction at w_1, one can form a triangle as a "local linking." Then one can proceed to find a maximum on this triangle. If the maximum is inside the triangle, go to the next step; otherwise, deform the triangle so as to contain this point as an interior point, and continue to search for an interior maximum. This method uses constrained maximizations at the first level and a local minimization at the second level. Since in the original version of the high linking theorem, the argument never left a mountain pass solution, and a global minimization is required at the second level, the theorem itself cannot serve as a justification of that algorithm. Accordingly, the theoretical justification of that algorithm will be very difficult. The problem is that once an iterate has departed a mountain pass solution, the triangle is no longer a "local linking"; the triangle may degenerate. However, this is the first time in the literature that the idea of a "local linking" is used to find solutions basically with MI = 2.

Inspired by the above numerical works, we developed a new minimax method for finding multiple saddle points. Many numerical results including those presented in section 5 were obtained in the summer of 1997 and appeared to be very promising. Then we decided to try to establish some mathematical justification for the algorithm and to find out why and under what conditions the algorithm works. This may eventually help us improve the algorithm. This has been soon proved to be a much more challenging task. Since we try to set a general framework and at the same time always keep some model problems in mind, every time a new model problem is solved numerically, new interesting mathematical questions can be asked. We try to answer the question in a more general way.

For the purpose of mathematical justification, let us adopt another approach. In studying a dynamic system, Nehari [25] introduced the concept of a solution submanifold $\mathcal{M}$, and proved that a global minimizer of the energy functional on $\mathcal{M}$ is a solution to the dynamic system with MI = 1. Ding–Ni used Nehari's idea to study the model problem with the zero Dirichlet BC and $\ell = 0$. They defined a solution submanifold

$$\mathcal{M} = \left\{ v \in H_0^1(\Omega) | v \neq 0, \int_\Omega \left[|\nabla v|^2 - v f(v) \right] \, dx = 0 \right\}. \tag{1.7}$$

Under the condition that $f'(t) > \frac{f(t)}{t}, t \neq 0$, Ding–Ni [27] proved that a global minimizer of the energy function J on $\mathcal{M}$ is a solution with MI = 1 to the model problem.

Our basic idea to design an algorithm for finding multiple saddle points consists of three main elements.

(1) First define a solution (stable) submanifold $\mathcal{M}$ s.t. a local minimum point of $J(u)$ on $\mathcal{M}$ yields a critical point. Thus the problem becomes a minimization of J on the submanifold $\mathcal{M}$, and a saddle point becomes stable on the submanifold $\mathcal{M}$. If a monotone decreasing search is used in the minimization process, the algorithm will be stable. At a point on $\mathcal{M}$, we can apply, e.g., a steepest descent search to approximate a local minimizer of J on $\mathcal{M}$.

(2) There must be a *return rule.* As a steepest descent search usually leaves the submanifold $\mathcal{M}$, for the algorithm to continue to iterate we need to design a return rule for the search to return to $\mathcal{M}$.

(3) There must be a strategy to *avoid degeneracy.* Since we are searching for a saddle point at a higher critical level, at least, for a new solution, a simple minimization may cause degeneracy to an (old) saddle point at a lower critical level. Thus a strategy to avoid degeneracy is crucial to guarantee that the new critical point found is different from the old ones. This strategy may also be incorporated in the definition of the solution submanifold. It can be seen as follows.

(a) Choi–McKenna's algorithm has a return rule and a strategy to avoid a degeneracy to the zero with MI = 0. We will provide a mathematical justification for their algorithm in section 2; see Theorem 2.1 with $L = \{0\}$ or Theorem 4.2.

(b) Ding–Costa–Chen's algorithm has a return rule and a strategy to avoid a degeneracy to a solution with MI = 1. We will modify their algorithm and then provide a mathematical justification; see Theorem 2.1 with $L = \{w_1\}$.

The organization of this paper is as follows. In section 2, we first prove an important technical lemma. Then we establish a local minimax characterization of a saddle point, which can be used to design a numerical minimax algorithm for finding multiple saddle points. An existence theorem is also proved in this section. Section 3 presents our numerical minimax algorithm in detail and gives some convergence related properties. In section 4, we apply our minimax method to solve a class of semilinear elliptic equations and present some related analysis. Finally, in section 5, we exhibit some numerical examples for multiple solutions and their graphics to illustrate the algorithm and theory.

We do not intend to prove any new existence theorem. Our objective is to develop numerical algorithms and corresponding mathematical theory for finding multiple critical points. It is understood that many critical points cannot be approximated. Only those with "nice properties" can be numerically approximated. We try to classify those "nice" saddle points through mathematical analysis. It is reasonable that the hypotheses in our local minimax characterization of saddle points in this paper are stronger than those in the existence theorems. Our hypotheses will be gradually localized and generalized as research advances. Methods to check these hypotheses will also be developed in subsequent papers.

2. Local min-max theorems. Let H be a Hilbert space with inner product $\langle , \rangle$ and norm $\| \ \|$, and let J be a real C^1-generic energy functional on H. For any subspace $H' \subset H$, denote $S_{H'} = \{v | v \in H', \|v\| = 1\}$ the unit sphere in H'. Let L be a closed subspace in H, called a base subspace, and let $H = L \oplus L^{\perp}$ be the orthogonal decomposition, where $L^{\perp}$ is the orthogonal complement of L in H. For each $v \in S_{L^{\perp}}$ let $[L, v] = \{tv + w | w \in L,\ t \geq 0\}$ be the closed half subspace. L can be either finite or infinite dimensional.

DEFINITION 2.1. *A set-valued mapping $P: S_{L^{\perp}} \to 2^H$ is called the* peak mapping *of J with respect to $H = L \oplus L^{\perp}$ if for any $v \in S_{L^{\perp}}$, $P(v)$ is the set of all local maximum points of J in $[L, v]$. A single-valued mapping $p: S_{L^{\perp}} \to H$ is a* peak selection *of J with respect to L if*

$$p(v) \in P(v) \quad \forall v \in S_{L^{\perp}}.$$

For a given $v \in S_{L^{\perp}}$, we say that J has a local peak selection with respect to L at v if there is a neighborhood $\mathcal{N}(v)$ of v and a function $p: \mathcal{N}(v) \cap S_{L^{\perp}} \to H$ s.t.

$$p(u) \in P(u) \quad \forall u \in \mathcal{N}(v) \cap S_{L^{\perp}}.$$

In a special case when $L = \{0\}$, the peak mapping $P(v)$ is the set of all local maximum points of J along the direction v, for any point $v \in S$.

Most minimax theorems in critical point theory used a (constrained) global maximization (on a compact set) at the first level. Thus a solution at the first level always exists. For algorithm implementation, existence is not enough; we want an approximation scheme to a solution. Therefore, we use unconstrained local maximization at the first level. Numerically this is great. However, it then raises three major problems in analysis. (a) For some $v \in S_{L^{\perp}}$, $P(v)$ may contain multiple local maxima in $[L, v]$. In particular, P may contain multiple branches, even U-turn or bifurcation points. (b) p may not be defined at some points in $S_{L^{\perp}}$. (c) The limit of a sequence of local maximum points may not be a local maximum point. Thus the analysis involved becomes much more complicated. Although, it is well known that the energy function (1.5) of our model problem goes to negative infinity uniformly in any finitely dimensional subspace (see [30]). Thus, in any finite-dimensional subspace, there must be at least one maximum point, i.e., $P(v) \neq \emptyset$. That is, problem (b) will not happen. Problem (a) is not concerned in the numerical works of Choi–McKenna and Ding–Costa–Chen. However, for more general settings, all three problems have to be resolved. As for (a), we use a selection p to choose one branch from the others. Numerically it is done by following a certain negative gradient flow and developing some consistent strategies to avoid jumps between different branches. For (b), we need only p to exist locally around a point v along a negative gradient flow. We are currently working on a min-orthogonal algorithm where $p(v)$ need only be a local suborthogonal point. This will further resolve problem (b). When P contains some U-turn or bifurcation points, if a saddle point happens to be a U-turn or bifurcation point of P, a local minimax search at either one of the branches connecting to the point will follow the negative gradient flow to the saddle point. If a saddle point is not a U-turn or bifurcation point of P, when a local minimax search is at the same branch of P as the saddle point, according to our local characterization results, it is not a problem. In the case when a local minimax search is at a different branch of P connecting to the U-turn or bifurcation point, we will develop a technique to allow the search to pass through a U-turn or bifurcation point. We will address this technique in a future article. As for problem (c), more analysis is required. One of the reasons for problem (c) to take place is that under the definition of a peak selection $p(v)$, the solution submanifold $\mathcal{M} = \{p(v) : v \in S_{L^\perp}\}$ is not closed. In a future paper, we will define a more general local (orthogonal) selection $p(v)$ with which the new solution submanifold is closed and contains the current solution submanifold as a subset. Then we prove that a local minimum point of J on the new solution submanifold also yields a saddle point, and the implicit function theorem can be used to check if p is continuous (differentiable) at a given point v, a condition required by all the results we proved in this paper. This study leads to a completely different approach and will be addressed in a future paper.

The following technical lemma plays a crucial role in this paper. It describes the relation between the gradient of J and the variation of a peak selection.

LEMMA 2.1. *For $v_\delta \in S_{L^\perp}$, if there is a local peak selection p of J with respect to L at v_δ s.t.* (i) *p is continuous at v_δ,* (ii) *$d(p(v_\delta), L) > \alpha > 0$, and* (iii) *$\|\nabla J(p(v_\delta))\| > \delta > 0$, then there exists $\varepsilon > 0$ s.t.*

$$J(p(v(s))) - J(p(v_\delta)) < -\alpha\delta\|v(s) - v_\delta\| \quad \forall 0 < s < \varepsilon$$

and

$$v(s) = \frac{v_\delta + sw}{\|v_\delta + sw\|}, \quad w = -\frac{\nabla J(p(v_\delta))}{\|\nabla J(p(v_\delta))\|}.$$

Proof. Let $\mathcal{N}(v_\delta)$ be a neighborhood of v_δ for the local peak selection p. By (iii),

$$\langle \nabla J(p(v_\delta)), w\rangle < -\delta.$$

Since $p(v_\delta) \in [L, v_\delta]$, $p(v_\delta) = x_\delta + t_\delta v_\delta$, where $x_\delta \in L$ and $t_\delta \in \mathbb{R}$, we have $t_\delta > \alpha$. $p(v_\delta)$ is a local maximum point of $J(p(x))$ on $[L, v_\delta]$. So it is clear that $w \in L^\perp$ and $w \perp v_\delta$. Due to the continuity, there exist positive numbers ε_1, ε_2, t_1, and t_2 with $0 < t_1 < t_\delta < t_2$ and open balls $B_{\varepsilon_1,L} = \{x | x \in L,\ \|x - x_\delta\| < \varepsilon_1\}$ and $B_{\varepsilon_2,L^\perp} = \{x | x \in L^\perp,\ \|x - t_\delta v_\delta\| < \varepsilon_2\}$ s.t.

(a) $p(v_\delta)$ is a maximum point of J on $[t_1, t_2] \times v_\delta + B_{\varepsilon_1,L}$,

(b) $\langle \nabla J(x_1 + x_2), w\rangle < -\delta$ for all $x_1 \in B_{\varepsilon_1,L}, x_2 \in B_{\varepsilon_2,L^\perp}$,

(c) $tv_\delta + sw \in B_{\varepsilon_2,L^\perp}$ for any $t \in [t_1, t_2]$ and s, $0 < s < \frac{\varepsilon_2}{2}$.

For any $x_1 \in B_{\varepsilon_1,L}$, and $t \in [t_1, t_2]$, by the mean value theorem,

$$J(tv_\delta + x_1 + sw) - J(tv_\delta + x_1) = \langle \nabla J(tv_\delta + x_1 + \xi w), sw\rangle < -s\delta. \tag{2.1}$$

By (a), $J(tv_\delta + x_1) \le J(p(v_\delta))$, so we have $J(tv_\delta + x_1 + sw) - J(p(v_\delta)) < -s\delta$. Since $w \perp v_\delta$,

$$\lim_{s\to 0^+} \left\| \frac{1}{s}\left(\frac{t_\delta v_\delta + sw}{\|t_\delta v_\delta + sw\|} - v_\delta\right)\right\| = \frac{1}{t_\delta} < \frac{1}{\alpha}. \tag{2.2}$$

Thus, without loss of generality, we can assume that for $s > 0$ small

$$\left\| \frac{t_\delta v_\delta + sw}{\|t_\delta v_\delta + sw\|} - v_\delta \right\| < \frac{s}{\alpha}. \tag{2.3}$$

By our notation

$$v\left(\frac{s}{t_\delta}\right) = \frac{t_\delta v_\delta + sw}{\|t_\delta v_\delta + sw\|},$$

so we have, for sufficient small s,

$$\alpha \| v\left(\frac{s}{t_\delta}\right) - v_\delta \| < s. \tag{2.4}$$

Combine (2.1) and (2.4), and we can find $\varepsilon_3 > 0$ s.t.

$$J(tv_\delta + x_1 + sw) - J(p(v_\delta)) < -\alpha\delta \| v\left(\frac{s}{t_\delta}\right) - v_\delta \| \quad \forall t \in [t_1, t_2],\ x_1 \in B_{\varepsilon_1, L},\ 0 < s < \varepsilon_3.$$

Denote $D = \{tv_\delta + x_1 + sw | s < \varepsilon_3,\ t \in (t_1, t_2),\ x_1 \in B_{\varepsilon_1, L}\}$. Then D is an open neighborhood of $p(v_\delta) = t_\delta v_\delta + x_\delta$ in the subspace spanned by L, v_δ, and w. By the continuity of p at v_δ, there exists positive ε s.t. $v(\frac{s}{t_\delta}) \in S_{L^\perp} \cap \mathcal{N}(v_\delta)$ and $p(v(\frac{s}{t_\delta})) \in D$ for all $\frac{s}{t_\delta} < \varepsilon$. Besides, since $p(v(\frac{s}{t_\delta})) \in [L, v(\frac{s}{t_\delta})]$, $p(v(\frac{s}{t_\delta})) = ct_\delta v_\delta + x_1 + csw$ uniquely for some constant number c and $x_1 \in B_{\varepsilon_1, L}$ with $t_1 < ct_\delta < t_2$ and $cs < \varepsilon_3$. Thus

$$\begin{aligned} J\left(p\left(v\left(\frac{s}{t_\delta}\right)\right)\right) - J(p(v_\delta)) &= J(ct_\delta v_\delta + x_1 + csw) - J(p(v_\delta)) \\ &< -\alpha\delta \| v\left(\frac{cs}{ct_\delta}\right) - v_\delta \| = -\alpha\delta \| v\left(\frac{s}{t_\delta}\right) - v_\delta \|. \qquad \square \end{aligned}$$

The following theorem characterizes a saddle point as a local minimax solution and serves as a mathematical justification for our algorithm to be proposed.

THEOREM 2.1. *Let $v_0 \in S_{L^\perp}$. If J has a local peak selection p with respect to L at v_0 s.t.* (i) *p is continuous at v_0,* (ii) $d(p(v_0), L) > 0$, *and* (iii) *v_0 is a local minimum point of $J(p(v))$ on $S_{L^\perp}$, then $p(v_0)$ is a critical point of J.*

Proof. Suppose that $p(v_0)$ is not a critical point of J. By Lemma 2.1, for a sufficient small positive s, set

$$w = -\frac{\nabla J(p(v_0))}{\|\nabla J(p(v_0))\|}, \quad v(s) = \frac{v_\delta + sw}{\|v_\delta + sw\|}, \tag{2.5}$$

and we have

$$J(p(v(s))) - J(p(v_0)) \leq -\frac{1}{2} \|\nabla J(p(v_0))\| \|v(s) - v_0\| d(p(v_0), L) < 0,$$

which contradicts the fact that v_0 is a local minimum point of $J(p(v))$ on $S_{L^\perp}$. Thus $p(v_0)$ must be a critical point of J. $\square$

Remark 2.1. Condition (i) is required for a numerical algorithm to be stable and convergent. Condition (ii) is a separation condition posed by almost all minimax theorems to have a new critical point. Condition (iii) replaces a global minimization, used in most minimax theorems in the literature, by a local minimization. Due to the local nature of the above characterization and (2.5), it is clear that $S_{L^\perp}$ in condition (iii) can be localized to be any subset containing $v(s)$ for small $s \geq 0$.

To establish an existence result, the following Palais–Smale (PS) condition is used to replace the usual compactness condition.

Definition 2.2. *A function $J \in C^1(H)$ is said to satisfy the PS condition if any sequence $\{u_n\} \in H$ with $J(u_n)$ bounded and $J'(u_n) \to 0$ has a convergent subsequence.*

Theorem 2.2. *Let J be C^1 and satisfy the PS condition. If there is a peak selection p of J with respect to L s.t.* (i) *p is continuous,* (ii) *$d(p(v), L) \geq \alpha$ for some positive α, and for any $v \in S_{L^\perp}$,* (iii) $\inf_{v \in S_{L^\perp}} J(p(v)) > -\infty$, *then there exists $v_0 \in S_{L^\perp}$ s.t. $p(v_0)$ is a critical point of J, and*

$$J(p(v_0)) = \min_{v \in S_{L^\perp}} J(p(v)).$$

Proof. Since $S_{L^\perp}$ is a closed metrical subspace and $J(p(v))$ is a continuous function on $S_{L^\perp}$, bounded from below, by Ekeland's variational principle, for any integer n, there exists $v_n \in S_{L^\perp}$ s.t.

$$J(p(v_n)) \leq \inf_{v \in S_{L^\perp}} J(p(v)) + \frac{1}{n}, \tag{2.6}$$

and for any $v \in S_{L^\perp}, v \neq v_n$,

$$J(p(v)) - J(p(v_n)) \geq -\frac{1}{n}\|v - v_n\|.$$

From Lemma 2.1, for some v in $S_{L^\perp}$ and close to v_n,

$$J(p(v)) - J(p(v_n)) \leq -\frac{1}{2}\|v - v_n\| \|\nabla J(p(v_n))\| d(p(v_n), L).$$

Thus

$$\|\nabla J(p(v_n))\| \leq \frac{2}{n d(p(v_n), L)} < \frac{2}{n\alpha}. \tag{2.7}$$

By the PS condition, $\{p(v_n)\}$ has a subsequence, denoted again by $\{p(v_n)\}$, converging to some point $u_0 \in H$. Note that $p(v_n) = t_n v_n + x_n$ for some scalar $t_n > 0$, $v_n \in S_{L^\perp}$, and $x_n \in L$. It follows from $\|p(v_n) - p(v_m)\|^2 = \|t_n v_n - t_m v_m\|^2 + \|x_n - x_m\|^2$ that $\{t_n v_n\}$ is a Cauchy sequence as well. Since $\|v_n\| = 1$, $\|t_n v_n\| = t_n \to t_0$. By our assumption (ii), $t_0 \geq \alpha > 0$. Thus $v_n \to v_0 \in S_{L^\perp}$. By the continuity, we have $u_0 = p(v_0)$. Then by (2.7), $p(v_0)$ is a critical point of J, and, moreover, (2.6) leads to

$$J(p(v_0)) = \min_{v \in S_{L^\perp}} J(p(v)). \qquad \square$$

The following corollary is a special case of Theorem 2.2 with $L = \{0\}$, which can be viewed as a mathematical justification of the modified Choi–McKenna algorithm to find a solution with MI = 1.

Corollary 2.1. *Let $J \in C^1(H)$ and satisfy the PS condition. Let S be the unit sphere of H and $p(v)$ be a local maximum point of J on $\{tv | t \in (0, +\infty)\}$ for each $v \in S$ s.t.* (i) *p is continuous,* (ii) *$\|p(v)\| \geq \alpha$ for some $\alpha > 0$, and for any $v \in S$,* (iii) $\inf_{v \in S} J(p(v)) > -\infty$; *then there exists $v_0 \in S$ s.t. $p(v_0)$ is a critical point of J, and, moreover,*

$$J(p(v_0)) = \min_{v \in S} J(p(v)).$$

Proof. We can simply apply Theorem 2.2 with $L = \{0\}$ to draw the conclusion. □

Remark 2.2. To apply Ekeland's variational principle, a function must be bounded from below. In general, J is not bounded from below. However, $J(p(v))$ has a much better chance to be bounded from below. Therefore, we use condition (iii) in Theorem 2.2. This condition is satisfied automatically by our model problem; see section 4.

3. A minimax algorithm for finding critical points. If we define a solution submanifold by

$$\mathcal{M} = \{p(v) : v \in S_{L^\perp}\},$$

then by Theorem 2.1 a local minimum point $p(v_0)$ of J on $\mathcal{M}$ is a critical point of J. Numerically, a local minimum point can be approximated by the steepest descent search. When the steepest descent search leaves $\mathcal{M}$ to a point v, a local maximization on $[L, v]$ yields $p(v)$, a point returned to $\mathcal{M}$. Thus our algorithm can continue to iterate. Since $J'(p(v(s))) \perp L$, the search will not degenerate to L, which contains previously found solutions.

```
The Flow Chart of a Minimax Algorithm
Step 1. Let $n-1$ critical points $w_1, w_2, \ldots, w_{n-1}$ of $J$ be previously found
        and $w_{n-1}$ have the highest critical value. Set the base space
        $L = \text{span}\{w_1, w_2, \ldots, w_{n-1}\}$. Let $v^0 \in L^\perp$ be an ascent direction at $w_{n-1}$ and
        $\bar{\alpha}_k > 0$ be given;
Step 2. For $k = 0$, solve
            $w^k \equiv p(v^0) \equiv t_0^* v^0 + v_L^* = \arg \max_{u \in [L, v^0]} J(u);$
Step 3. Compute the steepest descent direction $d^k$ of $J$ at $w^k$;
Step 4. if $\|d^k\| \leq \varepsilon$, then output $w_n \equiv w^k$ and stop, else goto Step 5;
Step 5. For each $0 < \alpha \leq \bar{\alpha}_k$, denote $v^k(\alpha) = \frac{v^k - \alpha * d^k}{\|v^k - \alpha * d^k\|}$ and with the initial guess
        $u = t_0^* v^k(\alpha) + v_L^*$, solve
            $p(v^k(\alpha)) = \arg \max_{u \in [L, v^k(\alpha)]} J(u);$
        then solve
            $w^{k+1} \equiv p(v^{k+1}) \equiv t_0^* v^k(\alpha^*) + v_L^* = \arg \min_{0 < \alpha \leq \bar{\alpha}_k} J(p(v^k(\alpha)));$
Step 6. Update $k = k + 1$ and goto step 3.
```

Remark 3.1. Let us make some remarks on each step in the above algorithm.

In step 1, when we start from some known critical points, if the critical point w_0 with $\text{MI} = 0$ is not zero, we should add w_0 to the base space L. We first start from w_0 to find w_1 and so on. As for our model problem, $w_0 = 0$, so $L = \{0\}$, and according to the Morse theory, to find w_1, we may use any direction $v^0 \in H$ as an ascent direction of J at w_0.

Now assume $w_1, w_2, \ldots, w_{n-1}$ have been found this way and w_{n-1} is the one with the highest critical value. The base space L is spanned by $w_1, w_2, \ldots, w_{n-1}$. We start the algorithm with an ascent direction v^0 of J at w_{n-1}.

It is clear that the separation condition is quite reasonable to ensure that the critical point found is not in the base space L spanned by the old critical points and therefore will be a new one at a higher critical level.

The idea to choose a direction $v^0 \in L^\perp$ as an ascent direction is quite useful in practice. It makes the separation condition easier to satisfy. For example, in section 4, to choose an ascent direction, we always use a functional in $H_0^1(\Omega)$ whose peak is away from the peaks of the known critical points $w_1, w_2, \ldots, w_{n-1}$. This idea works very well.

In step 2, we have a simple unconstrained local maximization problem.

In step 3, to find the steepest descent direction d^k of J at a point w^k is usually equivalent to solving a linear system. As for the model problem, a direct calculation shows that the steepest descent direction d^k of J at w^k can be obtained by solving the linear elliptic BVP

$$(3.1) \qquad \begin{cases} \Delta d^k(x) - \ell d^k(x) = \Delta w^k(x) - \ell w^k(x) + f(x, w^k(x)), \quad x \in \Omega, \\ d^k(x) = 0, \quad x \in \partial\Omega \qquad\qquad \text{for the zero Dirichlet BC}, \\ \dfrac{\partial d^k(x)}{\partial n} = \dfrac{\partial w^k(x)}{\partial n}, \quad x \in \partial\Omega \qquad \text{for the zero Neumann BC}, \end{cases}$$

which can be solved numerically by various finite element, finite difference, or boundary element solvers. Since one important topic in nonlinear analysis is to study how a variation of the domain affects the profile of a solution [14] and the boundary element method can easily handle a complex domain or trace a variation of the domain, we use a boundary element method. Note that for the zero Neumann BC problem, if we choose an initial ascent direction v^0 with $\frac{\partial v^0}{\partial n}(x)|_{\partial\Omega} = 0$, then in every iterate we have $\frac{\partial d^k}{\partial n}(x)|_{\partial\Omega} = 0$ in (3.1).

In step 4, for our model problem, we can use a norm $\|\Delta d^k - l \cdot d^k\|_{L^2} < \varepsilon$ to control the error. Note that this is an absolute error indicator for our model problem, since

$$\Delta d^k(x) - l \cdot d^k(x) = \Delta w^k(x) - \ell w^k(x) + f(x, w^k(x)).$$

In step 5, for each point $v^k(\alpha) = v^k - \alpha \cdot d^k$ in the steepest descent direction, we can find a local maximum $p(v^k(\alpha))$ of J in $[L, v^k(\alpha)]$. We then try to find the point in the steepest descent direction with the smallest such maximum. We must follow a consistent way to find a local maximum point of J so that $p(v)$ depends on v continuously and is kept away from L. Thus we specify the initial guess $u = t_0^* v^k(\alpha) + v_L^*$ in searching for a local maximum in $[L, v^k(\alpha)]$. This initial guess closely and consistently traces the position of the previous point $w^k = t_0^* v^k + v_L^*$. This strategy is also to avoid the algorithm from possible oscillating between different branches of the peak mapping P.

The number $\bar{\alpha}$ is to enhance the stability of the algorithm. It controls the stepsize of the search along the steepest descent direction to avoid the search going too far, i.e., to leave the solution (stable) submanifold $\mathcal{M}$ too far to lose stability of the algorithm.

The following theorem implies that the local minimax algorithm is strictly descending and therefore a stable algorithm.

THEOREM 3.1. *If $d^k = \nabla J(w^k) \neq 0$, $w^k \notin L$, and p is continuous at v^k, then*

$$J(w^{k+1}) < J(w^k).$$

If p is continuous and $d(p(v), L) > \alpha > 0$ for all $v \in S_{L^\perp}$, then there exist $\bar{\alpha}_k > 0$ and $d > 0$ s.t.

$$J(w^{k+1}) - J(w^k) < -d\|\nabla J(w^k)\|\|v^{k+1} - v^k\| \quad \forall k = 1, 2, \ldots, \tag{3.2}$$

where $w^k = p(v^k)$ and $w^{k+1} = p(v^{k+1})$ are determined in step 5 of the algorithm.

Proof. We have only to prove (3.2). Using the notation in the algorithm scheme, write

$$v^{k+1} = v^k(\alpha_k^*) = \frac{v^k - \alpha_k^* d^k}{\|v^k - \alpha_k^* d^k\|}.$$

Take $\delta = \|d^k\|/2$; by Lemma 2.1, we can find $\bar{\alpha}_k > 0$, s.t. if $\bar{\alpha}_k > s > 0$, then

$$J(p(v^k(s))) - J(p(v^k)) < -\alpha\delta\|v^k(s) - v^k\|.$$

In particular, we choose such $\bar{\alpha}_k$ in the algorithm, and $0 < \alpha_k^* \leq \alpha_k$ is satisfied; with $d = 2\alpha$, we have

$$J(p(v^k(\alpha_k^*))) - J(p(v^k)) < -d\|d^k\|\|v^k(\alpha) - v^k\|,$$

i.e., (3.2). □

Convergence is always a paramount issue for any numerical algorithm. Due to multiplicity, degeneracy, and instability of saddle points, general convergence analysis will be very difficult. More profound analysis is required. We will establish some convergence results of the algorithm in a subsequent paper [20]. Instead, in section 4, we present some applications of our minimax theorem and method to a class of semilinear elliptic PDEs.

4. Application to semilinear elliptic PDEs.

In this section, we apply our local minimax method to study the model problem. We use the notations as in [30] with a slight change. Ω is a smooth bounded domain in $\mathbb{R}^n$. Consider a semilinear elliptic Dirichlet BVP

$$\begin{cases} \Delta u(x) + f(x, u(x)) &= 0, \qquad x \in \Omega, \\ u(x) &= 0, \qquad x \in \partial\Omega, \end{cases} \tag{4.1}$$

where the function $f(x, \xi)$ satisfies the following standard hypotheses.

(h1) $f(x, \xi)$ is locally Lipschitz on $\bar{\Omega} \times \mathbb{R}$.

(h2) There are positive constants a_1 and a_2 s.t.

$$|f(x, \xi)| \leq a_1 + a_2|\xi|^s, \tag{4.2}$$

where $0 \leq s < \frac{n+2}{n-2}$ for $n > 2$. If $n = 2$,

$$|f(x, \xi)| \leq a_1 \exp\phi(\xi), \tag{4.3}$$

where $\phi(\xi)\xi^{-2} \to 0$ as $|\xi| \to \infty$.

(h3) $f(x, \xi) = o(|\xi|)$ as $\xi \to 0$.

(h4) There are constants $\mu > 2$ and $r \geq 0$ s.t. for $|\xi| \geq r$,

$$0 < \mu F(x, \xi) \leq \xi f(x, \xi), \tag{4.4}$$

where $F(x, \xi) = \int_0^\xi f(x, t)dt$.

In our later numerical computation, we solve problems in $\mathbb{R}^2$, where (h2) is not a substantial restriction. (h4) says that f is superlinear, which implies that there exist positive numbers a_3 and a_4 s.t. for all $x \in \bar{\Omega}$ and $\xi \in \mathbb{R}$

$$F(x, \xi) \geq a_3|\xi|^\mu - a_4. \tag{4.5}$$

The variational functional associated to the Dirichlet problem (4.1) is

$$J(u) = \frac{1}{2}\int_\Omega |\nabla u(x)|^2 dx - \int_\Omega F(x, u(x))dx, \quad u \in H \equiv H_0^1(\Omega), \tag{4.6}$$

where we use an equivalent norm $\|u\| = \int_\Omega |\nabla u(x)|^2 dx$ for the Sobolev space $H = H_0^1(\Omega)$.

It is well known [30] that under conditions (h1)–(h4), J is C^1 and satisfies the PS condition. A critical point of J is a weak solution and also a classical solution of (4.1). 0 is a local minimum point (MI = 0) of J. Moreover, in any finitely dimensional subspace of H, J goes to negative infinity uniformly. Therefore, for any finite dimensional subspace L, the peak mapping P of J with respect to L is nonempty. We need one more hypothesis:

(h5) $\frac{f(x,\xi)}{|\xi|}$ is increasing with respect to ξ, or

(h5′) $f(x,\xi)$ is C^1 with respect to ξ and $f_\xi(x,\xi) - \frac{f(x,\xi)}{\xi} > 0$.

It is clear that (h5′) implies (h5). If $f(x,\xi)$ is C^1 in ξ, then (h5) and (h5′) are equivalent. All the power functions of the form $f(x,\xi) = |\xi|^k\xi$ with $k > 0$ satisfy (h1) through (h5′), and so do all the positive linear combinations of such functions. Under (h5) or (h5′), J has only one local maximum point in any direction, or, the peak mapping P of J with respect to $L = \{0\}$ has only one selection. In other words, $P = p$. The proof can be found in [27] and [14].

Lemma 4.1. *Under* (h5) *or* (h5′)*, for any* $u \in H$*, the function* $g(t) = J(tu)$*,* $t \geq 0$*, has a unique local and so a global maximum point.*

Let $L = \{0\}$ and $\mathcal{M} = \{p(v) | v \in S_{L^\perp} = S_H\}$, where $p(v)$ is the unique peak selection of J with respect to L. By the above lemma, it can be easily checked that $\mathcal{M}$ is exactly the solution submanifold (1.7) defined by Ding and Ni. Our definition displays the essence of why such a solution submanifold works. Our definition is also given in a more general way. It also works for finding a critical point as a local minimax solution with a higher MI. Actually, under (h5), the peak selection is not only unique but also continuous. In other words, $\mathcal{M}$ is a topological manifold. The following theorem is given in a more general form, which mainly states that uniqueness implies continuity.

Theorem 4.1. *Under the hypotheses* (h1)–(h5)*, if the peak mapping* P *of* J *with respect to a finitely dimensional subspace* L *is singleton at* $v_0 \in S_{L^\perp}$ *and for any* $v \in S_{L^\perp}$ *around* v_0*, a peak selection* $p(v)$ *is a global maximum point of* J *in* $[L, v]$*, then* p *is continuous at* v_0*.*

Proof. (h4) implies (4.5); namely, $F(x,\xi) \geq a_3|\xi|^\mu - a_4$, where a_3 and a_4 are positive constants depending on F and Ω. It is known that $\int_\Omega |u(x)|^\mu dx$ is a positive continuous functional in H and $S_H \cap [L, v_0]$ is compact; thus we can write

$$\alpha_0 \equiv \min_{v \in S_H \cap [L, v_0]} \int_\Omega |v(x)|^\mu dx > 0.$$

Let $\alpha = \frac{\alpha_0}{2}$. For each $v \in S_H \cap [L, v_0]$ there is a neighborhood $\mathcal{N}(v)$ of v s.t.

$$\int_\Omega |u(x)|^\mu dx > \alpha \quad \forall u \in \mathcal{N}(v).$$

Since

$$\Big(S_H \cap [L, v_0]\Big) \subset \bigcup_{v \in S_H \cap [L, v_0]} \mathcal{N}(v)$$

and $S_H \cap [L, v_0]$ is compact, there exist $v_1, \ldots, v_n \in S_H \cap [L, v_0]$ s.t.

$$\Big(S_H \cap [L, v_0]\Big) \subset \bigcup_{i=1}^{n} \mathcal{N}(v_i) \quad \text{and} \quad \int_\Omega |u(x)|^\mu dx > \alpha \quad \forall u \in \bigcup_{i=1}^{n} \mathcal{N}(v_i).$$

Note that for each $v \in S_{L^\perp}$ and $w \in S_H \cap [L, v]$, we can write $w = w_l + t_w v$ with $w_L \in L$ and $|t_w| \leq 1$. Then $w_0 = w_l + t_w v_0 \in S_H \cap [L, v_0]$ and $\|w - w_0\|^2 = t_w^2 \|v - v_0\|^2 \leq \|v - v_0\|^2$. Thus we can find a neighborhood D of v_0 s.t.

$$\Big(S_H \cap [L, v]\Big) \subset \bigcup_{i=1}^{n} \mathcal{N}(v_i) \quad \forall v \in D \cap S_{L^\perp}.$$

Therefore, we have

$$\int_\Omega |w|^\mu dx \geq \alpha \quad \forall v \in D \cap S_{L^\perp},\ w \in S_H \cap [L, v]. \tag{4.7}$$

For any $v \in D \cap S_{L^\perp}$, $p(v)$ is a maximum point of J in $[L, v]$. In particular, $p(v)$ is a maximum point of J on the half line $\{tp(v)|\, t \in \mathbb{R}^+\}$. Set $w = \frac{p(v)}{\|p(v)\|}$ and define

$$g(t) = J(tw) = \frac{1}{2} t^2 \int_\Omega |\nabla w(x)|^2 \, dx - \int_\Omega F(x, tw(x)) \, dx.$$

Then g is positive and increasing near 0 and goes to $-\infty$ as $t \to \infty$. A local maximum is solved from

$$0 = \frac{dg}{dt}|_{t=t_1} = t_1 \int_\Omega |\nabla w(x)|^2 \, dx, - \int_\Omega w(x) f(x, t_1 w(x)) \, dx$$

or

$$0 = \|p(v)\| \int_\Omega |\nabla w|^2 dx - \int_\Omega w f(x, \|p(v)\| w) dx. \tag{4.8}$$

Since $\|w\| = 1$, by using (4.4) and (4.7), we have

$$\begin{aligned} 1 &= \frac{1}{\|p(v)\|^2} \int_\Omega \|p(v)\| w f(x, \|p(v)\| w) dx \geq \frac{1}{\|p(v)\|^2} \int_\Omega \mu F(x, \|p(v)\| w) dx \\ &\geq \frac{1}{\|p(v)\|^2} \int_\Omega \mu (a_3 \|p(v)\|^\mu |w|^\mu - a_4) dx \\ &= a_3 \mu \|p(v)\|^{\mu - 2} \int_\Omega |w|^\mu dx - \frac{1}{\|p(v)\|^2} \int_\Omega \mu a_4 dx \\ &\geq \mu a_3 \|p(v)\|^{\mu - 2} \alpha - \frac{1}{\|p(v)\|^2} \int_\Omega \mu a_4 dx. \end{aligned}$$

Note that $\mu > 2$, the right-hand side of the last inequality goes to ∞ if $\|p(v)\| \to \infty$, and this violates the above inequalities. Therefore, there must exist $\beta > 0$ s.t. $\|p(v)\| \leq \beta$.

Now let $\{v_n\} \subset D \cap S_{L^\perp}$ be any sequence s.t. $v_n \to v_0$. Denote $p(v_n) = t_n v_n + x_n$, where $x_n \in L$. Since $\|p(v_n)\| \leq \beta$ and $v_n \perp x_n$, we have $t_n \leq \beta$ and $\|x_n\| \leq \beta$. Therefore, we can find a subsequence $\{v_{n_k}\}$ s.t. t_{n_k} and x_{n_k} converge, respectively, to t_0 and x_0. In other words, $p(v_{n_k})$ goes to $t_0 v_0 + x_0$, which lies in $[L, v_0]$. Since we assume that $p(v_{n_k})$ is a global maximum point of J in $[L, v_{n_k}]$ for each n_k, the limit point $t_0v_0 + x_0$ must be a maximum point of J in $[L, v_0]$ as well. But by the assumption, the peak mapping P of J is singleton at v_0, so $t_0v_0 + x_0 = p(v_0)$. Since $\{v_n\}$ is arbitrary, by the above argument, p is continuous at v_0. □

As an immediate conclusion of Theorem 4.1, we have the following continuous result (see [36, Lemma 4.1]).

COROLLARY 4.1. *Under the hypotheses* (h1)–(h5)*, the only peak selection p of J with respect to $L = \{0\}$ is continuous.*

Proof. By Lemma 4.1, there is only one peak selection p of J with respect to $L = \{0\}$. In any direction v, the function $g(t) = J(tu)$ possesses only one local maximum point and therefore a global maximum point of J over the subset $\{tv|t \in \mathbb{R}^+\}$. Thus from Theorem 4.1, it is continuous at any point. □

Moreover, the unique selection p of peak mapping with respect to $L = \{0\}$ satisfies all the requirements of Theorems 2.1 and 2.2. Thus we can apply Theorems 2.1 and 2.2 to establish the following existence result.

THEOREM 4.2. *Under the hypotheses of* (h1)–(h5)*, there exists at least one solution to*

$$\text{local} \min_{x \in \mathcal{M}} J(x), \tag{4.9}$$

and any such solution is a critical point of J and therefore a solution to problem (4.1)*.*

Proof. $\mathcal{M}$ is the image of the unique peak selection p of J with respect to $L = \{0\}$. By Corollary 4.1, p is continuous. Under the conditions of (h2)–(h4), we know that (see [30])

$$J(u) = \frac{1}{2}\|u\|^2 + o(\|u\|^2)$$

as $u \to 0$. Thus we can find $\delta > 0$ s.t.$\|p(v)\| > \delta$ for any direction $v \in H$. This is exactly the separation condition in Theorems 2.1 and 2.2. Obviously, $J(p(v)) > 0$ for each direction $v \in H$, and thus is bounded from below. Therefore, all the conditions in Theorems 2.1 and 2.2 are satisfied. Theorem 2.2 states that there is at least one critical point as a minimax solution, and Theorem 2.1 confirms that any local minimax solution is a critical point. □

By a similar argument as in the proof of the above theorem and taking Lemma 4.1 into account, we can show that for BVP (4.1), for any closed subspace L and any peak selection p of J with respect to L, we have $\inf_{v \in S_{L^\perp}} J(p(v)) > \delta > 0$.

It is known that for BVP (4.1), solution prefers open space. When the domain has multiple compartments connected by narrow corridors, such as a dumbbell-shaped domain in section 5 for our computational examples, multiple solutions do exist as local minimax solutions. The one with the smallest energy is the global minimax solution, i.e., the ground state.

The following results indicate that under conditions (h1)–(h5′), for $L = \{0\}$, the minimax algorithm is actually a minimization on a differentiable submanifold $\mathcal{M}$.

THEOREM 4.3. *Assume that conditions* (h1)–(h5′) *are satisfied and that there exist $a_5 > 0$ and $a_6 > 0$ s.t. for s as specified in* (h2)*,*

$$|f_\xi(x, \xi)| \le a_5 + a_6|\xi|^{s-1}. \tag{4.10}$$

Then the only peak selection p of J with respect to $L = \{0\}$ is C^1.

Proof. Set $G: S_H \times \mathbb{R}^+ \to \mathbb{R}$, $G(v,t) = t - \int_\Omega v(x)f(x,tv(x))dx$. Thus under (4.10), G is C^1. Denote $\mathcal{M} = \{p(v)|v \in S_H\}$, where $p(v)$ is the only peak selection of J with respect to $L = \{0\}$, i.e., $p(v)$ is the maximum point of J on $\{tv|t > 0\}$. As in the proof of Theorem 4.1, for any v, if $p(v) = tv$, then we have

$$0 = t - \int_\Omega v(x)f(x,tv(x))dx.$$

Thus $\mathcal{M}$ is essentially the inverse image of G at 0, i.e., $\mathcal{M} = G^{-1}(0)$, and $\|p(v)\|$, as a positive number, is the solution of $t(v)$ to the equation $G(v,t(v)) = 0$. On the other hand,

$$\frac{\partial G}{\partial t} = 1 - \int_\Omega v^2(x) f_\xi(x,tv(x))\,dx. \tag{4.11}$$

For each v_0, at each pair (v_0,t_0) with $G(v_0,t_0) = 0$,

$$0 = 1 - \int_\Omega v_0^2(x)\frac{f(x,t_0v_0(x))}{t_0v_0(x)}\,dx > 1 - \int_\Omega v_0^2 f_\xi(x,t_0v_0(x))\,dx \quad \text{(by (h5}'\text{))},$$

i.e., $\frac{\partial G}{\partial t} < 0$ at (v_0,t_0), provided $G(v_0,t_0) = 0$. By the implicit function theorem, the solution $t(v)$ to the equation $G(v,t(v)) = 0$ exists uniquely in a neighborhood of each v_0 and is C^1 in v. Therefore, $\|p(v)\|$ is a C^1 function in v, and so is $p(v)$. □

Actually, in the proof, we can see that the solution submanifold, $\mathcal{M}$, is a differentiable manifold because Lemma 2.1 implies

$$\|\nabla J|_{\mathcal{M}}\| \geq \delta\|\nabla J\| \quad \text{for some } \delta > 0.$$

Example 4.1. Let us consider the BVP on a smooth bounded domain $\Omega \subset \mathbb{R}^n$ for $p > 2$:

$$\begin{cases} \Delta u(x) + |u(x)|^{p-2}u(x) = 0, & x \in \Omega, \\ u(x) = 0, & x \in \partial\Omega. \end{cases} \tag{4.12}$$

The associated variational functional is

$$J(u) = \frac{1}{2}\int_\Omega |\nabla u|^2\,dx - \frac{1}{p}\int_\Omega |u(x)|^p\,dx, \quad u \in H = H_0^1(\Omega).$$

For each $v \in S$, let $u = tv, t > 0$; then

$$J(tv) = \frac{t^2}{2}\int_\Omega |\nabla v|^2\,dx - \frac{t^p}{p}\int_\Omega |v(x)|^p\,dx = \frac{t^2}{2} - \frac{t^p}{p}\int_\Omega |v(x)|^p\,dx.$$

Thus

$$0 = \frac{\partial}{\partial t}J(tv) = t - t^{p-1}\int_\Omega |v(x)|^p\,dx$$

leads to

$$t_v = \left[\frac{1}{\int_\Omega |v(x)|^p\,dx}\right]^{\frac{1}{p-2}} > 0.$$

The peak selection p of J with respect to $L = \{0\}$ is

$$p(v) = t_v v = \left[\frac{1}{\int_\Omega |v(x)|^p \, dx} \right]^{\frac{1}{p-2}} v \quad \forall v \in S,$$

a continuously differentiable function, and the solution manifold

$$\mathcal{M} = \left\{ \left[\frac{1}{\int_\Omega |v(x)|^p \, dx} \right]^{\frac{1}{p-2}} v : v \in S \right\}$$

is a differentiable manifold.

5. Computational examples. We have applied our numerical algorithm to solve many semilinear BVPs with zero Dirichlet BC on various domains, such as the Lane–Emden equation, the Henon equation, and the Chandrasekhar equation on a disk, a rectangle, a concentric annulus, a nonconcentric annulus, dumbbell-shaped domains, and dumbbell-shaped domains with cavities. Here we present the computational results for the Lane–Emden equation

$$\begin{cases} \Delta u(x) + u^3(x) = 0, & x \in \Omega, \\ u(x) = 0, & x \in \partial\Omega, \end{cases} \tag{5.1}$$

where the domain Ω is, respectively, a dumbbell-shaped domain, a dumbbell-shaped domain with cavities (nonsymmetric), and a concentric annulus (highly degenerate). Here the solution $u(x)$ represents the density, so we are interested only in positive solutions. In all examples, we use a *cos* function to create a "mound-shaped" function as an initial ascent direction v_0 and a norm $\|\Delta \hat{u}\|_{L^2} = \|\Delta u + u^3\|_{L^2} < \varepsilon$ to control the error and terminate the iterate. Some solutions with MI = 1 have been computed elsewhere; see, e.g., [12, 17, 11] and references therein. It is to the best of our knowledge that those solutions with higher MI are to be computed for the first time.

Case 1. We have a dumbbell-shaped domain, as shown in Figure 1.

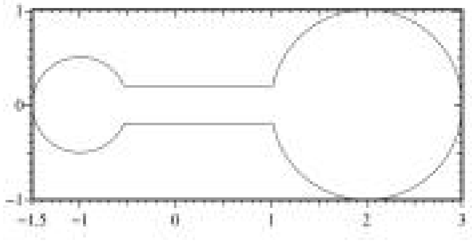


Fig. 1. *A dumbbell-shaped domain.*

We use, respectively, the following three "mound" functions as initial ascent directions.

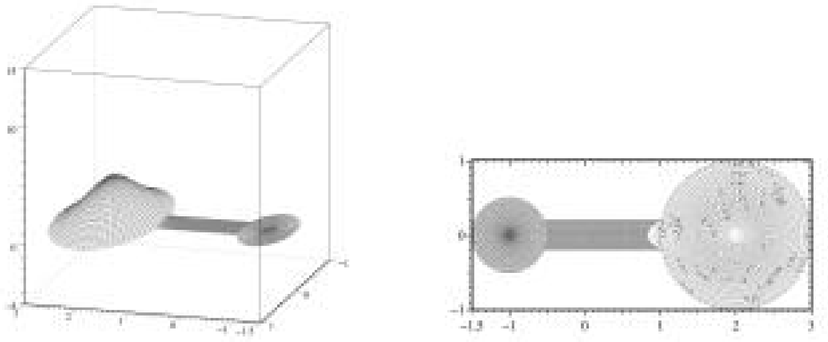


FIG. 2. *The ground state solution* w_1^1 *with MI* $= 1$ *and its contours.* $v_0 = v_0^1$, $\varepsilon = 10^{-4}, J = 10.90, u_{\max} = 3.652$.

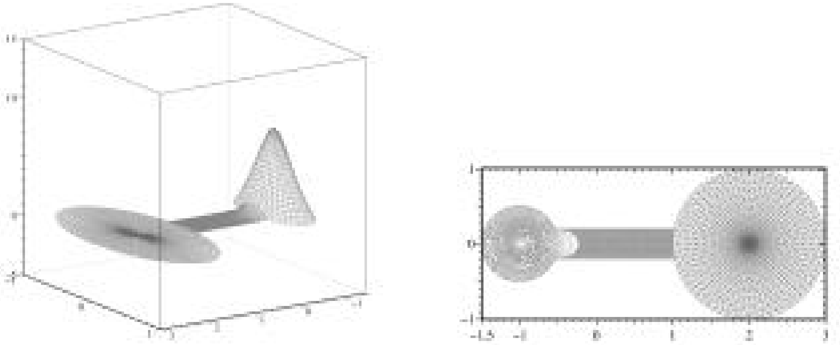


FIG. 3. *The second solution* w_1^2 *with MI* $= 1$ *and its contours.* $v_0 = v_0^2$, $\varepsilon = 10^{-4}$, $J = 42.22, u_{\max} = 7.037$.

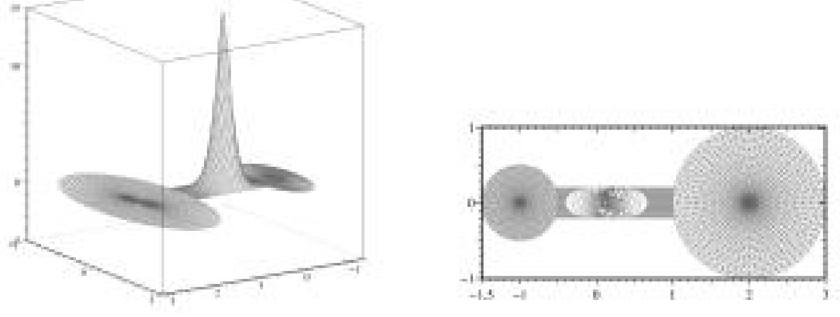


FIG. 4. *The third solution* w_1^3 *with MI* $= 1$ *and its contours.* $v_0 = v_0^3$, $\varepsilon = 10^{-4}$, $J = 159.0, u_{\max} = 13.63$. *So far, the existence of such a positive solution is still an open problem.*

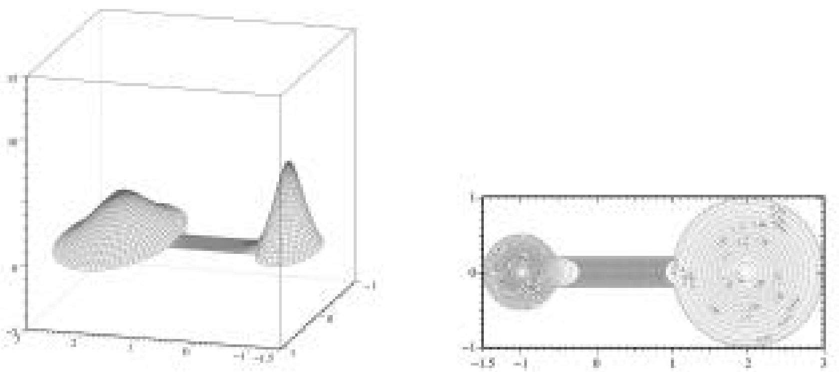


FIG. 5. *A solution w_2 with MI $= 2$ and its contours.* $L = [w_1^1], v_0 = v_0^2$, $\varepsilon = 10^{-4} \times 6, J = 53.12, u_{\max} = 7.037$.

$$v_0^i(x) = \begin{cases} \cos\left(\dfrac{|x - x_i|}{d_i}\dfrac{\pi}{2}\right) & \text{if } |x - x_i| \le d_i, \\ 0 & \text{otherwise,} \end{cases}$$

where $x_1 = (2, 0), d_1 = 1, x_2 = (-1, 0), d_2 = 0.5, x_3 = (0.25, 0)$, and $d_3 = 0.2$.

If we use $L = [w_1^1, w_2]$ and $v_0 = v_0^3$ to search for a solution with MI $= 3$, the algorithm yields a solution with positive and negative peaks. This can be explained as follows. A function with a larger energy value becomes less stable, and a solution with a larger MI is also less stable. Note that

$$J(w_1^1) < J(w_1^2) < J(w_1^3).$$

When we use $L = [w_1^1, w_2]$ and $v_0 = v_0^3$ to search for a solution with MI $= 3$, we start the process at searching for a peak with lower energy for a solution with a lower MI and then go to search for a peak with larger energy for a solution with a higher MI. The process becomes very unstable. Now if we switch the order, we start the process at searching for a peak with higher energy for a solution with a lower MI and then go to next stage to search for a peak with lower energy for a solution with a higher MI. The stability of the process is balanced. Thus we use $L = [w_1^3]$ and $v_0 = v_0^2$ to find a solution w_2^2 with MI $= 2$ and two positive peaks in the left compartment and the central corridor. Then we use $L = [w_1^3, w_2^2]$ and $v_0 = v_0^1$ to search for a positive solution with MI $= 3$, and we obtain the following solution as shown in Figure 6.

Case 2. We have a dumbbell-shaped domain with two cavities.

We use, respectively, the following four “mound” functions as initial ascent directions.

$$v_0^i(x) = \begin{cases} \cos\left(\dfrac{|x - x_i|}{d_i}\dfrac{\pi}{2}\right) & \text{if } |x - x_i| \le d_i, \\ 0 & \text{otherwise,} \end{cases}$$

with

$$x_1 = (2, -0.6), \quad d_1 = 0.4, x_2 = (-0.5, 0), \quad d_2 = 0.2,$$
$$x_3 = (0.25, 0), \quad d_3 = 0.2, \, x_4 = (-1.35, 0), \quad d_4 = 0.15.$$

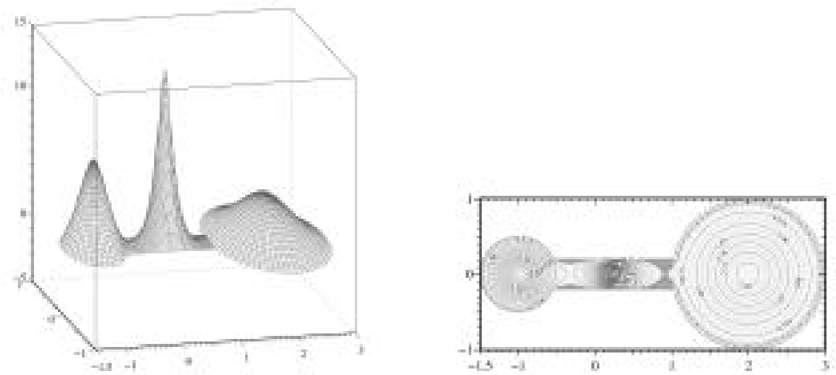


FIG. 6. *A solution w_3 with MI $= 3$ and its contours.* $\varepsilon = 10^{-3}, J = 212.5$*,* $u_{\max} = 13.78$*. This is the only positive solution with MI $= 3$ we can find.*

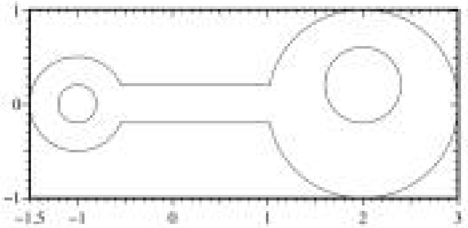


FIG. 7. *A dumbbell-shaped domain with two cavities.*

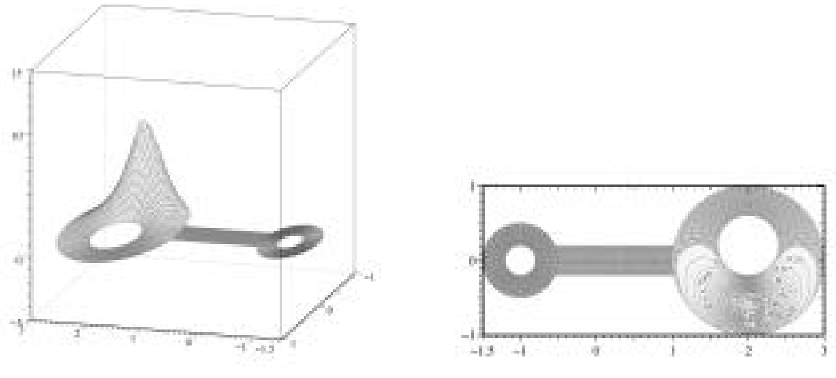


FIG. 8. *The ground state solution w_1^1 with MI $= 1$ and its contours.* $v_0 = v_0^1$*,* $\varepsilon = 10^{-4}, J = 44.18, u_{\max} = 6.664$*.*

Case 3. We have a concentric annulus with inner radius $= 0.7$ and outer radius $= 1$.

The domain is a nice geometric figure. However, due to the symmetry, any solution being rotated for any angle is still a solution. Thus each solution belongs to a one parameter family of solutions. For this case, the existence of nonradially symmetric positive solutions has been established by Coffman [13] and Li [22]. The number of positive peaks that a solution may have depends on the width of the annulus.

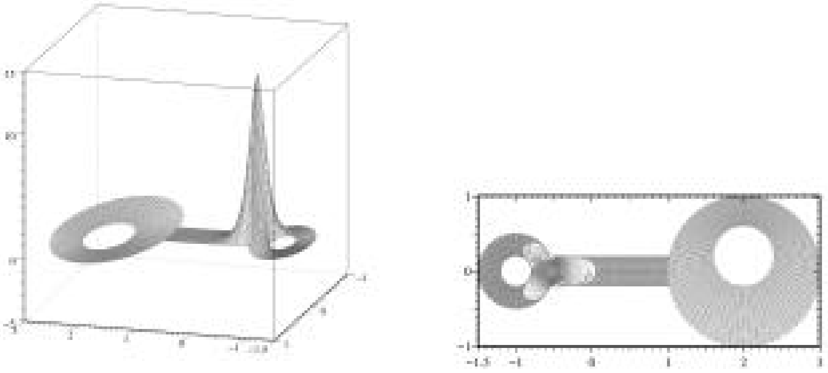


Fig. 9. *The second solution* w_1^2 *with MI* $= 1$ *and its contours.* $v_0 = v_0^3$, $\varepsilon = 10^{-4}, J = 153.5$, $u_{\max} = 12.94$.

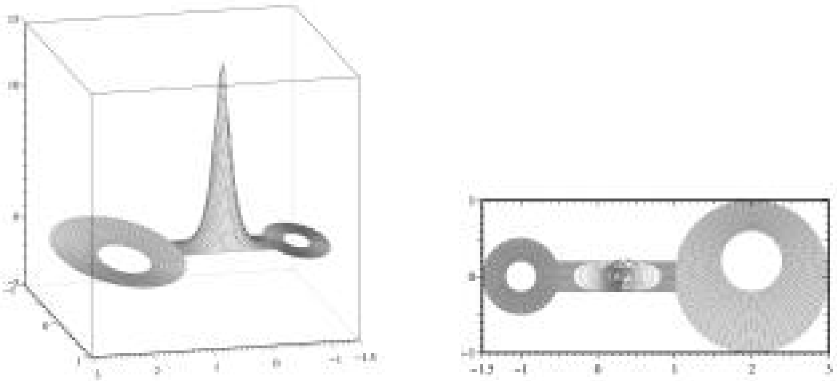


Fig. 10. *The third solution* w_1^3 *with MI* $= 1$ *and its contours.* $v_0 = v_0^2$, $\varepsilon = 10^{-4}, J = 165.6, u_{\max} = 13.64$. *Its profile is similar to* w_1^3 *in Figure* 4*. So far, the existence of such a positive solution is still an open problem.*

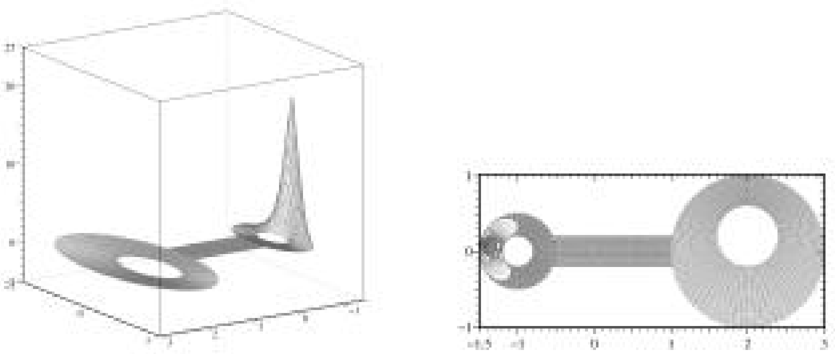


Fig. 11. *The fourth solution* w_1^4 *with MI* $= 1$ *and its contours.* $v_0 = v_0^4$, $\varepsilon = 10^{-4}, J = 286.1, u_{\max} = 17.85$.

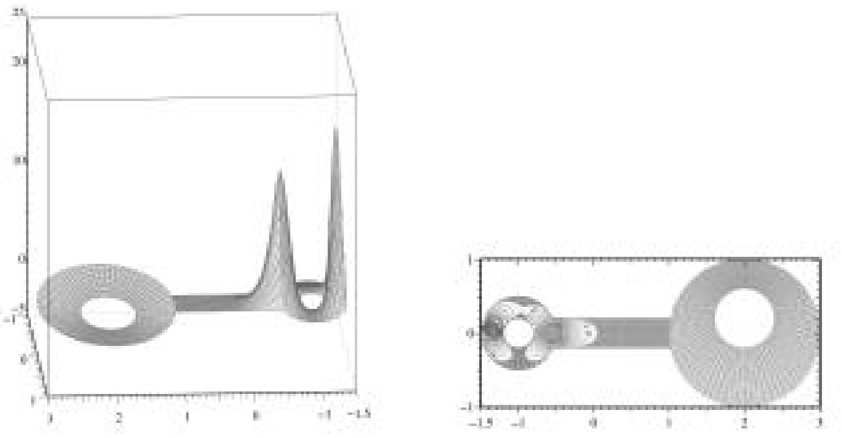


FIG. 12. *A solution* w_2 *with MI* $= 2$ *and its contours.* $L = [w_1^4], v_0 = v_0^2$*,* $\varepsilon = 10^{-4} \times 6, J = 439.5, u_{\max} = 17.85$*. There are other positive solutions with MI* $= 2$*.*

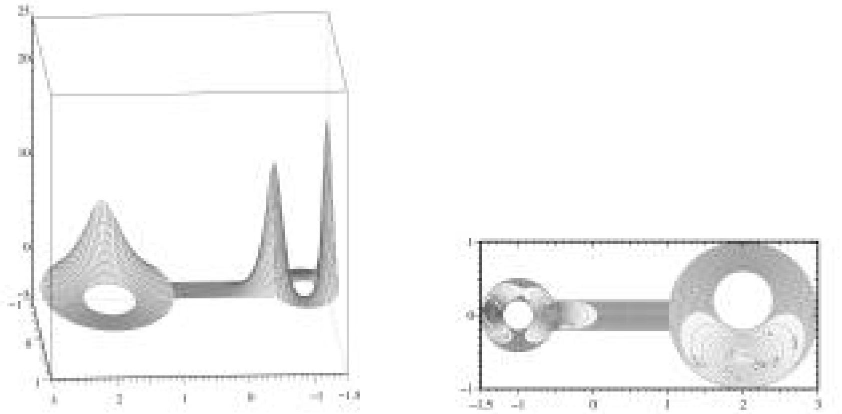


FIG. 13. *A solution* w_3 *with MI* $= 3$ *and its contours.* $L = [w_1^4, w_2], v_0 = v_0^1$*,* $\varepsilon = 10^{-3}, J = 483.6, u_{\max} = 17.86$*. There are other positive solutions with MI* $= 3$*.*

If we utilize the symmetry, we can find a radially symmetric solution that turns out to be a local (global) maximum. Otherwise, this case is highly degenerate. When the boundary is discretized into a polygon, theoretically the case becomes nondegenerate. However, when the discretization is fine, each solution has other solutions nearby. The computation becomes even tougher. After several iterations, there are multiple solutions inside a small neighborhood of the numerical solution. The algorithm may start to wander around. We have used 384 elements on the outer circle, 192 elements on the inner circle, and the following "mound" function as an initial ascent direction for $\theta_i = \frac{(i-1)\pi}{4}$,

$$v_0^i(x) = \begin{cases} \cos\left(\dfrac{|x - 0.85(\cos\theta_i, \sin\theta_i)|}{0.15}\dfrac{\pi}{2}\right) & \text{if } |x - 0.85(\cos\theta_i, \sin\theta_i)| \le 0.15, \\ 0 & \text{otherwise.} \end{cases}$$

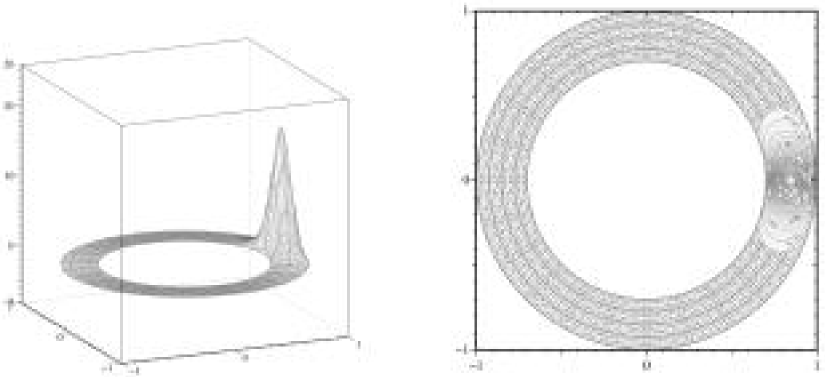


FIG. 14. *A ground state solution w_1 with MI = 1 and its contours.* $v_0 = v_0^1$*,* $\varepsilon = 10^{-4}, J = 289.1, u_{\max} = 18.12$*.*

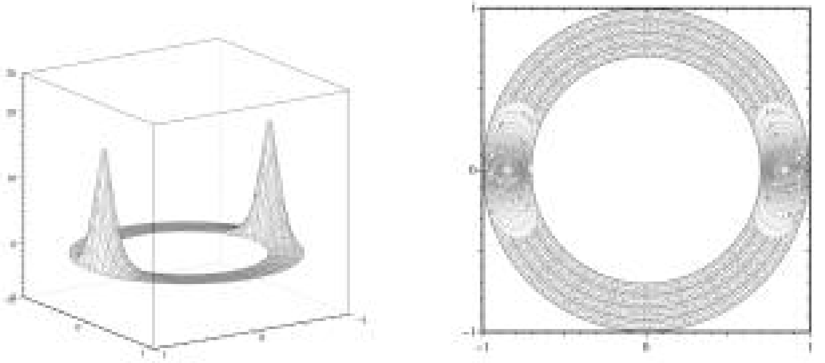


FIG. 15. *A solution w_2 with MI = 2 and its contours.* $L = [w_1], v_0 = v_0^3$*,* $\varepsilon = 10^{-4}, J = 579.0, u_{\max} = 18.12$*.*

Finding multiple saddle points is important for both theory and applications. However, it is very challenging. Little is known in the literature. We try to develop some numerical algorithms and corresponding mathematical theory for finding such saddle points in a stable way. It is known that many saddle points cannot be approximated. One can only numerically approximate those multiple saddle points with some "nice" properties, e.g., minimax solutions. We classify those saddle points through mathematical analysis. The results presented in this paper are under some "reasonably nice" conditions. They provide a mathematical foundation for our further research. Meanwhile, those conditions will be further generalized. Methods to check those conditions (e.g., the continuity or differentiability condition of p) will be developed as research as this direction progresses.[1] So far, the algorithm is still better than mathematical analysis. It produced many interesting numerical results that go beyond theoretical results. For example, when the profiles of the solutions in Figures 4 and 10 were presented to nonlinear PDE analysts in 1997 and 1998, they generated

[1] Results in this project have been obtained and will be presented in a future paper.

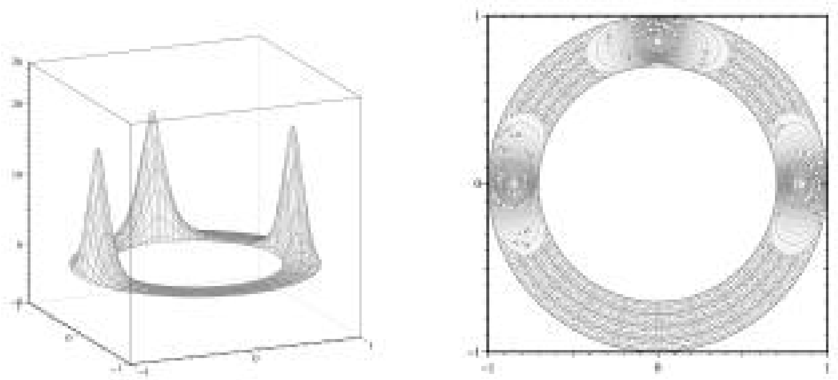


FIG. 16. *A solution w_3 with MI $= 3$ and its contours.* $L = [w_1, w_2], v_0 = v_0^2$*,* $\varepsilon = 10^{-4} \times 6, J = 868.8, u_{\max} = 18.12$.

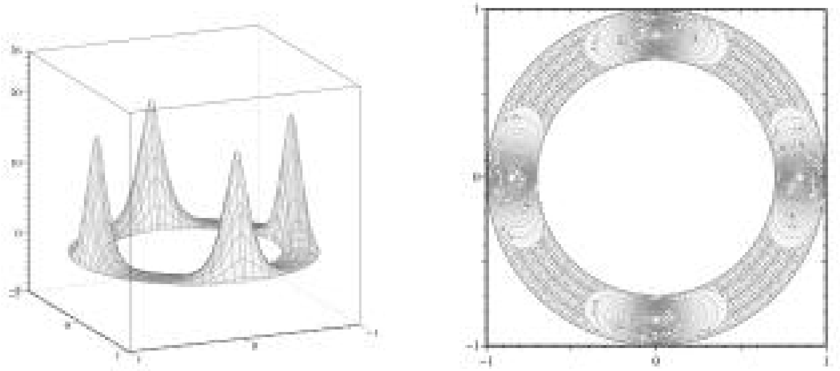


FIG. 17. *A solution w_4 with MI $= 4$ and its contours.* $L = [w_1, w_2, w_3]$*,* $v_0 = v_0^4$*,* $\varepsilon = 10^{-3} \times 3, J = 1159, u_{\max} = 18.12$.

warm debates about the existence and the MIs of such solutions. We are pleased to know that some results on the existence of such solutions have been recently proved (see [35]). Our algorithm can be used to solve for a critical point which is not a minimax solution, e.g., a Monkey saddle point. However, the analysis is beyond the scope of any minimax principle; a more profound approach is required. As mathematical analysis in this research progresses, the algorithm will be accordingly modified. Convergence is a paramount issue of any numerical algorithm. The MI of a solution is an important notion that provides understanding of the local structure of a saddle point and can be used to measure instability of a saddle point. Although, in the above numerical examples, we have printed the MI for each numerical solution, it is based on the way we compute the solution in the algorithm; its mathematical verification has not been established. Due to the length limitations of this paper, results on those issues will be addressed in a subsequent paper [20] and future papers [21, 37].

Acknowledgment. The authors would like to thank two anonymous referees for their helpful comments.

REFERENCES

[1] R. A. Adams, *Sobolev Spaces*, Academic Press, New York, 1975.

[2] H. Amann, *Supersolution, monotone iteration and stability*, J. Differential Equations, 21 (1976), pp. 367–377.

[3] A. Ambrosetti and P. Rabinowitz, *Dual variational methods in critical point theory and applications*, J. Funct. Anal., 14 (1973), pp. 349–381.

[4] J. Aubin and I. Ekeland, *Applied Nonlinear Analysis*, Wiley, New York, 1984.

[5] T. Bartsch and Z. Q. Wang, *On the existence of sign-changing solutions for semilinear Dirichlet problems*, Topol. Methods Nonlinear Anal., 7 (1996), pp. 115–131.

[6] V. Benci and G. Cerami, *The effect of the domain topology on the number of positive solutions of nonlinear elliptic problems*, Arch. Ration. Mech. Anal., 114 (1991), pp. 79–94.

[7] L. Bieberbach, *$\Delta u = e^u$ und die automorphen Functionen*, Math. Ann., 77 (1916), pp. 173–212.

[8] H. Brezis and L. Nirenberg, *Remarks on finding critical points*, Comm. Pure Appl. Math., 44 (1991), pp. 939–963.

[9] A. Castro, J. Cossio, and J. M. Neuberger, *A sign-changing solution for a superlinear Dirichlet problem*, Rocky Mountain J. Math., 27 (1997), pp. 1041–1053.

[10] K. C. Chang, *Infinite Dimensional Morse Theory and Multiple Solution Problems*, Birkhäuser Boston, Boston, 1993.

[11] G. Chen, W. Ni, and J. Zhou, *Algorithms and visualization for solutions of nonlinear elliptic equations*, Internat. J. Bifur. Chaos Appl. Sci. Engrg., 10 (2000), pp. 1565–1612.

[12] Y. S. Choi and P. J. McKenna, *A mountain pass method for the numerical solution of semilinear elliptic problems*, Nonlinear Anal., 20 (1993), pp. 417–437.

[13] C. V. Coffman, *A nonlinear boundary value problem with many positive solutions*, J. Differential Equations, 54 (1984), pp. 429–437.

[14] E. N. Dancer, *The effect of domain shape on the number of positive solutions of certain nonlinear equations*, J. Differential Equations, 74 (1988), pp. 120–156.

[15] Y. Deng, G. Chen, W. M. Ni, and J. Zhou, *Boundary element monotone iteration scheme for semilinear elliptic partial differential equations*, Math. Comp., 65 (1996), pp. 943–982.

[16] W. Y. Ding and W. M. Ni, *On the existence of positive entire solutions of a semilinear elliptic equation*, Arch. Ration. Mech. Anal., 91 (1986) pp. 283–308.

[17] Z. Ding, D. Costa, and G. Chen, *A high linking method for sign changing solutions for semilinear elliptic equations*, Nonlinear Anal., 38 (1999) pp. 151–172.

[18] D. Greenspan and S. V. Parter, *Mildly nonlinear elliptic partial differential equations and their numerical solution,* II, Numer. Math., 7 (1965), pp. 129–146.

[19] K. Ishihara, *Monotone explicit iterations of the finite element approximations for the nonlinear boundary value problems*, Numer. Math., 45 (1984), pp. 419–437.

[20] Y. Li and J. Zhou, *Convergence results of a minimax method for finding critical points*, SIAM J. Sci. Comput., submitted.

[21] Y. Li and J. Zhou, *Local characterizations of saddle points and their Morse indices*, in Advances in Control of Nonlinear Distributed Parameter Systems, Marcel Dekker, New York, 2001, pp. 233–251.

[22] Y. Y. Li, *Existence of many positive solutions of semilinear elliptic equations on annulus*, J. Differential Equations, 83 (1990), pp. 348–367.

[23] F. Lin and T. Lin, *Minimax solutions of the Ginzburg–Landau equations*, Slecta Math. (N.S.), 3 (1997), pp. 99–113.

[24] J. Mawhin and M. Willem, *Critical Point Theory and Hamiltonian Systems*, Springer-Verlag, New York, 1989.

[25] Z. Nehari, *On a class of nonlinear second-order differential equations*, Trans. Amer. Math. Soc., 95 (1960), pp. 101–123.

[26] W. M. Ni, *Some Aspects of Semilinear Elliptic Equations*, Department of Mathematics, National Tsing Hua University, Hsinchu, Taiwan, Republic of China, 1987.

[27] W. M. Ni, *Recent progress in semilinear elliptic equations*, in RIMS Kokyuroku 679, Kyoto University, Kyoto, Japan, 1989, pp. 1–39.

[28] C. V. Pao, *Nonlinear Parabolic and Elliptic Equations*, Plenum Press, New York, 1992.

[29] S. V. Parter, *Mildly nonlinear elliptic partial differential equations and their numerical solutions* I, Numer. Math., 7 (1965), pp. 113–128.

[30] P. Rabinowitz, *Minimax Method in Critical Point Theory with Applications to Differential Equations*, CBMS Reg. Conf. Ser. Math. 65, AMS, Providence, RI, 1986.

[31] M. Schechter, *Linking Methods in Critical Point Theory*, Birkhäuser Boston, Boston, 1999.

[32] S. Shi, *Ekeland's variational principle and the mountain pass lemma*, Acta Math. Sinica, 1 (1985), pp. 348–355.

[33] M. Struwe, *Variational Methods*, Springer-Verlag, Berlin, 1996.

[34] Z. Wang, *On a superlinear elliptic equation*, Ann. Inst. H. Poincaré, Anal. Non Linéaire, 8 (1991), pp. 43–57.

[35] J. Wei and L. Zhang, *On the Effect of the Domain Shape on the Existence of Large Solutions of Some Superlinear Problems*, Department of Mathematics, The Chinese University of Hong Kong, Shatin, Hong Kong, China, Institute of Mathematics, Academia Sinica, Beijing, China, 2000, preprint.

[36] M. Willem, *Minimax Theorems*, Birkhäuser Boston, Boston, 1996.

[37] J. Zhou, *Instability Indices of Saddle Points by a Local Minimax Method*, Department of Mathematics, Texas A & M University, College Station, TX, 2000, preprint.

SIAM J. SCI. COMPUT.
Vol. 23, No. 3, pp. 866–890

A LIE ALGEBRAIC APPROACH TO NUMERICAL INTEGRATION OF STOCHASTIC DIFFERENTIAL EQUATIONS[*]

TETSUYA MISAWA[†]

Abstract. In this article, "composition methods (or operator splitting methods)" for autonomous stochastic differential equations (SDEs) are formulated to produce numerical approximation schemes for the equations. In the proposed methods, the exponential map, which is given by the solution of an SDE, is approximated by composition of the stochastic flows derived from simpler and exactly integrable vector field operators having stochastic coefficients. The local and global errors of the numerical schemes derived from the stochastic composition methods are investigated. The new schemes are advantageous to preserve the special character of SDEs numerically and are useful for approximations of the solutions to stochastic nonlinear equations. To examine their superiority, several numerical simulations on the basis of the proposed schemes are carried out for SDEs which arise in mathematical finance and stochastic Hamiltonian dynamical systems.

Key words. stochastic differential equations, Lie algebra, composition methods (operator splitting method), mathematical finance, stochastic Hamiltonian dynamical systems, stochastic nonlinear system

AMS subject classifications. 65C30, 17B66, 91B28, 70-08, 70H33

PII. S106482750037024X

1. Introduction. The theory of stochastic differential equations (SDEs) is understood as a fundamental tool for the description of random phenomena treated in physics, engineering, economics, and so on. Particularly, as the famous Black–Scholes option pricing model, the stochastic equations are used to describe the price process of underlying asset in mathematical finance. However, it is often difficult to obtain the solutions of SDEs explicitly, and hence there has been increasing interest in numerical analysis of SDEs. Indeed, many numerical schemes for SDEs have been proposed (e.g., [8], [15], [23]).

The purpose of the present article is to propose some new numerical schemes for autonomous SDEs on the basis of "composition methods." The reasons why we address this topic are as follows.

In numerical analysis for deterministic ordinary differential equations, whether or not some special character or structure of the equations is preserved precisely is an important point in performing reliable numerical calculations. From this point of view, various numerical methods to realize the characters of differential equations have been proposed. Indeed, we can find such examples as energy conservative methods [10], [13], symplectic integrators for Hamiltonian dynamical systems [24], [7], [26], and composition methods [19], [22]. In particular, the composition methods are useful to produce numerical schemes which leave some structure or character of general differential equations numerically invariant. Hence, it seems to be quite natural that we investigate the methods for SDEs to produce numerical schemes having the conservation properties; this is the first reason for setting up the numerical schemes proposed here.

[*] Received by the editors April 5, 2000; accepted for publication (in revised form) May 22, 2001; published electronically August 29, 2001. This work was supported by Grant-in-Aid for Scientific Research 11640132 from the Japan Society for the Promotion of Science.

http://www.siam.org/journals/sisc/23-3/37024.html

[†] Faculty of Economics, Nagoya City University, Nagoya 467-8501, Japan (misawa@econ.nagoya-cu.ac.jp).

Moreover, in the theory of differential equations, composition methods are known as operator splitting methods, and they are often utilized for approximation of nonlinear equations to which solutions are not obtained explicitly [25], [12]. From this viewpoint, we may expect that the methods bring us a convenient and powerful way of approximations for "stochastic nonlinear" differential equations, and this is the second reason for our investigation.

We will first outline the original composition methods for ordinary differential equations. Let X denote vector fields on some space with coordinates x with flows $\exp(tX)$; that is, the solutions of differential equations of the form $\dot{x}(t) = X(x)$ are given in the form $x(t) = \exp(tX)(x(0))$. Then, the vector field X is to be integrated numerically with fixed time step t. In this framework, we can apply composition methods to the differential equation if we can write $X = A + B$ in such a way that $\exp(tA)$ and $\exp(tB)$ can both be calculated *explicitly.* More generally, this can be relaxed by approximations of the exponential maps. In the most elementary case, the method gives the approximation for $x(t)$ through

$$\tilde{x}(t) = \exp(tA)\exp(tB)(x) = x(t) + O(t^2);$$

the last equality is obtained by using Baker–Campbell–Hausdorff (BCH) formula, which is well known in Lie algebraic theory.

Thus, in composition methods, we use the exponential representation of solutions to differential equations as an important tool. To formulate the methods for SDEs, therefore, one needs a stochastic version of the notion of exponential representation of solutions. In [16], this topic has been investigated in detail. Hence, in section 2, we first review Kunita's work on an explicit expression of solutions of SDEs as a functional of multiple Wiener integrals. We next give a formal extension of Kunita's explicit expression of solutions of SDEs, and on the basis of the formal result, we will propose new schemes for SDEs. Moreover, to estimate the error for the new schemes, we touch upon some lemmas and a proposition concerning mean-square order of multiple Wiener integrals.

In section 3, on the basis of the results in section 2, composition methods are formulated for autonomous SDEs with a one-dimensional Wiener process. Through these methods, we will obtain some numerical schemes for the stochastic equations. Then, the approximation error of numerical solutions derived from the schemes must be estimated. In this paper, we first apply the local error estimation in mean-square sense to the obtainable numerical solutions. The BCH formula will be useful for calculating the approximation error in a way analogous to that in the deterministic case. Using the result on local error estimation, we next address global error estimation for our new schemes. At the present stage, we obtain only local and global error orders through indirect estimation, and hence such orders will be said to be "in a weak sense."

In section 4, in order to examine the superiority of the new schemes, we investigate some illustrative examples of the numerical simulations using the proposed schemes. In the first example, the following nonlinear scalar SDE is treated, which is often adopted as a model of an asset price process in mathematical finance [9]:

$$dS(t) = S(t)dt + 2\sqrt{S(t)} \circ dW(t), \quad S(0) = s(> 0). \tag{1.1}$$

It is known that the value of the solution to this equation is "nonnegative" for any $t \in [0, T]$. Through the standard stochastic numerical schemes, however, such a character of this equation is not always preserved numerically. In contrast with this, we will see that our new numerical schemes by composition methods in section 3 leave the structure invariant numerically, and thereby the first advantage of composition methods is revealed. In the second and third examples, we examine the second advantage of composition methods as a tool of approximation for stochastic nonlinear systems. Particularly, such a superiority is often found in Hamiltonian dynamical systems through the dimensional splitting methods. Therefore, in the third example, we treat the composition methods for "stochastic" Hamiltonian systems. At the end of that section, to show the advantage of composition methods, we further touch upon a way to produce numerical schemes which realize the numerical preservation of conserved quantities for stochastic systems [20], [21].

Some concluding remarks and a discussion of future problems are given in section 5.

Finally, we note the recent remarkable work by K. Burrage and P.M. Burrage [4] as another numerical approach to SDEs through Lie algebra.

2. Representation of solutions of SDEs. In this section, we first review Kunita's work mentioned in section 1 [16]. Next we will give a formal extension of Kunita's explicit expression of solutions of SDEs. On the basis of this formal expression, our new schemes for SDEs are formulated in the next section. At the end of this section, as a preparation for the error estimation of the new schemes, we will discuss mean-square order of multiple Wiener integrals dealt with in the representation of solutions of SDEs.

2.1. Review of Kunita's explicit expression of solutions of SDEs. In order to explain Kunita's main result, we first address Campbell–Hausdorff formula for n vector fields. Let $Y_1, \dots, Y_n$ be n C^∞ vector fields on a connected C^∞-manifold M of dimension d. Suppose that $Y^{i_1 \cdots i_m} = [[\dots[Y_{i_1}, Y_{i_2}]\dots]Y_{i_m}], \quad m = 1, 2, \dots$, and their sums are all complete vector fields, where $[X, Y]$ is the Lie bracket defined by $XY - YX$. Moreover, we assume that the power series

$$Z = \sum_{m=1}^{\infty} \sum_{(i_1 \cdots i_m)} c_{i_1 \cdots i_m} Y^{i_1 \cdots i_m} \tag{2.1}$$

is absolutely convergent and defines a complete vector field. Here, for a given multi-index $I = (i_1, \dots, i_m)$, each coefficient $c_{i_1 \cdots i_m}$ is determined through (2.3) given below. Then (2.2) holds

$$\exp Y_n \cdots \exp Y_1 = \exp Z. \tag{2.2}$$

This is the Campbell–Hausdorff formula for n vector fields [16].

We will touch upon the explicit form of the coefficients $c_{i_1 \cdots i_m}$. Let us divide the multi-index $I = (i_1, \dots, i_m)$ into a sequence of shorter ones $I_j, j = 1, \dots, \ell$, and denote it by $\hat{I}$;

$$\hat{I} = (I_1, \dots, I_{k_1})(I_{k_1+1}, \dots, I_{k_2}) \cdots (I_{k_{\ell-1}+1}, \dots, I_{k_\ell}),$$

where each I_k consists of the same number $\hat{i}_k$ and the numbers $\hat{i}_k, k = 1, \dots, k_\ell$, satisfy

$$\hat{i}_1 > \hat{i}_2 > \cdots > \hat{i}_{k_1} < \hat{i}_{k_1+1} > \cdots > \hat{i}_{k_2} \cdots < \hat{i}_{k_{\ell-1}+1} > \cdots > \hat{i}_{k_\ell}.$$

We call $\hat{I}$ a natural division. Moreover, we denote the number of elements in I_k by n_k, where $\sum_{k=1}^{k_\ell} n_k = m$. Divide again each index I_k into j_k indices, each of which consists of $n_k^{(i)} (i = 1, \dots, j_k)$ elements (hence $\sum_{i=1}^{j_k} n_k^{(i)} = n_k$). Then according to [16], the coefficients $c_{i_1 \cdots i_m}$ are given by (2.3):

$$\begin{aligned} c_{i_1,\cdots i_m} &= \frac{1}{m} \sum_{s=0}^{\ell-1} \sum_{*} {}_{\ell-1}C_s (-1)^{j_1+\cdots+j_{k_\ell}-s-1} (j_1 + \cdots + j_{k_\ell} - s)^{-1} \\ &\quad \times \frac{1}{n_1^{(1)}! \cdots n_1^{(j_1)}! \cdots n_{k_\ell}^{(1)}! \cdots n_{k_\ell}^{(j_\ell)}!}, \end{aligned} \tag{2.3}$$

where the sum $\sum_*$ is taken for all subdivisions of $I_k (k = 1, \dots, k_\ell)$, that is, for positive integers $n_k^{(i)} (i = 1, \dots, j_k; k = 1, \dots, k_\ell)$ under $\sum_{i=1}^{j_k} n_k^{(i)} = n_k$.

Let I' be another multi-index of length m such that the natural division is given by $\hat{I}' = (I'_1, \dots, I'_{k'_1}) \cdots (I'_{k'_{\ell-1}+1}, \dots, I'_{k'_\ell})$. We say I and I' are equivalent if for each k, I_k and I'_k contain the same number of elements and $k'_j = k_j (j = 1, \dots, \ell)$ hold. Then we note that $c_I = c_{I'}$ holds.

Now, we proceed to our stochastic systems governed by SDEs. Let us consider an autonomous SDE of Stratonovich type (e.g., [11], [1]) under the probability space $(\Omega, \mathcal{F}, \mathcal{P})$:

$$dS(t) = b(S(t))dt + \sum_{j=1}^{r} g_j(S(t)) \circ dW^j(t) \tag{2.4}$$

defined on a connected C^∞-manifold M of dimension d, where $b = (b^i)_{i=1}^d$ and $g_j = (g_j^i)_{i=1}^d$ $(j = 1, \dots, r)$ are d-dimensional C^∞ functions on M, respectively, and $W(t) = (W^1(t), \dots, W^r(t))$ is a standard Wiener process. Here $S(t)$ is assumed to be adapted with a nondecreasing family of sigma-algebra $(\mathcal{F}_t)_{t \geq 0} \subset \mathcal{F}$. Note that (2.4) is rewritten in the form of SDE of Itô type as follows:

$$dS_t = \{b(S(t)) + \frac{1}{2} \sum_{k=1}^{r} \sum_{i=1}^{d} g_k^i \partial_i g_k(S(t))\}dt + \sum_{j=1}^{r} g_j(S(t)) dW^j(t), \tag{2.5}$$

where $\partial_i = \partial / \partial S^i$. Using the coefficient-functions in (2.4), we define C^∞ vector fields $X_0, X_1, \dots, X_r$ as follows:

$$X_0 = \sum_{i=1}^{d} b^i \partial_i, \quad X_j = \sum_{i=1}^{d} g_j^i \partial_i \quad (j = 1, \dots, r). \tag{2.6}$$

We will now review Kunita's result (Lemma 2.1 in [16]). For this purpose, we begin with notations on multi-index. Let us divide a multi-index $I = (i_1, \dots, i_m)$ $(i_1, \dots, i_m \in 0, 1, \dots, r)$ into shorter ones $I_1 \cdots I_q (q \leq m)$, where each I_k consists of the same element $\hat{i}_j$. For given positive integers $k_1 < k_2 < \cdots < k_\ell = q$, we define a division of I as

$$\Delta I = (I_1, \dots, I_{k_1})(I_{k_1+1}, \dots, I_{k_2}) \cdots (I_{k_{\ell-1}+1}, \dots, I_{k_\ell}). \tag{2.7}$$

We call ΔI single or double when each I_k contains a single element or at most two. Here we remark that ΔI may not be equal to a natural division of I; but if there is an index $\tilde{I}$ such that its natural division is equivalent to ΔI, we set $c_{\Delta I} = c_{\tilde{I}}$ in (2.3) for convention.

Now suppose that we are given an index I and a divided index ΔI. Moreover, we set $W^0(t) = t$. For a single divided index ΔI, we define the multiple Wiener–Stratonovich integral $W^{\Delta I}(t)$ as

$$W^{\Delta I}(t) = \int \cdots \int_A \circ dW^{i_1}(t_1) \cdots \circ dW^{i_m}(t_m), \tag{2.8}$$

where $A = \{t_{k_1} < \cdots < t_1 < t, \dots, t_{k_\ell} < \cdots < t_{k_{\ell-1}+1} < t, t_{k_i} < t_{k_i+1}\,(i = 1, \dots, \ell)\}$. On the other hand, if ΔI is a double index, we define

$$W^{\Delta I}(t) = \int \cdots \int_A \circ dW^{I_1}(t_1) \cdots \circ dW^{I_{k_\ell}}(t_\ell), \tag{2.9}$$

where

$$\begin{aligned} W^{I_k}(t) &= W^{i_k}(t) \quad (I_k : \text{single}; \quad I_k = \{i_k\}) \\ &= t \quad (I_k : \text{double}; \quad I_k = \{i_k, i_k\} \quad \text{and} \quad i_k \neq 0) \\ &= 0 \quad (I_k = \{0, 0\}). \end{aligned} \tag{2.10}$$

Under the notations defined above, Kunita proved the following lemma.

Kunita's lemma. Suppose that the Lie algebra $L = L(X_0, \dots, X_r)$ generated by $X_0, \dots, X_r$ is nilpotent of step p. Then the solution $S(t)$ of (2.4) with $S(0) = s$ is represented as

$$S(t) = \exp(Y_t)(s), \tag{2.11}$$

where $Y_t(\omega)$ is the vector field for each t and almost surely (a.s.) $\omega \in \Omega$ given by

$$Y_t = \sum_{i=0}^{r} W^i(t) X_i + \sum_{J;2 \le |J| \le p} \left\{ {\sum_{\Delta J}}^{*} c_{\Delta J} W^{\Delta J}(t) \right\} X^J, \tag{2.12}$$

and $X^J = [[\cdots[X_{j_1}, X_{j_2}]\cdots]X_{j_m}] \quad (J = (j_1, \dots, j_m))$. Here, $\sum_{\Delta J}^{*}$ is the sum for all single and double divided indices of J, and $c_{\Delta J}$ are the coefficients determined through (2.3) for the divided indices of J. Moreover, $|J|$ denotes the length of a multi-index J.

We remark that (2.11) with (2.12) means that the solution $S(t, \omega)$ equals $\phi(1, s, \omega)$ a.s., where $\phi(\tau, s, \omega)$ is the solution of the *ordinary* differential equation

$$\frac{d\phi(\tau)}{d\tau} = Y_t(\omega)(\phi(\tau)), \quad \phi(0) = s, \tag{2.13}$$

regarding t and ω as parameters.

The proof of Kunita's lemma is outlined as follows. For a fixed positive integer n and positive time t, we define $\delta W_k^i = W^i(\frac{k}{n}t) - W^i(\frac{(k-1)}{n}t) \quad (i = 1, \dots, r)$ and define $Z_j = \frac{t}{n} X_0 + \sum_{i=1}^{r} \delta W_j^i(t) X_i \quad (j = 1, \dots, n)$. We further set

$$S^n(t) = (\exp Z_n \cdots \exp Z_1)(s). \tag{2.14}$$

Then, there is a subsequence of $S^n(t)$ converging to $S(t)$ a.s.. On the other hand, applying the Campbell–Hausdorff formula to the right-hand side of (2.14), we obtain

$$\sum_J \left\{ \sum_{\Delta J} c_{\Delta J} \sum_{I:\hat{I} \sim \Delta J} \delta W_{i_1}^{j_1} \cdots \delta W_{i_m}^{j_m} \right\} X^J,$$

where $\sum_{I:\hat{I}\sim\Delta J} \delta W_{i_1}^{j_1} \cdots \delta W_{i_m}^{j_m}$ means the sum for all indices I such that the natural division $\hat{I}$ is equivalent to ΔJ. Then the sum converges to $W^{\Delta J}(t)$ if ΔJ is a single or double index, and it converges to 0 if ΔJ is more than double. Note that the last assertion corresponds to the following result from formal calculus with respect to stochastic differentials dt and $dW^i \quad (i = 1, \ldots, r)$ [11]; $dW^i dW^j = \delta_{ij} dt$, $dtdW^i = 0$, $dtdt = 0$, and $dW^i dW^j dW^k = 0$. Finally, we can also prove that the sum converges to the right-hand side of (2.11) with (2.12) a.s..

Remark 2.1. According to Lemma 2.2 in Kunita's paper [16], we can calculate the coefficients of X^J in the case of $|J| = 2, 3$. On account of these results, the vector field (2.12) is rewritten as

$$
\begin{aligned}
(2.15)\quad Y_t = &\sum_{i=0}^{r} W^i(t) X_i + \frac{1}{2}\sum_{i<j}^{r} [W^i, W^j](t)[X_i, X_j] \\
&+\frac{1}{36}\sum_{i=0}^{r}\sum_{j=1}^{r} tW^i(t)[[X_i, X_j], X_j] \\
&+\frac{1}{18}\sum_{i<j,k}^{r} [[W^i, W^j], W^k](t)[[X_i, X_j], X_k] \\
&+\sum_{J;4\le|J|\le p}\left\{\sum_{\Delta J}{}^{*} c_{\Delta J} W^{\Delta J}(t)\right\} X^J \ (i = 0, 1, \ldots, r; j, k = 1, \ldots, r).
\end{aligned}
$$

In (2.15), we define $[W^i, W^j](t)$ and $[[W^i, W^j], W^k](t)$ as multiple Wiener–Stratonovich integrals of degrees equal to 2 and 3 given by

$$[W^i, W^j](t) = \int_0^t W^i(\tau) \circ dW^j(\tau) - \int_0^t W^j(\tau) \circ dW^i(\tau)$$

and

$$[[W^i, W^j], W^k](t) = \int_0^t [W^i, W^j](\tau) \circ dW^k(\tau) - \int_0^t W^k(\tau) \circ d[W^i, W^j](\tau),$$

respectively.

Note that the condition that Lie algebra L is nilpotent of step p is required only in order to make the sum in (2.12) finite. This suggests that if L is general, we may *formally* represent the solution $S(t)$ of (2.4) as

$$(2.16) \qquad S(t) = (\exp Y_t)(s),$$

where

$$(2.17) \qquad Y_t = \sum_{i=0}^{r} W^i(t) X_i + \sum_{J;2\le|J|}\left\{\sum_{\Delta J}{}^{*} c_{\Delta J} W^{\Delta J}(t)\right\} X^J,$$

and this is just the formal expression we want. In what follows, we suppose that one may obtain the explicit representation of solution (2.16) with (2.17).

Remark 2.2. If the Lie algebra generated by $X_0, X_1, \ldots, X_r$ is of finite dimension, Ben Arous has proved that the stochastic infinite series in (2.17) actually converges before a stopping time. Therefore, in such a case, the representation of solution (2.16) with (2.17) is well-defined (Theorem 20 in [2]). We will see such examples in subsections 4.1 and 4.2 of section 4.

Moreover, we restrict ourselves to the SDEs (2.4) with a one-dimensional Wiener process; that is, we consider

$$dS(t) = b(S(t))dt + g(S(t)) \circ dW(t), \tag{2.18}$$

where b and g are d-dimensional vector-valued C^∞ functions. Then, we may rewrite (2.17) in a simpler form in terms of Kloeden–Platen's representation for multiple Wiener–Stratonovich integrals and multiple Wiener–Itô ones [14]; they are defined by

$$J_{(\alpha)}(t,s) = \int_s^{s+t}\int_s^{\tau_k}\cdots\int_s^{\tau_2} \circ dY^{(j_1)}(\tau_1)\cdots \circ dY^{(j_{k-1})}(\tau_{k-1}) \circ dY^{(j_k)}(\tau_k), \tag{2.19}$$

$$I_{(\alpha)}(t,s) = \int_s^{s+t}\int_s^{\tau_k}\cdots\int_s^{\tau_2} dY^{(j_1)}(\tau_1)\cdots dY^{(j_{k-1})}(\tau_{k-1}) dY^{(j_k)}(\tau_k), \tag{2.20}$$

respectively, where $\alpha = (j_1, \dots, j_k)$ $(j_i = 0, 1; i = 1, \dots, k)$ and

$$dY^{(j)}(u) = \begin{cases} du & \text{for } j = 0, \\ dW(u) & \text{for } j = 1. \end{cases}$$

In what follows, we denote $J_{(\alpha)}(t,s)$ and $I_{(\alpha)}(t,s)$ by $J_{(\alpha)}(t)$ and $I_{(\alpha)}(t)$, respectively, if $s = 0$.

Remark 2.3. Note that $J_{(\alpha)}(t,s)$ can be rewritten in terms of $I_{(\alpha)}(t,s)$ (see pp. 174–175 in [15]). For example, we have

$$J_{(j_1)} = I_{(j_1)} \quad (j_1 = 0, 1), \tag{2.21}$$

$$J_{(j_1,j_2)} = I_{(j_1,j_2)} + \frac{1}{2}1_{\{j_1=j_2=1\}}I_{(0)} \quad (j_1, j_2 = 0, 1), \tag{2.22}$$

$$J_{(j_1,j_2,j_3)} = I_{(j_1,j_2,j_3)} + \frac{1}{2}(1_{\{j_1=j_2=1\}}I_{(0,j_3)} + 1_{\{j_2=j_3=1\}}I_{(j_1,0)}) \quad (j_1, j_2, j_3 = 0, 1), \tag{2.23}$$

where $1_{\{\cdot\}}$ denotes the defining function. In general, any multiple Stratonovich integral $J_{(\alpha)}$ can be written as a multiple Itô integral $I_{(\alpha)}$ or a finite sum of $I_{(\alpha)}$ and multiple Itô integrals $I_{(\beta)}$ satisfying

$$\ell(\alpha) + n(\alpha) \le \ell(\beta) + n(\beta), \tag{2.24}$$

where $\ell(\alpha)$ ={the number of elements of α} and $n(\alpha)$ ={the number of 0's in the elements of α} (Remark 5.2.8 in [15]).

In terms of (2.19) and stochastic Stratonovich integration by parts formula for $W^i(t)$ $(i = 0, 1)$ and $[W^0, W^1](t)$, we can rewrite (2.17) in the following form:

$$\begin{aligned} Y_t = {} & J_{(0)}(t)X_0 + J_{(1)}(t)X_1 + \frac{1}{2}(J_{(0,1)}(t) - J_{(1,0)}(t))[X_0, X_1] \\ & + \frac{1}{18}\{2J_{(0,1,0)}(t) - 2J_{(1,0,0)}(t) + J_{(0)}(t)J_{(1,0)}(t) - J_{(0)}(t)J_{(0,1)}(t)\}[[X_0, X_1], X_0] \\ & + \frac{1}{18}\{2J_{(0,1,1)}(t) - 2J_{(1,0,1)}(t) + J_{(1)}(t)J_{(1,0)}(t) - J_{(1)}(t)J_{(0,1)}(t)\}[[X_0, X_1], X_1] \\ & + \frac{1}{36}\{J_{(0)}(t)\}^2[[X_0, X_1], X_1] + \sum_{J;4\le|J|} K^J(t)X^J. \end{aligned} \tag{2.25}$$

Here, in the last term on the right-hand side of the above equation, $K^J(t) = \{\sum_{\Delta J}{}^{*} c_{\Delta J} W^{\Delta J}(t)\}$, and $J = (j_1, \dots, j_\ell)$ $(j_i = 0, 1; i = 1, \dots, \ell;\ \ell \geq 4)$. In terms of Remark 2.3, this can be described by multiple Wiener–Itô integrals as follows:

$$\begin{aligned}
(2.26)\qquad Y_t &= I_{(0)}(t)X_0 + I_{(1)}(t)X_1 + \frac{1}{2}(I_{(0,1)}(t) - I_{(1,0)}(t))[X_0, X_1] \\
&+\frac{1}{18}\{2I_{(0,1,0)}(t) - 2I_{(1,0,0)}(t) + I_{(0)}(t)I_{(1,0)}(t) - I_{(0)}(t)I_{(0,1)}(t)\}[[X_0, X_1], X_0] \\
&+\frac{1}{18}\{2I_{(0,1,1)}(t) - 2I_{(1,0,1)}(t) + I_{(1)}(t)I_{(1,0)}(t) - I_{(1)}(t)I_{(0,1)}(t)\}[[X_0, X_1], X_1] \\
&+\frac{1}{36}\{I_{(0,0)}(t) + \{I_{(0)}(t)\}^2\}[[X_0, X_1], X_1] + \sum_{J;4\leq|J|} H^J(t)X^J,
\end{aligned}$$

where $H^J(t)$ is another version of $K^J(t)$ under each multi-index J, which is derived by transforming multiple Stratonovich integrals in $K^J(t)$ into Itô ones through Remark 2.3. Therefore, for each multi-index J, $H^J(t)$ is described as a polynomial function of multiple Itô integrals. In the next section, we will formulate the numerical schemes for the solution of SDE (2.18) on the basis of (2.16) with (2.26).

2.2. Mean-square order of multiple Wiener integrals. Now, in the remainder of this section, as a preparation for the error estimation for the new schemes in the next section, we will prove a proposition concerning the coefficients $H^J(t)$ of X^J for multiple indices J in (2.26). For this purpose, we first touch upon a lemma for multiple Itô integrals (2.20) proved by Kloeden and Platen (Lemma 5.7.5 in [15]). Let $E[\cdot|\mathcal{F}_s]$ be the conditional expectation with respect to a nondecreasing family of σ-subalgebra $\mathcal{F}_s$.

LEMMA 2.1. *For any* $\alpha = (j_1, \dots, j_k)$, $(j_i = 0, 1;\ i = 1, \dots, k)$, *and* $q = 1, 2, \dots,$

$$(2.27)\qquad E[|I_{(\alpha)}(\Delta t, s)|^{2q}|\mathcal{F}_s] = O((\Delta t)^{q(\ell(\alpha)+n(\alpha))}) \quad (\Delta t \downarrow 0),$$

where $\ell(\alpha)$ *and* $n(\alpha)$ *are the indices defined in Remark* 2.3*; that is,* $\ell(\alpha) =\{$*the number of elements of* $\alpha\}$ *and* $n(\alpha) = \{$*the number of* 0*'s in the elements of* $\alpha\}$.

Let $F(t, s)$ be a function of multiple stochastic integrals $I_{(\alpha)}(t, s)$. Suppose that

$$(2.28)\qquad E[\{F(\Delta t, s)\}^2|\mathcal{F}_s] = O((\Delta t)^m) \quad (\Delta t \downarrow 0)$$

holds. Then, in what follows, we call the real number m "*mean-square order (MSO)*" of $F(t, s)$.

From Lemma 2.1 we see that MSO of a multiple Itô integral $I_{(\alpha)}(t, s)$ is equal to $\ell(\alpha) + n(\alpha)$. Moreover, the following lemma shows that MSO of $I_{(\alpha)}(t, s)I_{(\beta)}(t, s)$ is given by $\ell(\alpha) + n(\alpha) + \ell(\beta) + n(\beta)$; that is, MSO of a product of multiple Itô integrals equals the sum of MSO of each of the stochastic integrals.

LEMMA 2.2. *For any multi-indices* α *and* β,

$$(2.29)\quad E[|I_{(\alpha)}(\Delta t, s)I_{(\beta)}(\Delta t, s)|^2|\mathcal{F}_s] = O((\Delta t)^{(\ell(\alpha)+n(\alpha)+\ell(\beta)+n(\beta))}) \quad (\Delta t \downarrow 0)$$

holds.

Proof. Through the Schwartz inequality, we obtain

$$E[|I_{(\alpha)}(\Delta t, s)I_{(\beta)}(\Delta t, s)|^2|\mathcal{F}_s] \leq \sqrt{E[\{I_{(\alpha)}(\Delta t, s)\}^4|\mathcal{F}_s]E[\{I_{(\beta)}(\Delta t, s)\}^4|\mathcal{F}_s]}.$$

The lemma is straightforwardly proved using this inequality and Lemma 2.1.

Remark 2.4. By Lemmas 2.1 and 2.2 together with Remark 2.3, we may verify that multiple Stratonovich integrals $J_{(\alpha)}(t,s)$ also satisfy the results in Lemmas 2.1 and 2.2. Hence, for a given α, we find that MSO of a multiple Stratonovich integral $J_{(\alpha)}(t,s)$ is also equal to $\ell(\alpha)+n(\alpha)$; that is, MSO of $J_{(\alpha)}(t,s)$ agrees with that of $I_{(\alpha)}(t,s)$ for the same multi-index α.

We now proceed to our goal. Using Lemmas 2.1 and 2.2, we can estimate MSO of each coefficient of X^J in (2.26) which is given by a polynomial function of multiple Itô integrals on $[0,t]$. For example, in the case of $|J|=1$, MSOs of the coefficients of X_0 and X_1, that is, MSOs of $I_0(t)$ and $I_1(t)$ are equal to 2 and 1, respectively. In the case of $|J|=2$, MSO of $I_{(0,1)}(t)$ or $I_{(1,0)}(t)$ in the coefficient of $[X_0,X_1]$ is given by 3, and thereby we can easily prove that MSO of the coefficient $(I_{(0,1)}(t)-I_{(1,0)}(t))/2$ itself is also equal to 3. In the same manner, we find MSOs of the coefficients of X^J when $|J|=3$; that is, MSOs of the coefficients of $[X_0,X_1],X_0]$ and $[X_0,X_1],X_1]$ are given by 5 and 4, respectively. Hence, in this case, *the least value* of MSOs of the coefficients of X^J equals 4. These facts suggest that we may verify the following proposition which we wish to establish.

PROPOSITION 2.1. *Suppose that k is a given integer greater than or equal to* 2, *and that the multi-indices J satisfy $|J|=k$; that is, $J=(j_1,\dots,j_k)$ ($j_i=0$ or 1; $i=1,\dots,k$). Then the least value of MSOs of the coefficients $H^J(t)$ of X^J in* (2.26) *is equal to $k+1$.*

Proof. We may prove this by induction. From the above examples, this proposition is obvious in the case of $|J|=2,3$. We assume that the assertion of this proposition holds for the case of $|J|=\ell(\geq 3)$. That is, the least value of MSOs of the coefficients $H^J(t)$ of X^J under $J=(j_1,\dots,j_\ell)$ in (2.26) equals $\ell+1$. Then, note that under the same J, the least value of MSOs of $K^J(t)$ in (2.25) agrees with that of $H^J(t)$, since $H^J(t)$ is only another version of $K^J(t)$ in terms of multiple Itô integrals. Now, let us consider the coefficients $K^{\tilde{J}}(t)$ of $X^{\tilde{J}}$ under $|\tilde{J}|=\ell+1$. On account of the definition of multiple integrals $W^{\Delta J}(t)$ (2.8) or (2.9) with (2.10) in the coefficients, one may obtain $K^{\tilde{J}}(t)$ by adding a stochastic integral with respect to dW or dt to each multiple integral in $K^J(t)$. Moreover, the following equations show that the MSOs for the integrals with increments dW and dt correspond to 1 and 2, respectively:

$$E\left[|\int_s^{s+\Delta t} dW(\tau)|^2|\mathcal{F}_s\right]=O(\Delta t),\quad E\left[|\int_s^{s+\Delta t} dt|^2|\mathcal{F}_s\right]=O((\Delta t)^2)\ (\Delta t\downarrow 0).$$

Hence, the least value of MSOs of $K^{\tilde{J}}(t)$ is equal to $\ell+2$, and thereby, the least value of MSOs of $H^{\tilde{J}}(t)$ is also so. Thus, the assertion in our proposition is proved. □

3. Composition methods for numerical integration of SDEs.

In this section, we will formulate some new stochastic numerical schemes for SDEs on the basis of composition methods and estimate the approximation errors for the schemes in the local and global senses.

3.1. Procedures of composition methods for SDEs.

We start with a numerical integration of the stochastic equation (2.18) on the discretized time series in the framework of the previous results on representation of solutions to SDEs. It adopts a uniform discretization of the time interval $[0,T]$ with stepsize

$$\Delta t=\frac{T}{N}$$

for fixed natural number N. Let $t_n = n\Delta t(n = 0, 1, 2, \dots, N)$ be the nth step-point. Then, for all $n \in \{0, \dots, N\}$, we abbreviate $S_n = S(t_n)$. Moreover, we use ΔW_n for $n = 0, 1, \dots, N$ to denote the increments $W(t_{n+1}) - W(t_n)$; they are independent Gaussian random variables with mean 0 and variance Δt, that is, $N(0, \Delta t)$-distributed random variables.

On account of (2.16), we may find the numerical solutions $S_n(n = 0, 1, \dots, N)$ to SDE (2.18) by using

$$S_{n+1} = \exp{(Y_{n\Delta t})}(S_n) \ (n = 0, 1, 2, \dots, N - 1), \tag{3.1}$$

formally, where $Y_{n\Delta t}$ is a vector field derived by replacing all the multiple Wiener integrals $I_{(\alpha)}(t) = I_{(\alpha)}(t, 0)$ in (2.26) by $I_{(\alpha)}(\Delta t, n\Delta t)$. In the following, $I_{(\alpha)}(\Delta t, n\Delta t)$ is denoted by $I_{(\alpha),n}(\Delta t)$. Moreover, we set $S_0 = S(0) = s_0$. According to the theory of ordinary differential equations, $\exp{(Y_{n\Delta t})}(\cdot)$ is often called the time-Δt map or exponential map. However, it is usually difficult to find the explicit form of the exponential map, and hence we need to build an approximation for (3.1).

To do this, we formulate a new stochastic numerical scheme as the following two procedures, which are composed of the truncation of the vector field (2.26) and a composition method (or operator splitting method) applied to the exponential map derived from the truncated vector field.

Procedure 1. For the vector field Y_t described by (2.26), we define a "truncated" vector field $\hat{Y}_t$ which is given by a truncation of the higher-order terms with respect to MSO of the coefficients of X^J in (2.26). Then, we define a numerical sequence $(\hat{S}_n)_{n=0}^N$ through

$$\hat{S}_{n+1} = \exp(\hat{Y}_{n\Delta t})(\hat{S}_n) \ (n = 0, 1, \dots, N - 1), \tag{3.2}$$

where $\hat{S}_0 = S(0) = s_0$.

Procedure 2. For $\hat{S}_{n+1} = \exp(\hat{Y}_{n\Delta t})(\hat{S}_n)$, we apply a composition method in a way analogous to that in the theory of ordinary differential equations. Suppose that the vector field $\hat{Y}_{n\Delta t}$ is of the form

$$\hat{Y}_{n\Delta t} = A_{n\Delta t} + B_{n\Delta t}, \tag{3.3}$$

where $\exp(A_{n\Delta t})$ and $\exp(B_{n\Delta t})$ can both be explicitly calculated through (2.13). Then an approximation to the exponential map of $\hat{Y}_{n\Delta t}$ is given by $\exp(A_{n\Delta t})\exp(B_{n\Delta t})$. Hence, the sequence of $(\hat{S}_n)_{n=0}^N$ in Procedure 1 is approximated by

$$\tilde{S}_{n+1} = \exp(A_{n\Delta t})\exp(B_{n\Delta t})(\tilde{S}_n) \ (n = 0, 1, \dots, N - 1), \tag{3.4}$$

where $\tilde{S}_0 = S(0) = s_0$.

We regard $(\tilde{S}_n)_{n=0}^N$ as a numerical approximation to the exact discretized solutions $(S_n)_{n=0}^N$.

3.2. Local error estimation for the new numerical scheme. We will estimate the approximation error of the numerical scheme described above. For this purpose, we first examine "local errors in the mean-square sense" for the scheme. Using this result, we will finally address the "global" error estimation in the next subsection.

We start with the definition of "local error order" in a manner analogous to that in [15].

Definition 3.1. *Suppose that $S(t)$ and $(\bar{S}_n)_{n=0}^N$ are an exact solution and the numerical approximation solutions to SDE* (2.18), *respectively. Let $E_{\tau,\xi}$ be the expectation conditioned on starting at ξ at time τ. Then the local error order α is defined by*

$$E_{t_n,s}[|\bar{S}_{n+1} - S_{n+1}|^2] = O((\Delta t)^{2\alpha}) \ (\Delta t \downarrow 0), \tag{3.5}$$

where $t_n = n\Delta t$ $(n = 0, \ldots, N-1)$, $S_n = S(n\Delta t)$, and $|\cdot|$ denotes the Euclidean norm on the space R^d. Note that the condition in the expectation (3.5) *means $S_n = \bar{S}_n = s$.*

We note that the accuracy of a numerical scheme improves with increasing local order.

Remark 3.1. In the framework of local error order defined by [23], the local order of $\bar{S}_n$ satisfying (3.5) is given by $2\alpha - 1$.

On the basis of this definition of local error estimation, we will investigate the error estimation of our approximation procedures. In what follows, we set $S_n = s$, where s is a given value.

Local error estimation for the truncation error in Procedure 1. First, we investigate the truncation error in Procedure 1. Let ${H_n}^J(\Delta t)$ be the coefficient of X^J in $Y_{n\Delta t}$ which is represented by a polynomial function of multiple Itô integrals for a given multi-index J as in (2.26).

Proposition 3.1. *Suppose that a truncation vector field $\hat{Y}_{n\Delta t}$ is given in the following form:*

$$\hat{Y}_{n\Delta t} = \sum_{J;1\le|J|\le\gamma} {H_n}^J(\Delta t)X^J. \tag{3.6}$$

That is, we assume the terms in $Y_{n\Delta t}$ satisfying $|J| \ge \gamma + 1$ are neglected. Then,

$$E_{t_n,s}[|\hat{S}_{n+1} - S_{n+1}|^2] = O((\Delta t)^{\gamma+2}) \ (\Delta t \downarrow 0). \tag{3.7}$$

Proof. In terms of Proposition 2.1, we can easily show that the least value of MSOs of ${H_n}^J(\Delta t)$ in the neglected terms equals $\gamma + 2$, since $|J| \ge \gamma + 1$. This fact, together with the definition of exponential map (see 2.11 with 2.13 [16], [3]), straightforwardly shows (3.7). Thus, we obtain the local order $(\gamma/2)+1$ for numerical approximation solutions $(\hat{S}_n)_{n=0}^N$ in the sense of Definition 3.1.

Remark 3.2. If the Lie algebra generated by X_0 and X_1 is of finite dimension, our error estimation derived above agrees with that of truncation of stochastic exponential maps by [2], since the convergence of (2.17), and hence (2.26), are actually guaranteed (cf. Remark 2.2). That is, under the assumption that such a convergence holds, the local error estimation derived above holds exactly. In general, however, our result may give only a formal error estimation. Indeed, according to [6], when the convergence is not guaranteed, the asymptotic expansion of stochastic exponential maps is estimated only in a "probability" sense.

Local error estimation for the composition scheme in Procedure 2. Next, we will proceed to the local error estimation for Procedure 2. We can carry this out using the BCH formula [3] together with Lemma 2.2; the formula is given by the following form:

$$\begin{aligned} \exp(\epsilon(\Delta t)X)\exp(\delta(\Delta t)Y) &= \exp(\epsilon(\Delta t)X + \delta(\Delta t)Y + \frac{1}{2}\epsilon(\Delta t)\delta(\Delta t)[X,Y] \\ &+ \frac{1}{12}(\epsilon(\Delta t)^2\delta(\Delta t)[X,[X,Y]] + \epsilon(\Delta t)\delta(\Delta t)^2[Y,[Y,X]]) + \cdots), \end{aligned} \tag{3.8}$$

where X and Y are C^∞ vector fields, and $\epsilon(\Delta t)$ and $\delta(\Delta t)$ are any functions of Δt; in our case, they correspond to polynomial functions of multiple Itô stochastic integrals $I_{(\alpha),n}(\Delta t)$.

PROPOSITION 3.2. *Let $\hat{Y}_{n\Delta t}$ be a truncated vector field given by* (3.6). *Suppose that the vector fields $A_{n\Delta t}$ and $B_{n\Delta t}$ in a decomposition* (3.3) *for $\hat{Y}_{n\Delta t}$ are described by*

$$A_{n\Delta t} = \sum_{J;1\le |J| \le \gamma} {F_n}^J(\Delta t) X^J, \quad B_{n\Delta t} = \sum_{J;1\le |J| \le \gamma} {G_n}^J(\Delta t) X^J, \tag{3.9}$$

respectively, and that the least values of MSOs of ${F_n}^J(\Delta t)$ and ${G_n}^J(\Delta t)$ in (3.9) *are given by α and β, respectively. If $X_A^{J_\alpha}$ and $X_B^{J_\beta}$, which are vector fields corresponding to the coefficients with α and β as MSO, respectively, satisfy $[X_A^{J_\alpha}, X_B^{J_\beta}] \neq 0$, then*

$$E_{t_n,s}[|\tilde{S}_{n+1} - \hat{S}_{n+1}|^2] = O((\Delta t)^{\alpha+\beta})) \ (\Delta t \downarrow 0). \tag{3.10}$$

Proof. Let ${F_n}^{J_\alpha}(\Delta t)$ and ${G_n}^{J_\beta}(\Delta t)$ be the coefficients in (3.9) whose MSOs are equal to α and β, respectively. Then, in an analogous way to that in Lemma 2.2, we can prove that

$$E_{t_n,s}[|{F_n}^{J_\alpha}(\Delta t){G_n}^{J_\beta}(\Delta t)|^2] = O((\Delta t)^{\alpha+\beta}) \ (\Delta t \downarrow 0). \tag{3.11}$$

Therefore, on account of the BCH formula (3.8), (3.11), and the definition of α and β given above, one may find that

$$\begin{aligned} E_{t_n,s}[|\tilde{S}_{n+1} - \hat{S}_{n+1}|^2] &= E_{t_n,s}[|\exp(A_{n\Delta t} + B_{n\Delta t})(s) - \exp(A_{n\Delta t})\exp(B_{n\Delta t})(s)|^2] \\ &= E_{t_n,s}\Bigg[\Bigg| \quad \exp(A_{n\Delta t} + B_{n\Delta t})(s) \\ &\qquad - \exp\left(A_{n\Delta t} + B_{n\Delta t} + \frac{1}{2}[A_{n\Delta t}, B_{n\Delta t}] + \cdots\right)(s)\Bigg|^2\Bigg] \\ &= O((\Delta t)^{\alpha+\beta}). \end{aligned} \tag{3.12}$$

Thus, the local order between $(\tilde{S}_n)_{n=0}^N$ and $(\hat{S}_n)_{n=0}^N$ is given by $(\alpha+\beta)/2$.

Remark 3.3. By further manipulating the BCH formula to eliminate higher-order terms, we can obtain various schemes which give higher-order approximations to the exponential map. For example, the scheme corresponding to "leapfrog," which is well known in deterministic numerical analysis, is given by

$$\exp((\Delta t)(X+Y)) = \exp\left(\frac{\Delta t Y}{2}\right)\exp(\Delta t X)\exp\left(\frac{\Delta t Y}{2}\right) + O((\Delta t)^3). \tag{3.13}$$

In a way analogous to that in (3.4), we define a stochastic leapfrog scheme as follows:

$$\tilde{S}_{n+1} = \exp\left(\frac{B_{n\Delta t}}{2}\right)\exp(A_{n\Delta t})\exp\left(\frac{B_{n\Delta t}}{2}\right)(\tilde{S}_n) \ (n = 0, 1, \ldots, N-1). \tag{3.14}$$

Then, using the BCH formula (3.8) and (3.11) repeatedly, we can find that the local error for this scheme is given by $(\alpha + 2\beta)/2$ as follows:

$$E_{t_n,s}[|\tilde{S}_{n+1} - \hat{S}_{n+1}|^2] = O((\Delta t)^{\alpha+2\beta}). \tag{3.15}$$

Thus, we can produce another numerical scheme having a better local order than that of the scheme (3.4). Moreover, using this scheme as a basis element for further leapfrog schemes, we may also produce an approximation to exponential map up to any order in a similar way to that in ordinary numerical analysis.

Total local error estimation for the numerical scheme of Procedures 1 and 2. Finally, we estimate the local error order between the exact discretized solutions $(S_n)_{n=0}^N$ and the numerical approximation solutions $(\tilde{S}_n)_{n=0}^N$. This is easily carried out by using Propositions 3.1 and 3.2 (or Remark 3.3) for the local orders in the above two procedures together with

$$E_{t_n,s}[|S_{n+1} - \tilde{S}_{n+1}|^2] \leq E_{t_n,s}[|S_{n+1} - \hat{S}_{n+1}|^2] + E_{t_n,s}[|\hat{S}_{n+1} - \tilde{S}_{n+1}|^2], \tag{3.16}$$

and thereby we obtain the following theorem.

THEOREM 3.1. *Under the conditions of Propositions* 3.1 *and* 3.2*,*

$$E_{t_n,s}[|S_{n+1} - \tilde{S}_{n+1}|^2] \leq O((\Delta t)^\delta) \ (\Delta t \downarrow 0) \tag{3.17}$$

holds, where $\delta = \min(\alpha + \beta, \gamma + 2)$ *in the case of* (3.4) *and* $\delta = \min(\alpha + 2\beta, \gamma + 2)$ *in the case of* (3.14)*.*

On account of Definition 3.1, if (3.17) holds, we may regard $(\delta/2)$ as a local order for the scheme giving the numerical approximation solutions $(\tilde{S}_n)_{n=0}^N$. In this article, however, we call the value a local order in a "weak sense" for the scheme, since the estimation of error order is "indirectly" derived from the inequality (3.16).

3.3. Global error estimation and some examples for the new schemes. Now, we proceed to global error estimation for our schemes. We start with the definition of "global error order" in a manner analogous to that in [15] mentioned in Definition 3.1.

DEFINITION 3.2. *Suppose that* $S(t)$ *and* $(\bar{S}_n)_{n=0}^N$ *are an exact solution and the numerical approximation solutions to SDE* (2.18)*, respectively. Moreover, let* $E_{0,\xi}$ *be the expectation conditioned on starting at* ξ *at "initial time"* τ*. Then the global error order* λ *is defined by*

$$E_{0,s}[|\bar{S}_N - S_N|^2] = O((\Delta t)^{2\lambda}) \ (\Delta t \downarrow 0), \tag{3.18}$$

where $S_N = S(N\Delta t)$*,* $N\Delta t = T$ *and* $|\cdot|$ *denotes the Euclidean norm on the space* R^d*.*

Note that the condition in the expectation (3.18) means $S_0 = \bar{S}_0 = s$. It is obvious that the accuracy of a numerical scheme improves with increasing global order.

Remark 3.4. In the framework of global error order defined by [23], the global order for the numerical solutions $\bar{S}_n$ satisfying (3.18) is given by 2λ.

Using Theorem 3.1, we can prove the following lemma and thereby find the global order of our schemes in a weak sense.

LEMMA 3.1. *Suppose that the numerical approximation solutions* $(\tilde{S}_n)_{n=0}^N$ *to SDE* (2.18) *given by* (3.4) *or* (3.14) *satisfy* (3.17)*; that is, the local order in a weak sense for the solutions is equal to* $(\delta/2)$*. Then*

$$E_{0,s}[|S_{n+1} - \tilde{S}_{n+1}|^2] \leq (n+1) \times O((\Delta t)^\delta) \ (\Delta t \downarrow 0). \tag{3.19}$$

Proof. Here, we treat only the case of $(\tilde{S}_n)_{n=0}^N$ given by (3.4) because we can prove (3.19) for the case of numerical solutions by (3.14) in a way analogous to that shown below.

We prove this lemma by induction. First, if $n = 0$, the inequality (3.19) is obvious, since it reduces to (3.17). We assume that (3.19) holds in the case of $n = k-1$. Then, for $n = k$, we rewrite the left-hand side of (3.19) as follows:

$$\begin{aligned}(3.20)\qquad & E_{0,s}[|S_{k+1} - \tilde{S}_{k+1}|^2] \\ &= E_{0,s}[|S_{k+1} - \exp(A_{(k+1)\Delta t})\exp(B_{(k+1)\Delta t})(S_k) \\ &\qquad + \exp(A_{(k+1)\Delta t})\exp(B_{(k+1)\Delta t})(S_k) - \tilde{S}_{k+1}|^2] \\ &\leq E_{0,s}[|S_{k+1} - \exp(A_{(k+1)\Delta t})\exp(B_{(k+1)\Delta t})(S_k)|^2] \\ &+ E_{0,s}[|\exp(A_{(k+1)\Delta t})\exp(B_{(k+1)\Delta t})(S_k) - \tilde{S}_{k+1}|^2],\end{aligned}$$

where $\exp(A_{(k+1)\Delta t})\exp(B_{(k+1)\Delta t})$ is the composition exponential map given by (3.4). Using (3.4) for $n = k$, that is, $\tilde{S}_{k+1} = \exp(A_{(k+1)\Delta t})\exp(B_{(k+1)\Delta t})(\tilde{S}_k)$, we may rewrite the first term on the right-hand side in (3.20) in the following form:

$$E_{0,s}[|S_{k+1} - \exp(A_{(k+1)\Delta t})\exp(B_{(k+1)\Delta t})(S_k)|^2] = E[|S_{k+1} - \tilde{S}_{k+1}|^2 | S_k = \tilde{S}_k].$$

This fact, together with the assumption about the local order for $(\tilde{S}_n)_{n=0}^N$, gives

$$E_{0,s}[|S_{k+1} - \exp(A_{(k+1)\Delta t})\exp(B_{(k+1)\Delta t})(S_k)|^2] \leq O((\Delta t)^\delta).$$

In a similar manner, using (3.4), we can put the second term on the right-hand side in (3.20) into the following form:

$$\begin{aligned}& E_{0,s}[|\exp(A_{(k+1)\Delta t})\exp(B_{(k+1)\Delta t})(S_k) - \tilde{S}_{k+1}|^2] \\ &= E_{0,s}[|\exp(A_{(k+1)\Delta t})\exp(B_{(k+1)\Delta t})(S_k - \tilde{S}_k)|^2].\end{aligned}$$

Then, the definition of exponential map [16], [3], together with (3.8) and the assumption of the induction, proves

$$E_{0,s}[|\exp(A_{(k+1)\Delta t})\exp(B_{(k+1)\Delta t})(S_k - \tilde{S}_k)|^2] \leq k \times O((\Delta t)^\delta).$$

Therefore, we find

$$(3.21)\qquad E_{0,s}[|S_{k+1} - \tilde{S}_{k+1}|^2] \leq (k+1) \times O((\Delta t)^\delta) \quad (\Delta t \downarrow 0),$$

and hence (3.19) holds for $n = k$, thus completing the proof. □

We note that $N\Delta t = T = constant$. Therefore, Lemma 3.1, together with this fact, straightforwardly proves the following theorem.

THEOREM 3.2. *Suppose that the local order in a weak sense for numerical approximation solutions* $(\tilde{S}_n)_{n=0}^N$ *given by* (3.4) *or* (3.14) *is equal to* $(\delta/2)$. *Then*

$$(3.22)\qquad E_{0,s}[|S_N - \tilde{S}_N|^2] \leq O((\Delta t)^{\delta-1}) \ (\Delta t \downarrow 0).$$

Thus, we can estimate the global error order for our schemes, although the estimation is indirectly derived through the inequality (3.22). Hence, on account of Definition 3.2, if (3.22) holds, we call the value $(\delta - 1)/2$ a global order in a weak sense for the schemes of (3.4) or (3.14).

Now, in the following, we will investigate some examples of new numerical schemes for (2.18), which are derived from the procedures given above, and estimate the local and global errors on the basis of Theorems 3.1 and 3.2.

Example 3.1. Suppose that a truncated vector field $\hat{Y}_{n\Delta t}$ in Procedure 1 is given by

$$\begin{aligned}\hat{Y}_{n\Delta t} &= I_{(0),n}(\Delta t)X_0 + I_{(1),n}(\Delta t)X_1,\\ &= \Delta t X_0 + \Delta W_n X_1.\end{aligned} \tag{3.23}$$

On account of (2.26), we see that γ in Proposition 3.1 for this truncated vector field equals 1. We further set $A_{n\Delta t} = \Delta t X_0$ and $B_{n\Delta t} = \Delta W_n X_1$ in the decomposition (3.3) and assume that the explicit forms of both exponential maps for them are obtained through (2.13). In this case, α and β in Proposition 3.2 become 2 and 1, respectively, because of Lemma 2.1. Then, the scheme (3.4) can be put into the following form.

Scheme 3.1.

$$\tilde{S}_{n+1} = \exp(\Delta t X_0)\exp(\Delta W_n X_1)(\tilde{S}_n). \tag{3.24}$$

Assume that $[X_0, X_1] \neq 0$. Then, through Theorems 3.1 and 3.2, we obtain

$$E_{t_n,s}[|S_{n+1} - \tilde{S}_{n+1}|^2] \leq O((\Delta t)^3) \tag{3.25}$$

and

$$E_{0,s}[|S_N - \tilde{S}_N|^2] \leq O((\Delta t)^2). \tag{3.26}$$

Thus, we find that the local order and the global order in a weak sense for Scheme 3.1 equals 1.5 and 1, respectively.

Example 3.2. For $\hat{Y}_n(\Delta t)$ in Example 3.1, we set $A_{n\Delta t} = \Delta t X^A{}_0 + \Delta W_n X^A{}_1$ and $B_{n\Delta t} = \Delta t X^B{}_0 + \Delta W_n X^B{}_1$ in (3.3), where $X_0 = X^A{}_0 + X^B{}_0$ and $X_1 = X^A{}_1 + X^B{}_1$. We assume that $[X^A{}_1, X^B{}_1] \neq 0$ and that the explicit forms of both exponential maps for them are obtained. In this case, α and β in Proposition 3.2 are both equal to 1; hence the local order of the scheme (3.4) for the above $A_{n\Delta t}$ and $B_{n\Delta t}$ becomes 1. In order to produce a scheme having better accuracy than that of this scheme, we use (3.14) instead of (3.4).

Scheme 3.2.

$$\tilde{S}_{n+1} = \exp\left(\frac{B_{n\Delta t}}{2}\right)\exp(A_{n\Delta t})\exp\left(\frac{B_{n\Delta t}}{2}\right), \tag{3.27}$$

where $A_{n\Delta t} = \Delta t X^A{}_0 + \Delta W_n X^A{}_1$ and $B_{n\Delta t} = \Delta t X^B{}_0 + \Delta W_n X^B{}_1$ under $X_0 = X^A{}_0 + X^B{}_0$ and $X_1 = X^A{}_1 + X^B{}_1$.

Then, Theorems 3.1 and 3.2 indicate that (3.25) and (3.26) also hold in this case. That is, the local error order and the global order of this scheme are equal to 1.5 and 1, respectively.

Example 3.3. We will formulate a scheme with better accuracy than that of the schemes described above. For this purpose, we choose the following vector field as $\hat{Y}_{n\Delta t}$ in (3.6):

$$\hat{Y}_{n\Delta t} = \Delta t X_0 + \Delta W_n X_1 + \frac{1}{2}(I_{(0,1),n}(\Delta t) - I_{(1,0),n}(\Delta t))[X_0, X_1]. \tag{3.28}$$

Then, from (2.26), we see that γ in Proposition 3.1 for this truncated vector field becomes 2. Moreover, we set

$$\exp(A_{n\Delta t}) = \exp(\Delta t X_0) \tag{3.29}$$

and

$$\exp(B_{n\Delta t}) = \exp\left(\Delta W_n X_1 + \frac{1}{2}(I_{(0,1),n}(\Delta t) - I_{(1,0),n}(\Delta t))[X_0, X_1]\right). \tag{3.30}$$

Assume that the explicit forms of both exponential maps for these are obtained through (2.13). In this case, Lemmas 2.1 and 2.2 show that α and β in Proposition 3.2 are equal to 2 and 1, respectively. Moreover, we adopt (3.14) for these vector fields which leads to the following scheme.

Scheme 3.3.

$$\tilde{S}_{n+1} = \exp\left(\frac{B_{n\Delta t}}{2}\right)\exp(A_{n\Delta t})\exp\left(\frac{B_{n\Delta t}}{2}\right)(\tilde{S}_n), \tag{3.31}$$

where $\exp(A_{n\Delta t})$ is given by (3.29) and $\exp(B_{n\Delta t}/2)$ is derived by replacing $B_{n\Delta t}$ by $B_{n\Delta t}/2$ in (3.30).

Then, because of Theorems 3.1 and 3.2, we find that

$$E_{t_n,s}[|S_{n+1} - \tilde{S}_{n+1}|^2] \le O((\Delta t)^4) \tag{3.32}$$

and

$$E_{0,s}[|S_N - \tilde{S}_N|^2] \le O((\Delta t)^3), \tag{3.33}$$

and hence conclude that the local order and the global order in a weak sense for Scheme 3.3 equals 2 and 1.5, respectively.

4. Examples.

In this section, we will give several illustrative examples of applying our new stochastic numerical schemes to SDEs.

4.1. Numerical simulation of a nonlinear asset price process in mathematical finance.

As was mentioned in section 1, we first work with the following nonlinear scalar SDE which is often treated as a model of an asset price process of Bessel type in mathematical finance [9] (cf. Remark 4.1):

$$dS(t) = S(t)dt + 2\sqrt{S(t)} \circ dW(t), \quad S(0) = s(> 0). \tag{4.1}$$

This system has a structure such that the value of solution remains nonnegative for any $t \in [0, T]$. In standard stochastic numerical schemes, however, this property is not always preserved numerically; in particular, if an initial value s is close to zero, the numerical solutions often go into the domain of negative values in the midst of numerical simulations. Such troublesome behavior will be observed in the numerical results given later. In contrast with this, through the results in previous section, we may obtain a scheme which leaves the structure of the stochastic system (4.1) invariant numerically. We will examine it now.

First, from (2.6) and (4.1), we see that the vector fields X_0 and X_1 become

$$X_0 = S\frac{d}{dS}, \; X_1 = 2\sqrt{S}\frac{d}{dS}, \tag{4.2}$$

respectively. Here, we note that $[X_0, X_1] = -X_1/2$, and the Lie algebra generated by X_0 and X_1 is of finite dimension. Hence, Remark 2.2 indicates that (2.26) actually converges in this case.

We proceed to investigate the application of Scheme 3.1 to SDE (4.1). On account of (3.24), we suppose that $A_{n\Delta t}$ and $B_{n\Delta t}$ in (3.3) are given by

$$A_{n\Delta t} = \Delta t X_0 = \Delta t S\frac{d}{dS}, \quad B_{n\Delta t} = \Delta W_n X_1 = \Delta W_n 2\sqrt{S}\frac{d}{dS}. \tag{4.3}$$

Then, in view of (2.13), we obtain the exponential maps for $A_{n\Delta t}$ and $B_{n\Delta t}$ explicitly as follows:

$$\exp(A_{n\Delta t})(s) = s\exp(\Delta t), \quad \exp(B_{n\Delta t})(s) = \{\Delta W_n + \sqrt{s}\}^2. \tag{4.4}$$

Inserting these into (3.24), we find that Scheme 3.1 applied to SDE (4.1) leads to

$$\tilde{S}_{n+1} = \left\{\Delta W_n + \sqrt{\tilde{S}_n}\right\}^2 \exp(\Delta t), \tag{4.5}$$

where $\tilde{S}_0 = S(0) = s$. Clearly, the numerical solutions derived from our scheme *never* take negative values, and this is just the result we want. Moreover, as mentioned in section 3.3, the local and global orders in a weak sense for this scheme are given by 1.5 and 1, respectively.

Next we will apply Scheme 3.3 to SDE (4.1). On account of (3.29), in this case, we also obtain $s\exp(\Delta t)$ as $\exp(A_{n\Delta t})(s)$. In contrast with this, (3.30) takes the form

$$\exp(B_{n\Delta t}) = \exp\left\{\left(\Delta W_n - \frac{1}{4}(I_{(0,1),n}(\Delta t) - I_{(1,0),n}(\Delta t))\right) X_1\right\}, \tag{4.6}$$

since $[X_0, X_1] = -X_1/2$. In a way similar to that of Scheme 3.1, this is also calculated explicitly as follows:

$$\exp(B_{n\Delta t})(s) = \left\{\Delta W_n - \frac{1}{4}(I_{(0,1),n}(\Delta t) - I_{(1,0),n}(\Delta t)) + \sqrt{s}\right\}^2, \tag{4.7}$$

and thereby we obtain the result of Scheme 3.3 applied to SDE (4.1) as

$$\tilde{S}_{n+1} = \left\{\frac{\Delta W_n}{2} - \frac{1}{8}(I_{(0,1),n}(\Delta t) - I_{(1,0),n}(\Delta t)) + \sqrt{\hat{s}}\right\}^2, \tag{4.8}$$

together with

$$\hat{s} = \left\{\frac{\Delta W_n}{2} - \frac{1}{8}(I_{(0,1),n}(\Delta t) - I_{(1,0),n}(\Delta t)) + \sqrt{\tilde{S}_n}\right\}^2 \exp(\Delta t), \tag{4.9}$$

where $\tilde{S}_0 = S(0) = s$. This also indicates that the numerical solutions derived from this scheme take only nonnegative values. Then, as mentioned in section 3.3, the local and global orders for this scheme are equal to 2 and 1.5, respectively.

Here, we will examine the numerical solutions of SDE (4.1) given by these schemes. Then, we will compare these numerical results with those of several standard numerical schemes. For this purpose, we adopt the Euler–Maruyama scheme (Taylor scheme of

Table 1

An example of numerical solutions to SDE (4.1) from the Euler–Maruyama scheme (4.10)*, Kloeden's Taylor scheme* (4.11)*, Scheme* (4.5) *and Scheme* (4.8) *with* (4.9) *for an initial value* $s = 0.01$.

n	Euler–Maruyama	Kloeden's Taylor	Scheme (4.5)	Scheme (4.8)
189	0.0373054	0.014018	0.0140095	0.0140059
190	0.0459159	0.0190609	0.0190539	0.0190461
191	0.0428106	0.0164982	0.01649	0.0164839
192	0.0524778	0.0223075	0.0222992	0.0222907
193	0.0365604	0.0126323	0.0126217	0.0126192
194	0.0190258	4.07906E-03	0.0040724	4.07239E-03
195	0.0120115	1.20801E-03	1.20498E-03	1.20484E-03
196	7.86052E-04	4.42925E-04	4.46361E-04	4.45822E-04
197	2.07421E-03	6.85707E-04	6.89868E-04	6.8875E-04
198	-9.47943E-05	7.45417E-05	7.31484E-05	7.32705E-05

global order 0.5) and Kloeden's Taylor scheme of global order 1.5 [15], [23]; they are given in the following forms for the SDE (4.1):

Euler–Maruyama scheme.

$$S_{n+1} = S_n + (S_n + 1)\Delta t + 2\sqrt{S_n}\Delta W_n. \tag{4.10}$$

Kloeden's Taylor scheme of global order 1.5.

$$\begin{aligned} S_{n+1} = S\ _n &+ (S_n + 1)\Delta t + 2\sqrt{S_n}\Delta W_n \\ &+ \{(\Delta W_n)^2 - \Delta t\} \\ &+ 2\sqrt{S_n} I_{(1,0),n}(\Delta t) + \sqrt{S_n} I_{(0,1),n}(\Delta t) \\ &+ \frac{1}{2}(S_n + 1)(\Delta t)^2. \end{aligned} \tag{4.11}$$

In the schemes (4.5), (4.8) with (4.9), (4.10), and (4.11), ΔW_n, $I_{(1,0),n}(\Delta t)$, and $I_{(0,1),n}(\Delta t)$ are numerically realized by the independent $N(0,1)$ random numbers γ_n and $\hat{\gamma}_n$ $(n = 0, 1, \ldots)$ as follows [15]:

$$\begin{aligned} \Delta W_n &= \gamma_n \sqrt{\Delta t}, \\ I_{(1,0),n}(\Delta t) &= \frac{1}{2}\left(\gamma_n + \frac{1}{\sqrt{3}}\hat{\gamma}_n\right)(\Delta t)^{3/2}, \\ I_{(0,1),n}(\Delta t) &= \frac{1}{2}\left(\gamma_n - \frac{1}{\sqrt{3}}\hat{\gamma}_n\right)(\Delta t)^{3/2}. \end{aligned} \tag{4.12}$$

Moreover, we choose $T = 1$ and $N = 1000$ here, and hence the stepsize $\Delta t = 10^{-3}$.

Tables 1 and 2 display the results of the numerical solutions from these schemes listed above for the initial value $s = 0.01$ and $s = 0.001$, respectively. (Note: in the case of Table 2, the scheme (4.10) is excluded.) Here we have used the same sequences of random numbers for each scheme together with (4.12). As was mentioned in the introductory part of this section, from these results we observe that the values of numerical solutions derived from the standard schemes become negative during their computation if their initial values are close to zero. In contrast with these results, all of our schemes are free from such problems. Thus, our scheme (4.5) and (4.8) with (4.9) are superior with respect to numerical realization of the character of SDE (4.1); that is, nonnegativity of solutions is preserved in contrast to the results of the standard schemes.

Table 2

An example of numerical solutions to SDE (4.1) *from Kloeden's Taylor scheme* (4.11)*, Scheme* (4.5) *and Scheme* (4.8) *with* (4.9) *for an initial value* $s = 0.001$.

n	Kloeden's Taylor	Scheme (4.5)	Scheme (4.8)
358	4.28647E-03	4.38672E-03	4.27267E-03
359	7.06959E-03	7.19804E-03	7.05171E-03
360	2.92672E-04	3.21066E-04	0.0002912
361	5.76257E-04	6.15458E-04	5.7371E-04
362	2.57058E-03	2.65334E-03	2.56494E-03
363	3.09359E-03	3.18363E-03	3.08695E-03
364	8.20704E-04	8.66758E-04	8.17091E-04
365	2.12819E-04	1.90873E-04	2.14876E-04
366	5.3766E-04	5.01956E-04	5.40526E-04
367	-2.97776E-07	7.52643E-07	7.81725E-10

Table 3

An example of numerical solutions to SDE (4.1) *from the Euler–Maruyama scheme* (4.10)*, Kloeden's Taylor scheme* (4.11)*, Scheme* (4.5) *and Scheme* (4.8) *with* (4.9) *for an initial value* $s = 1$.

n	Euler–Maruyama	Kloeden's Taylor	Scheme (4.5)	Scheme (4.8)
991	4.33234	4.29109	4.29002	4.29109
992	4.25444	4.21288	4.2118	4.21288
993	3.97593	3.9392	3.93813	3.9392
994	3.72468	3.69204	3.69095	3.69204
995	3.48053	3.4519	3.45081	3.4519
996	3.6467	3.61833	3.61729	3.61834
997	3.79678	3.76838	3.76736	3.76838
998	3.74771	3.71863	3.7176	3.71863
999	3.72407	3.69411	3.69307	3.69412
1000	3.62125	3.5914	3.59035	3.59141

Table 3 displays the results for the initial value $s = 1$. In Table 3, it is observed that numerical results of the schemes (4.5) and (4.8) are closer to those of Kloeden's scheme (4.11) than to the results of Euler's scheme (4.10). In particular, the numerical solutions from (4.8) are very similar to those of (4.11); from theoretical consideration of local and global orders in section 3.3, these observations are as expected.

Remark 4.1. Let us consider the following SDE:

$$dS(t) = S(t)dt + \frac{1}{1-\gamma}\{S(t)\}^{\gamma} \circ dW(t), \quad S(0) = s(> 0),$$

where $0 < \gamma < 1$. This is also often treated as a model of an asset price process in mathematical finance, and it is a generalization of (4.1). For this process, we can also construct numerical schemes like those described above in a similar way. Indeed, Scheme 3.1 for this SDE, of which local and global orders equal 1.5 and 1, respectively, is given by

$$\tilde{S}_{n+1} = [\{\Delta W_n + \tilde{S}_n^{1-\gamma}\}^2]^{1/\{2(1-\gamma)\}} \exp(\Delta t),$$

where $\tilde{S}_0 = S(0) = s$. Note that the numerical solutions derived from this scheme also satisfy nonnegativity.

4.2. Example of Scheme 3.2 for a nonlinear SDE. We study an example for Scheme 3.2 given by (3.27). Let us consider the following nonlinear scalar SDE:

$$dS(t) = S(t)dt + \{S(t) + 2\sqrt{S(t)}\} \circ dW(t), \quad S(0) = s(> 0). \tag{4.13}$$

In this case, the vector fields X_0 and X_1 are set as

$$X_0 = S\frac{d}{dS}, \qquad X_1 = (S + 2\sqrt{S})\frac{d}{dS}, \tag{4.14}$$

respectively. Then, we remark that $[X_0, X_1] = -\sqrt{S}(d/dS)$, $[X_0, [X_0, X_1]] = -[X_0, X_1]/2$, and $[X_1, [X_0, X_1]] = -[X_0, X_1]/2$ hold; hence the Lie algebra generated by X_0 and X_1 is of finite dimension. Therefore, as in section 4.1, (2.26) also actually converges in this case.

We may regard the SDE (4.13) as a linear SDE with the additional random perturbation $2\sqrt{S(t)} \circ dW(t)$. On account of this, as $A_{n\Delta t}$ and $B_{n\Delta t}$ in (3.27), we adopt

$$A_{n\Delta t} = \Delta t S\frac{d}{dS} + \Delta W_n S\frac{d}{dS}, \quad B_{n\Delta t} = \Delta W_n 2\sqrt{S}\frac{d}{dS}; \tag{4.15}$$

that is, we set $X_0^A = S(d/dS)$, $X_1^A = S(d/dS)$, $X_0^B = 0$, and $X_1^B = 2\sqrt{S}(d/dS)$. Then, in view of (2.13), we obtain the exponential maps for them explicitly as follows:

$$\exp(A_{n\Delta t})(s) = s\exp(\Delta t + \Delta W_n), \quad \exp(B_{n\Delta t})(s) = \{\Delta W_n + \sqrt{s}\}^2. \tag{4.16}$$

Inserting these equations into (3.27), we find Scheme 3.2 for the SDE (4.13); it is given by

$$\tilde{S}_{n+1} = \left\{\Delta W_n/2 + \sqrt{\{\Delta W_n/2 + \sqrt{\tilde{S}_n}\}^2 \exp(\Delta t + \Delta W_n)}\right\}^2, \tag{4.17}$$

where $\tilde{S}_0 = S(0) = s$. Then, the theoretical consideration of error stimation in section 3.3 proves that the local and global orders for this scheme equal 1.5 and 1, respectively.

Here, in a way similar to that in section 4.1, we will observe the results of numerical solutions through this scheme and compare them with the results of the following Euler–Maruyama scheme and Kloeden's Taylor scheme for (4.13):

Euler–Maruyama scheme.

$$S_{n+1} = S_n + \left\{\frac{3}{2}(S_n + \sqrt{S_n}) + 1\right\}\Delta t + (S_n + 2\sqrt{S_n})\Delta W_n. \tag{4.18}$$

Kloeden's Taylor scheme of global order 1.5.

$$\begin{aligned} S_{n+1} = S_n &+ \left\{\frac{3}{2}(S_n + \sqrt{S_n}) + 1\right\}\Delta t + (S_n + 2\sqrt{S_n})\Delta W_n \\ &+ \frac{1}{2}\left(S_n + 3\sqrt{S_n} + 2\right)\{(\Delta W_n)^2 - \Delta t\} \\ &+ \frac{3}{2}\left(S_n + \frac{5}{2}\sqrt{S_n} + 1\right) I_{(1,0),n}(\Delta t) + \frac{3}{2}\left(S_n + \frac{11}{6}\sqrt{S_n} + 1\right) I_{(0,1),n}(\Delta t) \\ &+ \frac{1}{6}\left(S_n + \frac{7}{2}\sqrt{S_n} + 3\right)\{(\Delta W_n)^3 - 3\Delta t\Delta W_n\} \\ &+ \frac{9}{8}\left(S_n + \frac{17}{12}\sqrt{S_n} + \frac{5}{6}\right)(\Delta t)^2. \end{aligned} \tag{4.19}$$

Finally, inserting (4.12) into the schemes (4.17)–(4.19), we obtain numerical solutions which are shown in Table 4 for the initial value $s = 1$, $T = 1$, $N = 1000$, and $\Delta t = 10^{-3}$.

TABLE 4

An example of numerical solutions to SDE (4.13) *from the Euler–Maruyama scheme* (4.18)*, Kloeden's Taylor scheme* (4.19) *and Scheme* (4.17) *for an initial value* $s = 1$.

n	Euler–Maruyama	Kloeden's Taylor	Scheme (4.17)
991	9.07412	9.83114	9.83382
992	9.77413	10.5897	10.5926
993	9.35542	10.1398	10.1426
994	9.79685	10.6085	10.6114
995	9.86074	10.6656	10.6686
996	9.91146	10.7086	10.7115
997	9.57355	10.3428	10.3456
998	9.27501	10.0183	10.021
999	9.64218	10.4042	10.407
1000	9.47959	10.2216	10.2244

From Table 4 it is also observed that the numerical results of the scheme (4.17) are closer to those of Kloeden's scheme than to the results of Euler's scheme. On account of local and global orders for each scheme, this is also to be expected.

4.3. Composition method applied to stochastic Hamiltonian dynamical systems. As mentioned in section 1, composition methods (or operator splitting methods) are not only a superior integration method for differential equations in preserving the special character or structure of the equations but also often useful for approximations of nonlinear equations whose solutions are not obtained explicitly. The results in sections 4.1 and 4.2 given above show that this is also true in the case of stochastic systems. As also mentioned, such an advantage is remarkable in the case of dynamical systems with multiple space dimensions or Hamiltonian dynamical systems using dimensional splitting methods (e.g., [25], [12]). To illustrate this, we will investigate numerical schemes by composition methods for stochastic dynamical systems with "Hamiltonian structure" [20], [21].

First we review stochastic Hamiltonian dynamical systems [20]. Let us consider the following 2ℓ-dimensional stochastic dynamical system:

$$(4.20)\qquad d\begin{pmatrix} x^i(t) \\ x^{\ell+i}(t) \end{pmatrix} = \begin{pmatrix} \partial_{\ell+i}H_0(x(t)) \\ -\partial_i H_0(x(t)) \end{pmatrix} dt + \begin{pmatrix} \partial_{\ell+i}H_1(x(t)) \\ -\partial_i H_1(x(t)) \end{pmatrix} \circ dW(t) \qquad (i = 1, \cdots \ell),$$

where $x = (x^k)_{k=1}^{2\ell}$ and $\partial_j = \partial/\partial x^j$ $(j = 1, 2, \ldots, 2\ell)$, respectively. In (4.20), $H_\alpha(x)$ $(\alpha = 0, 1)$ are smooth scalar functions on $R^{2\ell}$. Formally, one may regard this as a Hamiltonian dynamical system

$$\frac{d}{dt}\begin{pmatrix} x^i \\ x^{\ell+i} \end{pmatrix} = \begin{pmatrix} \partial_{\ell+i}\hat{H}(x) \\ -\partial_i \hat{H}(x), \end{pmatrix} \qquad (i = 1, \ldots, \ell)$$

with a "randomized" Hamiltonian $\hat{H}$ given by

$$\hat{H} = H_0 + H_1\gamma_t,$$

where γ_t is a one-dimensional Gaussian white noise. With these definitions, we call (4.20) and H_α $(\alpha = 0, 1)$ an (ℓ-dimensional) *stochastic Hamiltonian dynamical system* and the *Hamiltonian*, respectively.

Now, we proceed to an illustration of our new scheme for stochastic Hamiltonian systems. For simplicity, we set $\ell = 1$ and denote $x^1(t)$ and $x^2(t)$ by $q(t)$ and $p(t)$, respectively. Let us consider the class of Hamiltonian systems with the typical Hamiltonian $H_0 = p^2/2 + V_0(q)$ and $H_1 = p^2/2 + V_1(q)$, where $V_0(q)$ and $V_1(q)$ are any potential functions. With this notation, (4.20) becomes

$$d\begin{pmatrix} q(t) \\ p(t) \end{pmatrix} = \begin{pmatrix} p(t) \\ -V_0'(q(t)) \end{pmatrix} dt + \begin{pmatrix} p(t) \\ -V_1'(q(t)) \end{pmatrix} \circ dW(t). \tag{4.21}$$

In general, this is a stochastic nonlinear system. For this system, the vector fields X_0 and X_1 become

$$X_0 = p\partial_q - V_0'(q)\partial_p, \quad X_1 = p\partial_q - V_1'(q)\partial_p, \tag{4.22}$$

respectively.

We will apply our scheme to this system. As an important example of the decomposition of (3.3) for the above system, we choose the following splitting:

$$A_{n\Delta t} = p(\Delta t + \Delta W_n)\partial_q, \quad B_{n\Delta t} = -(V_0'(q)\Delta t + V_1'(q)\Delta W_n)\partial_p. \tag{4.23}$$

This corresponds to the decomposition mentioned in Scheme 3.2; that is, X_0^A, X_1^A, X_0^B, and X_1^B in Scheme 3.2 are given by $p\partial_q$, $p\partial_q$, $-V_0'(q)\partial_p$, and $-V_1'(q)\partial_p$, respectively. Then we note that $\exp(A_{n\Delta t})$ and $\exp(B_{n\Delta t})$ are exponential maps which correspond to the flows of solutions to the following SDEs, respectively:

$$d\begin{pmatrix} q(t) \\ p(t) \end{pmatrix} = \begin{pmatrix} p(t) \\ 0 \end{pmatrix} dt + \begin{pmatrix} p(t) \\ 0 \end{pmatrix} \circ dW(t),$$

$$d\begin{pmatrix} q(t) \\ p(t) \end{pmatrix} = \begin{pmatrix} 0 \\ -V_0'(q(t)) \end{pmatrix} dt + \begin{pmatrix} 0 \\ -V_1'(q(t)) \end{pmatrix} \circ dW(t).$$

Therefore, we can obtain the explicit forms of them; this may be regarded as an example of dimensional splitting. The results are given by

$$\exp(A_{n\Delta t})\begin{pmatrix} q_n \\ p_n \end{pmatrix} = \begin{pmatrix} p_n(\Delta t + \Delta W_n) + q_n \\ p_n \end{pmatrix},$$

$$\exp(B_{n\Delta t})\begin{pmatrix} q_n \\ p_n \end{pmatrix} = \begin{pmatrix} q_n \\ -(\Delta t V_0'(q_n) + \Delta W_n V_1'(q_n)) + p_n \end{pmatrix}.$$

Inserting these equations into (3.27), we finally find Scheme 3.2 for this system as follows:

$$\begin{pmatrix} \tilde{q}_{n+1} \\ \tilde{p}_{n+1} \end{pmatrix} = \begin{pmatrix} \hat{q}_n \\ -\frac{1}{2}(\Delta t V_0'(\hat{q}_n) + \Delta W_n V_1'(\hat{q}_n)) + \hat{p}_n \end{pmatrix}, \tag{4.24}$$

where

$$\begin{pmatrix} \hat{q}_n \\ \hat{p}_n \end{pmatrix} = \begin{pmatrix} \hat{p}_n(\Delta t + \Delta W_n) + \tilde{q}_n \\ -\frac{1}{2}(\Delta t V_0'(\tilde{q}_n) + \Delta W_n V_1'(\tilde{q}_n)) + \tilde{p}_n \end{pmatrix}. \tag{4.25}$$

As in the example in section 4.2, the local and global orders in a weak sense for this scheme are equal to 1.5 and 1, respectively. Thus, for the class of stochastic Hamiltonian dynamical systems with typical Hamiltonians mentioned above, we can numerically approximate them through our scheme (4.24) with (4.25) and achieve an accuracy corresponding to Taylor scheme of global order 1.

4.4. Remark on composition methods and conserved quantities in stochastic dynamical systems. Finally, we remark on numerical schemes for stochastic dynamical systems which preserve "conserved quantities" of the systems. It is well known that conserved quantities play an essential role to determine the structure of dynamical systems; hence, it is important to find a numerical scheme which has the conservation properties for the quantities related to stochastic systems. On the other hand, composition methods often give such schemes for deterministic systems. Therefore, we may expect that one may obtain such schemes through our results, which have the advantage of numerically preserving the conserved quantities for stochastic systems; and in the remainder of this section, we will briefly examine this feature of our schemes.

Let us consider d-dimensional stochastic dynamical systems (2.18). Suppose that a smooth function $I = I(S)$ satisfies

$$X_0 I = 0, \quad X_1 I = 0, \tag{4.26}$$

where X_0 and X_1 are the vector fields given by (2.6). According to [20], $I(S)$ becomes a constant quantity; that is, $I(S(t)) = \textit{constant}$ holds for the diffusion process $S(t)$ governed by SDE (2.18).

Under some conditions, we may straightforwardly formulate a stochastic scheme satisfying numerical preservation of conserved quantities. Assume that the exponential maps of $A_{n\Delta t} = \Delta t X_0$ and $B_{n\Delta t} = \Delta W_n X_1$ are explicitly calculated. Then, it is obvious that Scheme 3.1 preserves the conserved quantity I numerically because of the definition of exponential map and (4.26).

Now, we investigate a trivial example of a stochastic dynamical system with a conserved quantity and the numerical scheme through composition methods. Let us consider

$$d\begin{pmatrix} S^1(t) \\ S^2(t) \end{pmatrix} = \begin{pmatrix} S^2(t) \\ -S^1(t) \end{pmatrix} dt + \begin{pmatrix} S^2(t) \\ -S^1(t) \end{pmatrix} \circ dW(t); \tag{4.27}$$

this is a stochastic system with the conserved quantity $I(S) = \frac{1}{2}((S^1)^2 + (S^2)^2)$, since (4.26) holds. However, as mentioned in [21], ordinary schemes do not conserve $I(S)$ numerically. On the other hand, for this system, we adopt Scheme 3.1 with $A_{n\Delta t} = \Delta t X_0 = \Delta t(S^2\partial_1 - S^1\partial_2)$ and $B_{n\Delta t} = \Delta W_n X_1 = \Delta W_n(S^2\partial_1 - S^1\partial_2)$; then through (2.13), the numerical scheme is explicitly given by

$$\begin{pmatrix} \tilde{S}^1_{n+1} \\ \tilde{S}^2_{n+1} \end{pmatrix} = \begin{pmatrix} \cos(\Delta t) & \sin(\Delta t) \\ -\sin(\Delta t) & \cos(\Delta t) \end{pmatrix} \begin{pmatrix} \cos(\Delta W_n) & \sin(\Delta W_n) \\ -\sin(\Delta W_n) & \cos(\Delta W_n) \end{pmatrix} \begin{pmatrix} \tilde{S}^1_n \\ \tilde{S}^2_n \end{pmatrix}. \tag{4.28}$$

Therefore, for any n, the numerical solutions (4.28) for (4.27) satisfy $I(\tilde{S}^1_n, \tilde{S}^2_n) = \textit{constant}$. Thus, our scheme numerically preserves a conserved quantity I of the stochastic system (4.27), and this fact also shows the superiority of the scheme derived through composition methods.

5. Concluding remarks.

In this article, we have formulated composition methods for SDEs, and through these we have proposed some stochastic numerical schemes. Several illustrative examples show that the new schemes are superior in their conservation properties related to the character of SDEs, and they are useful for approximating the solutions to SDEs. Moreover, we have investigated local and global error orders for our schemes. Finally, we offer some remarks and note some future research problems related to this work.

(i) As mentioned in Remark 3.2, we plan to carry out a more detailed analytical error estimation for our schemes using the result on time asymptotics of exponential maps for SDEs by [6], since the stochastic series (2.26) is only a formal representation.

(ii) In our error estimation, we have addressed local and global error orders "in a weak sense" for our new schemes, since we have *indirectly* estimated the error orders for numerical solutions using these schemes. To obtain a precise local or global error order, it would be necessary to carry out a direct error estimation for such numerical solutions.

(iii) In this article, we have treated the SDEs with a one-dimensional Wiener process. However, it often happens that the error order of a numerical method collapses if there is more than one Wiener process. Hence, it would be important to investigate the error orders of our schemes in the case of SDEs with a multidimensional Wiener process. Moreover, in any approach to this problem, account should be taken of the remarkable work by K. Burrage and P.M. Burrage [4], [5]. In their papers, they have developed stochastic Runge–Kutta schemes of higher order for such SDEs through the Magnus formula related to Lie algebra. Therefore, based on their work, we should be able to improve our composition methods and offer new schemes with high order for SDEs with a multidimensional Wiener process.

(iv) It is to be noted that Li and Liu [18] and Kunita [17] have studied stochastic exponential maps for a more general class of stochastic processes (e.g., Lévy processes). Hence, using their results, it would be interesting to formulate stochastic composition methods for such general stochastic processes.

The research on these topics will be reported in future papers.

Acknowledgments. The author would like to thank Professor N. Nakamura, Professor M. Maejima, Professor Y. Miyahara, Professor S. Ogawa, and Professor A. Shimizu for useful discussions about the numerical approach to SDE (4.1). He also sincerely thanks Professor H. Kunita, Professor P. E. Kloeden, Professor K. Burrage, Professor T. Mitsui, Professor Y. Saito, and Professor M. Suzuki for their valuable suggestions and for making available their recent research.

REFERENCES

[1] L. Arnold, *Stochastic Differential Equations: Theory and Applications*, John Wiley and Sons, New York 1973.

[2] G. Ben Arous, *Flots et series de Taylor stochastiqes*, Probab. Theory Related Fields, 81 (1989), pp. 29–77.

[3] N. Bourbaki, *Lie Groups and Lie Algebras*, Springer-Verlag, Berlin, 1989.

[4] K. Burrage and P. M. Burrage, *High Strong Methods for Non-commutative Stochastic Ordinary Differential Equation Systems and the Magnus Formula*, Phys. D, 133 (1999), pp. 34–48.

[5] K. Burrage and P. M. Burrage, *Order Conditions of Stochastic Runge–Kutta Methods by B-Series*, SIAM J. Numer. Anal., 38 (2000), pp. 1626–1646.

[6] F. Castell, *Asymptotic expansion of stochastic flows*, Probab. Theory Related Fields, 96 (1993), pp. 225–239.

[7] E. Forest and R. Ruth, *Fourth-order symplectic integration*, Phys. D, 43 (1990), pp. 105–117.

[8] T. C. Gard, *Introduction to Stochastic Differential Equations*, Marcel Dekker, New York, 1988.

[9] H. Geman and M. Yor, *Bessel processes, Asian options and perpetuities*, Math. Finance, 3 (1993), pp. 349–375.

[10] D. Greenspan, *Conservative numerical methods for $\ddot{x} = f(x)$*, J. Comput. Phys., 56 (1984), pp. 28–41.

[11] N. Ikeda and S. Watanabe, *Stochastic Differential Equations and Diffusion Processes* 2nd edition, North-Holland/Kodansha, Amsterdam/Tokyo, 1989.

[12] A. Iserles, *Composite methods for numerical solution of stiff systems of ODE's*, SIAM J. Numer. Anal., 21 (1984), pp. 340–351.

[13] Y. Ishimori, *Explicit energy conservative difference schemes for nonlinear dynamical systems with at most quartic potentials*, Phys. Lett. A, 191 (1994), pp. 373–378.

[14] P. E. Kloeden and E. Platen, *A survey of numerical methods for stochastic differential equations*, Stochastic Hydrology and Hydraulics, 3 (1989), pp. 155–178.

[15] P. E. Kloeden and E. Platen, *Numerical Solution of Stochastic Differential Equations*, Springer-Verlag, Berlin, 1992.

[16] H. Kunita, *On the Representation of Solutions of Stochastic Differential Equations*, Lecture Notes in Math. 784, Springer-Verlag, Berlin, 1980, pp. 282–304.

[17] H. Kunita, *Asymptotic self-similarity and short time asymptotics of stochastic flows*, J. Fac. Sci. Univ. Tokyo, Sect. IA Math., to appear.

[18] C. W. Li and X. Q. Liu, *Algebraic structure of multiple stochastic integrals with respect to Brownian motions and Poisson processes*, Stochastics Stochastics Rep., 61 (1997), pp. 107–120.

[19] R. I. McLachlan, *On the numerical integration of ordinary differential equations by symmetric composition methods*, SIAM J. Sci. Comput. 16 (1995), pp. 151–168.

[20] T. Misawa, *Conserved quantities and symmetries related to stochastic dynamical systems*, Ann. Inst. Statist. Math., 51 (1999), pp. 779–802.

[21] T. Misawa, *Energy conservative stochastic difference scheme for stochastic Hamilton dynamical systems*, Japan J. Indust. Appl. Math., 17 (2000), pp. 119–128.

[22] Y. Moreau and J. Vandewalle, *A lie algebraic approach to dynamical system prediction*, in Proceedings of the 1996 IEEE International Symposium on Circuits and Systems, Atlanta, GA, 1996, pp. 182–185.

[23] Y. Saito and T. Mitsui, *Simulation of stochastic differential equations*, Ann. Inst. Statist. Math., 45 (1993), pp. 419–432.

[24] M. Suzuki, *General theory of higher-order exponential product formulas*, Phys. Lett. A, 146 (1990), pp. 319–324.

[25] N. N. Yanenko, *A difference method of solution in the case of the multidimensional equation of heat conduction*, Dokl. Akad. Nauk USSR, 125 (1959), pp. 1207–1210.

[26] H. Yoshida, *Construction of higher order symplectic integrators*, Phys. Lett. A, 150 (1990), pp. 262–269.

SIAM J. Sci. Comput.
Vol. 23, No. 3, pp. 891–909

MODIFICATION OF A FINITE VOLUME SCHEME FOR LAPLACE'S EQUATION[*]

N. B. PETROVSKAYA[†]

Abstract. For Laplace's equation, we discuss whether it is possible to construct a linear positive finite volume (FV) scheme on arbitrary unstructured grids. Dealing with the arbitrary grids, we state a control volume which guarantees a positive FV scheme with linear reconstruction of the solution. The control volume is defined by a property of the analytical solution to the equation and does not depend on the grid geometry. For those problems where the choice of the control volume is prescribed a priori, we demonstrate how to improve positivity of the linear FV scheme by using corrected reconstruction stencils. The difficulties arising when grids with no geometric restrictions are used for the discretization are discussed. Numerical examples illustrating the developed approach to the stencil correction are given.

Key words. Laplace's equation, finite volume scheme, positivity, stencil correction

AMS subject classifications. 65N12, 74S10, 35J05

PII. S1064827500368925

1. Introduction.

A discrete Laplace operator is often considered to be a good model for investigating a discretization of partial differential equations which contain diffusion operator $\nabla\cdot(D\nabla)$. Two important examples are given by convection-diffusion equations and Navier–Stokes equations with possible applications that include the problems of fluid dynamics, chemical engineering, and environmental pollution. For the numerical solution of these equations, what is desirable are discretization schemes which satisfy a discrete maximum principle (monotone schemes); otherwise one can expect strong oscillations or even divergency of the solution.

There are two possible approaches for development of monotone schemes on unstructured grids. The first approach is to use the grids with some geometric constraints on the triangulation. It is well known that triangulations with no obtuse triangles allow us to construct monotone schemes. The relevant examples are given in [6, 11]. However, nonobtuse triangulations can be used on very few practical problems. In the process of the grid generation it is often required to resolve some complicated features of the problem geometry that makes strict angle control to be difficult.

Looking for a wider class of triangulations, in the two-dimensional case a Delaunay triangulation [10] is very attractive. For Laplace's equation, in the two-dimensional case the Delaunay triangulation provides positivity (that guarantees the discrete maximum principle) of the linear finite element/finite volume scheme (Barth [3]). This important property may be applied for the solution of a wide range of problems, even more general than discretization of the Laplace equation. For instance, Xu and Zikatanov [17] developed a linear monotone finite element scheme for convection-diffusion equations in any spatial dimension. To obtain a positive discretization of the diffusion operator they assumed the restrictive geometrical conditions which in the two-dimensional case mean that the triangulation is a Delaunay triangulation.

[*] Received by the editors March 15, 2000; accepted for publication (in revised form) April 23, 2001; published electronically August 29, 2001. This research was supported by the Boeing Company under contract 104R and Russian Foundation for Basic Research grant N 01-01-06273.

http://www.siam.org/journals/sisc/23-3/36892.html

[†] Keldysh Institute of Applied Mathematics/Boeing Operations International, Inc., Gazetny per., 17/9, Moscow, 103009, Russia (natalia.petrovskaya@boeing.com).

Sakovich [16] used a Delaunay triangulation to control the grid quality for the construction of the monotone scheme for the system of two-dimensional conservation laws. In [16] the Laplace equation has been discretized first on the Delaunay grid. The coefficients of this discretization have then been exploited to obtain grid quality functional for the unstructured grids considered in the work.

Although giving us all advantages of using monotone schemes, for many complex geometries (e.g., multicomponent airfoil configuration) the Delaunay triangulation does not provide grids optimal in the sense of the accuracy of the solution. In particular, the Delaunay triangulation does not control the maximum angle, so that nearly collapsed triangles may appear as the result of grid generation (Barth [4]). On the other hand, for the equations mentioned above, grid adaptation to the solution usually results in non-Delaunay meshes, while fully automatic generation of adapted Delaunay grids is a technically difficult task for many practical applications. That is why another approach appears, where the grids with no geometric restrictions are allowed for the discretization. In the present work, an effort has been made to investigate whether it is possible to construct a linear positive finite volume (FV) scheme for the Laplace equation on arbitrary unstructured grids. We demonstrate the impact of grid geometry on the quality of the FV scheme and discuss to what extent arbitrary grids are good for the positive discretization.

When dealing with the arbitrary grid cells, it has been shown many times that for linear FV schemes a proper choice of the control volume allows us to improve the results of the discretization both on the structured and unstructured grids. Barth and Linton [5] successfully used the containment dual volume instead of the median dual on stretched triangulated quadrilateral grids to compute viscous flows. Having applied the viscous term discretization to the Laplace operator, Delanaye et al. [8] showed that a correction of the diamond-shaped control volume on Cartesian grids leads to the more positive scheme and obtained a robust discretization of the viscous terms in Navier–Stokes equations. Putti and Cordes [13] have proposed a modification of the control volume that allowed them to obtain the positive discretization of the Laplace equation on three-dimensional Delaunay meshes. In all these cases the choice of the control volume has been dictated by the geometry of grid cells.

In the present paper we show that it is possible to derive a convex control volume which guarantees a positive FV scheme for the Laplace equation and does not depend on the grid geometry. A property of the analytical solution to the equation is taken into account for the construction of the control volume. Since a maximum principle is exploited to obtain the solution on the central node of a cell, the produced FV scheme is entirely positive on any two-dimensional unstructured mesh.

For those practical applications where the geometry of the control volume is prescribed by some a priori conditions, we discuss how to improve the measures of positivity of the linear FV scheme on non-Delaunay meshes. Based on the analysis of the grid cell geometry, the stencils used for the linear reconstruction are corrected to obtain a more positive scheme. For some triangles, additional nodes of the triangulation are included into the stencils, while "far" stencil nodes generating negative scheme coefficients are eliminated from the discretization. This correction technique gives good results, producing a linear "quasipositive" FV scheme on the arbitrary meshes.

The results obtained in the work lead us to a better understanding of the difficulties one may expect discretizing the Laplace equation on arbitrary meshes. The limits of using grids with no geometric restrictions for the positive discretization are

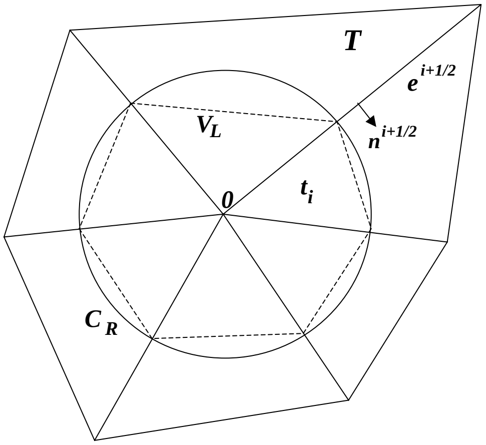


FIG. 1. *The geometry of an FV scheme for Laplace's equation.*

discussed in the conclusions.

2. Control volume for the discrete Laplace operator.

We consider Laplace's equation with Dirichlet boundary conditions in the unit square $\bar{\Omega}$:

$$\Delta u(x,y) \equiv \frac{\partial^2 u(x,y)}{\partial x^2} + \frac{\partial^2 u(x,y)}{\partial y^2} = 0, \quad (x,y) \in \Omega = (0,1) \times (0,1), \qquad (1)$$
$$u(x,y) = g(x,y), \quad (x,y) \in \partial\Omega.$$

Let a triangulation T be the union of all triangles t_i, $i = 1, \ldots, N$, incident to a central node 0 (see Figure 1). To obtain an FV discretization of (1) on node 0, the contour integral

$$\oint_{\partial V_{dual}} \frac{\partial u(x,y)}{\partial \mathbf{n}} dl = 0 \qquad (2)$$

is calculated over the edges of a control volume V_{dual} according to Green–Gauss theorem. For calculating $\frac{\partial u(x,y)}{\partial \mathbf{n}}$ =$(\nabla u, \mathbf{n})$ in (2) a linear reconstruction of the solution $u(x,y)$ in each triangle t_i is used:

$$u^i(x,y) = a_0^i + a_1^i x + a_2^i y. \qquad (3)$$

A standard stencil for the linear reconstruction in triangle t_i includes three nodes of the triangle. Expansion coefficients a_k^i, $k = 0,1,2$, in triangle t_i are calculated by the condition

$$u^i(x_k, y_k) = u(x_k, y_k) \equiv u_k.$$

After calculation $\frac{\partial u(x,y)}{\partial \mathbf{n}}$ and summation over all dual edges, the discrete Laplace operator on central node 0 is written as a weighted sum of nodal values u_n of the solution $u(x,y)$:

$$L(u) \equiv \sum_{n=0}^{N_T} \omega_n u_n = 0, \qquad (4)$$

where N_T is the number of the nodes of the triangulation. The requirement of non-negativity

$$\omega_0 < 0, \quad \omega_n \geq 0, \quad n = 1, \dots, N_T, \qquad \sum_{n=0}^{N_T} \omega_n = 0 \tag{5}$$

guarantees a maximum principle, which must hold for the discrete Laplacean (4) to provide stability and uniform convergence.

For the standard FV scheme (4) coefficients ω_n depend on what control volume V_{dual} is used for the discretization. A possible choice is a median dual or a centroid dual. The discretizations obtained over these control volumes are consistent and conservative, but they are not positive on arbitrary triangulations. Below we consider the following problem: *For arbitrary triangulation T, find a control volume V_L which provides conditions* (5) *for the scheme* (4).

Let $\mathbf{r}_0 = (x_0, y_0)$ be a radius-vector of central node 0: $u(\mathbf{r}_0) = u_0$. An analytical expression for the solution to the Laplace equation in point $\mathbf{r}_0$ is

$$u(\mathbf{r}_0) = \frac{1}{2\pi R} \int_{C_R} u(x,y)dl, \tag{6}$$

where C_R is the circumference of radius R, the circle center being point $\mathbf{r}_0$. Since triangulation T may contain boundary edges, the value R is restricted by the requirement $C_R \subset T$ to provide $u(x,y)$ being a harmonic function inside the circle C_R. Note that formula (6) expresses a maximum principle for Laplace's equation.

We calculate integral (6) assuming the linear reconstruction (3) of the solution. Substituting (3) into (6) and integrating over a circle arc A_i, we obtain the integral term for triangle t_i:

$$\begin{aligned}
\frac{1}{2\pi R} \int_{A_i} u(x,y)dl &= \frac{1}{2\pi R} \int_{A_i} (a_0^i + a_1^i x + a_2^i y)dl \\
= \frac{1}{2\pi R} &\int_{\phi_1^i}^{\phi_2^i} (a_0^i + a_1^i x_0 + a_2^i y_0 + a_1^i R\cos\phi + a_2^i R \sin\phi) R d\phi \\
= u_0 \frac{\delta\phi^i}{2\pi} &+ \frac{R}{2\pi}[a_1^i(\sin\phi_2^i - \sin\phi_1^i) - a_2^i(\cos\phi_2^i - \cos\phi_1^i)],
\end{aligned}$$

where ϕ_1^i and ϕ_2^i are the arc angles and $\delta\phi^i = \phi_2^i - \phi_1^i$. The summation over all triangles yields

$$u_0 = \frac{u_0}{2\pi} \sum_{i=1}^{N} \delta\phi_i + \frac{R}{2\pi} \sum_{i=1}^{N} [a_1^i(\sin\phi_2^i - \sin\phi_1^i) - a_2^i(\cos\phi_2^i - \cos\phi_1^i)].$$

Noting that $\sum_{i=1}^{N} \delta\phi_i = 2\pi$, the following condition for a discrete Laplace operator with the linear reconstruction appears:

$$\sum_{i=1}^{N} [a_1^i(\sin\phi_2^i - \sin\phi_1^i) - a_2^i(\cos\phi_2^i - \cos\phi_1^i)] = 0. \tag{7}$$

Since $\nabla u^i = (a_1^i, a_2^i)$ and $\mathbf{n}^{i+1/2} = (\sin\phi_2^i, -\cos\phi_2^i)$, where $\mathbf{n}^{i+1/2}$ is the unit normal vector for an interior edge $\mathbf{e}^{i+1/2}$, condition (7) takes the following form:

$$\sum_{i=1}^{N} (\nabla u^i, \mathbf{n}^{i+1/2} - \mathbf{n}^{i-1/2}) = 0. \tag{8}$$

Now let us consider a control volume V_L defined as a convex polygon created by connecting in every triangle t_i the endpoints of arc A_i (see Figure 1). For control volume V_L, the difference $\mathbf{n}^{i+1/2} - \mathbf{n}^{i-1/2}$ gives a vector, normal to dual edge $\mathbf{e}_L^i$. The unit outward normal vector $\mathbf{n}^i$ to edge $\mathbf{e}_L^i$ is calculated as

$$\mathbf{n}^i = \frac{R}{|\mathbf{e}_L^i|}(\mathbf{n}^{i+1/2} - \mathbf{n}^{i-1/2});$$

therefore, a flux Φ^i across edge $\mathbf{e}_L^i$ is

$$\Phi^i = |\mathbf{e}_L^i|(\nabla u^i, \mathbf{n}^i) = R(\nabla u^i, \mathbf{n}^{i+1/2} - \mathbf{n}^{i-1/2}).$$

Multiplying by R and taking into account that in the case of linear reconstruction (3) the sum (8) is the result of the exact integration, formula (8) transforms into the following:

$$\oint_{\partial V_L} \frac{\partial u(x,y)}{\partial \mathbf{n}} dl = 0. \tag{9}$$

The above results show us that the calculation of the solution on the central node of the triangulation by using formula (6) with condition (3) is equivalent to the linear FV discretization over control volume V_L. For the discretization over V_L, maximum principle (6) holds; therefore, scheme (4) is entirely positive on any two-dimensional unstructured mesh.

3. Correction of the stencils used for linear reconstruction. The construction of control volume V_L provides us with the positive discretization. However, on arbitrary grids the convex dual V_L is not consistent with the grid cell geometry, since gaps (or overlappings) appear in a convex dual mesh when stretched grid cells are considered. Concerning the issue of positivity, our next purpose is to discuss how far the geometry of the grid cells impacts on the discretization and whether it is possible to obtain the positive discretization (4) over the given control volume which completely covers domain Ω.

For the prescribed geometry of dual cell V_{dual} , the only way to render the fluxes in (2) closer to those in (9) is to change the approximation of the gradients. The following example illustrates the situation. Suppose a flux Φ_1 obtained as a result of the discretization on control volume V_L in triangle t_i is given as follows (index i is omitted):

$$\Phi_1 = (a_1 n_1 + a_2 n_2)|\mathbf{e}_L|.$$

Now consider a control volume V_{dual} different from V_L. Provided the same reconstruction (a_1, a_2) of the gradient is used, the discretization on control volume V_{dual} results in a flux Φ_2:

$$\Phi_2 = (a_1 n_1^{dual} + a_2 n_2^{dual})|\mathbf{e}_{dual}|,$$

where $\mathbf{n}^{dual} = (n_1^{dual}, n_2^{dual})$ is the unit vector, normal to control volume edge $\mathbf{e}_{dual}$ in triangle t_i. Evidently, if it is possible to find in every triangle t_i the new approximation of the gradient

$$\bar{a}_k = a_k \frac{n_k |\mathbf{e}_L|}{n_k^{dual}|\mathbf{e}_{dual}|} \quad k = 1, 2,$$

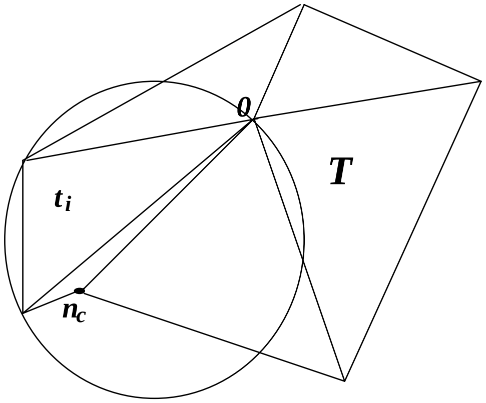


FIG. 2. *The Delaunay correction: implementation of a circumcircle test to include close nodes into the reconstruction stencil.*

then the positive scheme is guaranteed.

Below we consider modifications of a standard FV scheme obtained by including other nodes of the triangulation into the reconstruction stencils. In this case the following least squares problem is solved to calculate expansion coefficients a^i in triangle t_i:

$$\sum_{l=0}^{N_s}(u^i(x_l,y_l)-u(x_l,y_l))^2=\min, \tag{10}$$

where N_s is the number of stencil nodes, $N_s > 3$. Using extended stencils, it is possible to change the values of a^i in (2) and improve the discretization as a result. The crucial question for such a correction of the reconstruction stencils is how to select additional nodes.

Since a Delaunay triangulation provides a positive discretization on a dual cell different from V_L, the choice of convex polygon V_L as a control volume is not a necessary condition for the positive discretization. On the other hand, a simple linear search of the triangulation nodes shows that for many triangulations there is a variety of extended stencils providing a positive scheme. These facts allow us to suggest an existence of a more than one way for the stencil correction.

Unlike the linear reconstruction on a 3-point stencil, where it is possible to state exact conditions of positivity (cf. [3]), a strict geometric analysis of the scheme coefficients in the case of using the least squares method is very laborious and can hardly lead to production a fast and cheap algorithm for getting a positive scheme. That is why our present intention is not to obtain a rigorous mathematical formulation but to develop a reliable empirical approach to the selection of stencils.

The first possible way of stencil correction is based on the following hypothesis. Consider triangulation T shown in Figure 2. Since it is intuitively clear that it would be reasonable to include the closest to triangle t_i node n_c into the reconstruction stencil in the triangle, the following question arises:

Given triangulation T, differential operator $\mathcal{L}[u]$, and polynomial basis set $\{\Psi\}$ for a linear reconstruction, what is measure $M(T,\mathcal{L}[u],\{\Psi\})$ of proximity of stencil nodes? In other words, what nodes can be considered as being close to reconstruction stencil t_i?

Based on the theorem (Barth [3]), which states positivity of an FV scheme with a linear reconstruction on Delaunay triangulations, we suggest that for the Laplace operator the right answer to the above question is given by a circumcircle criterion which is the unique characterization of the Delaunay triangulation (Joe [10]). Namely, if node n_c belonging to T lies inside the circumcircle of given triangle t_i , it is considered as being close to the given triangle and should be included into reconstruction stencil t_i. In the general case, if the number k of triangulation nodes lie inside the circumcircle, they are included into the stencil for the reconstruction in triangle t_i by using the least squares method. This way of the stencil selection we call the Delaunay correction (DC).

For each grid node where the standard FV scheme is nonpositive, the algorithm for the Delaunay correction may be written as follows:

```
1. Define triangulation $T$ as a the union of all triangles $t_i$, $i = 1, \dots, N$, incident
   to the central node, and put all nodes of the triangulation (except of the central
   one) into array $T\_nodes$.
2. $For$ each triangle $t_i \subset T$, $Do$:
   2.1. Define the reconstruction stencil $S_i$ as a set of triangle's vertices.
   2.2. Define circumcircle $C_i$.
   2.3. Form array $marked\_nodes$ as a subset of the set $T\_nodes$:
      – $For$ each node $n_k$, $1 \leq k \leq N_T$, from the array $T\_nodes$ $Do$:
      – $If$ the circumcircle $C_i$ contains the node $n_k$ $And$ the node $n_k \notin S_i$ $Then$:
      – Add $n_k$ to the array $marked\_nodes$.
      – $EndDo$
   2.4. Include all nodes from the array $marked\_nodes$ into the reconstruction
        stencil $S_i$ by using the least squares method.
3. $EndDo$
```

Now let us discuss another type of the grid cell geometry. For triangulation T shown in Figure 3, there are no nodes obviously close to triangle t_i. On the contrary, node n_f is so far from others that if we were allowed to reconnect the nodes of the triangulation, it would be natural to connect node n_i with node n_l and remove n_f from T. Thus, another important for a proper correction of stencils question may be formulated as follows:

Given triangulation T, differential operator $\mathcal{L}[u]$, and basis set $\{\Psi\}$ for a linear reconstruction , what nodes have no effect on a discretization of $\mathcal{L}[u]$? In other words, what nodes in t_i can be considered as being far from any other reconstruction stencil?

To answer this question ("inverse" to the previous one), again consider formula (6) for solution u_0 on central node 0. We define the value R_{max} as the maximum radius which holds the requirement $C_R \subset T$ stated in the previous section (see Figure 3). Evidently, only those R which satisfy

$$R \leq R_{max} \tag{11}$$

should be considered in (6). Due to inequality (11) we conclude that for the discretization of the Laplace operator a characteristic size of the triangulation is not the maximum edge length but radius R_{max}. To make certain that far node n_f with the edge length $|\mathbf{e}_f| \gg R_{max}$ is odd for the scheme on node 0, let us consider standard FV scheme (4) on a median dual. According to [3], the coefficient ω_{n_f} corresponding to node n_f is calculated as

$$\omega_{n_f} = \frac{1}{2}\left(\frac{\sin(\alpha_i + \alpha_l)}{\sin(\alpha_i)\sin(\alpha_l)}\right), \tag{12}$$

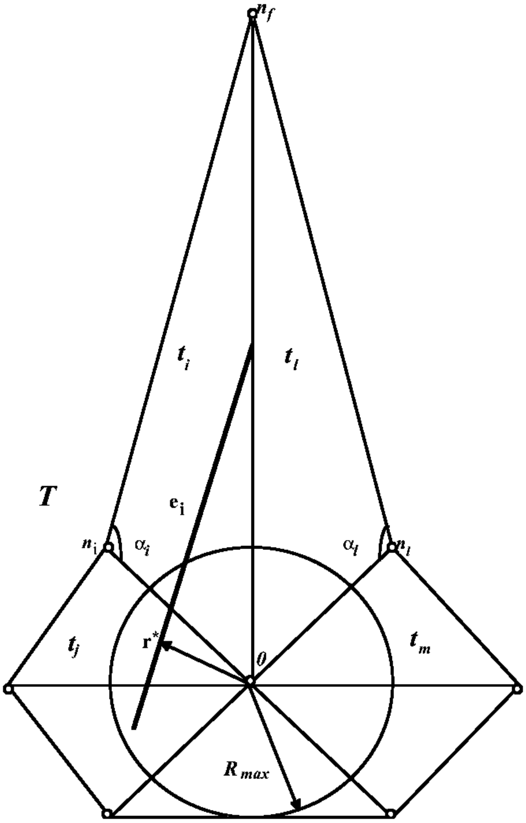


FIG. 3. *The gradient correction: elimination of a "far" node from the reconstruction stencil.*

where angles α_i and α_l are depicted in Figure 3. It can be seen from the figure that the more distant node n_f is from the central node of the triangulation, the more obtuse are the angles α_i and α_l. Obviously, the condition $\alpha_i + \alpha_l \leq \pi$, necessary for positive ω_{n_f}, fails for far node n_f.

To develop a correction technique for the stencils which contain node n_f, consider linear reconstruction (3) on a median control volume in triangle t_i. Since the gradient is constant in triangle t_i, the value of $\frac{\partial u}{\partial \mathbf{n}}$ in (2) is also constant at any point inside the given control volume in t_i

$$\frac{\partial u}{\partial \mathbf{n}} \approx a_1^i n_1^i + a_2^i n_2^i = \text{const} = C^i, \tag{13}$$

where $\mathbf{n}^i = (n_1^i, n_2^i)$ is the unit vector, normal to a control volume edge e_i. On the other hand, by definition,

$$\frac{\partial u}{\partial \mathbf{n}} = \lim_{\Delta t \to 0} \frac{u(\mathbf{r}_0 + \Delta t \mathbf{n}) - u(\mathbf{r}_0)}{\Delta t},$$

and we approximately calculate $\frac{\partial u}{\partial \mathbf{n}}$ at point $\mathbf{r}_0 \in t_i$ as

$$\frac{\partial u}{\partial \mathbf{n}} \approx \frac{u(\mathbf{r}^*) - u(\mathbf{r}_0)}{|\mathbf{r}^* - \mathbf{r}_0|}, \tag{14}$$

where $\mathbf{r}^* = (x^*, y^*)$ is an interior point of triangulation T, point $\mathbf{r}^*$ belongs to the line, normal to edge e_i, and $u(\mathbf{r}_0) = u_0$. The linear reconstruction at point $\mathbf{r}^*$ yields

$$u(\mathbf{r}^*) - u(\mathbf{r}_0) = a_0^* + a_1^* x^* + a_2^* y^* - u_0 = a_1^*(x^* - x_0) + a_2^*(y^* - y_0). \tag{15}$$

Since the components of the normal vector may be calculated as

$$n_1^i = \frac{x^* - x_0}{|\mathbf{r}^* - \mathbf{r}_0|}, \qquad n_2^i = \frac{y^* - y_0}{|\mathbf{r}^* - \mathbf{r}_0|},$$

we obtain from (14) and (15)

$$\frac{\partial u}{\partial \mathbf{n}} \approx a_1^* n_1^i + a_2^* n_2^i = \text{const} = C^*. \tag{16}$$

In "nice" triangle t_i point $\mathbf{r}^* \in t_i$; therefore,

$$a_1^* = a_1^*(u_0, u_1^i, u_2^i) = a_1^i, \qquad a_2^* = a_2^*(u_0, u_1^i, u_2^i) = a_2^i,$$

and the values C^i and C^* are the same. In "bad" triangle t_i point $\mathbf{r}^*$ belongs to another triangle t_j (see Figure 3), where the gradient depends on the values u_1^j and u_2^j: $\boldsymbol{a}^* = (a_1^*, a_2^*) = \boldsymbol{a}^*(u_0, u_1^j, u_2^j) = \boldsymbol{a}^j \neq \boldsymbol{a}^i$. In this case, when formula (13) is used for calculating $\frac{\partial u}{\partial \mathbf{n}}$, a nonphysical flux across edge e_i appears:

$$\begin{aligned}\left(\frac{\partial u}{\partial \mathbf{n}}\right)_{false} = C^* - C^i &= (a_1^* n_1^i + a_2^* n_2^i) - (a_1^i n_1^i + a_2^i n_2^i) = (a_1^j - a_1^i) n_1^i + (a_2^j - a_2^i) n_2^i \\ &= (\boldsymbol{\nabla} u_{false}, \mathbf{n}^i),\end{aligned}$$

where the false gradient $(\boldsymbol{\nabla} u)_{false}$ is defined as the difference between the gradients in triangles t_j and t_i, respectively. To improve the situation when false gradients appear, we suggest in "bad" triangle t_i to include stencil t_j into the reconstruction in t_i by using the least squares method. Another even more radical way is to change the stencil t_i by stencil t_j. These corrections change the value of the gradient in triangle t_i that may decrease the nonphysical flux. We refer to such a correction technique as the gradient correction (GC). Numerical experiments show that for geometry T from Figure 3 the gradient correction results in the positive scheme when the stencil points from triangles t_j and t_m are captured to form extended stencils for triangles t_i and t_l, respectively.

The idea of GC correction leads us to the following algorithm.

```
1. Define triangulation $T$ and all control volume edges $e_i$, $i = 1, \dots, N$.[1]
2. *For* each triangle $t_i \subset T$, *Do*:
    2.1 Define the reconstruction stencil $S_i$ as a set of triangle's vertices.
    2.2 Define point $\mathbf{r}^*$ as a point of intersection between the line $e_i$ and the perpendicular dropped to the line $e_i$ from the central node of the triangulation.
    2.3 Find a triangle $t_j$ point $\mathbf{r}^*$ belongs to.
    2.4 If $j \neq i$, include nodes of the triangle $t_j$ into the reconstruction stencil $S_i$ by using the least squares method.
3. *EndDo*
```

[1] According to Green–Gauss theorem, it is possible to consider a segment e_i created by connection of the edge midpoints (see Figure 3) instead of treating two median segments in each triangle.

Concerning practical realization of the suggested correction technique, there are still some open questions. Thus for the Delaunay correction, it is unclear whether or not the nodes which lie on a circumcircle boundary should be included into the reconstruction. Similar questions arise in the implementation of the GC. For instance, in the case when point $\mathbf{r}^*$ comes to an edge of the triangulation it is possible to include either one of the adjacent triangles or both of them into the stencil. Also, point $\mathbf{r}^*$ may coincide with a grid node n_g or lay outside domain $\bar{\Omega}$.

Since there is no strict mathematical foundation of the suggested algorithms, only practical recommendations can be given for correct treatment of these cases. One may find extended discussion based on our numerical experience with the correction algorithms in [12].

To conclude this section, let us make some remarks on the possible implementation of the suggested algorithms. Considering the discretization of a given differential operator on unstructured grids where stretched cells can appear, a most important problem is how to indicate cells which are "bad" for the discretization. Although the strict mathematical conditions based on error estimation are obtained for some important cases (Babuška and Aziz [1]), and stretched triangular cells proved to not always be bad (Rippa [14]), the general concept of "bad" or "nice" triangles requires further analysis, its formulation depending upon what differential operator is considered and what discretization method is used. For the FV discretization of the Laplace equation, the developed technique provides us with a kind of an empirical indicator of the "bad"/"nice" triangle, as those triangles where the stencil correction is needed may be considered as the "bad" ones.

4. Numerical results. The aim of this section is to present numerical validation of the suggested ways of the stencil correction. Unfortunately, by now we are not able to formulate precisely under what conditions each correction algorithm should be implemented. That is why a combined approach is used to correct scheme stencils. For those nodes where the standard FV scheme (4) is nonpositive, both DC and GC scheme stencils are constructed. Then the stencil providing the least nonpositive scheme is selected. This procedure may appear rather costly, but keeping in mind our present purpose we do not discuss here a computational efficiency of the developed algorithm.

To assess nonpositivity of the scheme coefficients we use a simple criterion taken from the work (Coirier [7]). Let us rewrite (4) as

$$u_0 = \sum_{n=1}^{N_T} \alpha_n u_n,$$

where $\alpha_n = -\omega_n/\omega_0$. Then the value α_{min}

$$\alpha_{min} = \frac{\min_{1\leq n\leq N_T}(\alpha_n, 0)}{\sqrt{\sum_{n=0}^{N_T} \frac{\alpha_n^2}{N_T}}} \tag{17}$$

is a measure of positivity[2] of the scheme coefficients for the given node u_0. For the nonpositive function α_{min} defined on the grid nodes we introduce the following

[2] The definition of α_{min} formally misses the case when $\omega_n < 0 \ \forall n \neq 0$, $\omega_0 > 0$. In practice, this case should be treated separately as it indicates strong degeneration of an FV volume cell. The consideration of this situation is beyond the scope of the paper.

Table 1

Positivity measures for standard (FVS) and corrected (CS) FV schemes with linear reconstruction.

Grid	$\alpha^{L_1}_{min}$		α^{C}_{min}	
	FVS	CS	FVS	CS
$G1$	-0.191	$-9.6 \cdot 10^{-3}$	-1.355	-0.354
$G2$	-0.126	$-6.9 \cdot 10^{-3}$	-3.593	-0.411
$G3$	-0.437	$-2.42 \cdot 10^{-2}$	-1.696	-0.377
$G4$	-0.188	$-4.2 \cdot 10^{-3}$	-1.587	-0.241
$G5$	-0.416	$-3.45 \cdot 10^{-2}$	-1.438	-0.412
$G6$	-0.485	$-7.2 \cdot 10^{-3}$	-1.602	-0.286

quantity:

$$\alpha^{L_1}_{min} = \frac{\sum_{m=0}^{N_g} \alpha^{m}_{min}}{N_g},$$

where N_g is the number of mesh nodes. The value $\alpha^{L_1}_{min}$, as well as α^{C}_{min}, where

$$\alpha^{C}_{min} = \min_{m}\{\alpha^{m}_{min}\},$$

are used to estimate nonpositivity of the discretization over the whole mesh. Evidently, $\alpha^{L_1}_{min} = 0$ for an entirely positive scheme.

What are reasonable values for the parameter α_{min}? Coirier [7] has investigated a number of different stencils for an FV discretization of the Laplace equation on adaptive Cartesian grids for further implementing to the Navier–Stokes equations. It was found that schemes with $\alpha^{L_1}_{min} \sim -10^{-1} \div -10^{-2}$ are acceptable for calculating a low Reynolds number laminar flow. At the same time the discretization with $\alpha^{L_1}_{min} \sim -1.0$ proved to be divergent. Delanaye et al. [8] have also considered a discrete Laplacean on Cartesian meshes. They discovered that the value $\alpha^{L_1}_{min} = -1.62$ leads to the loss of the scheme stability. After stencil correction a new value $\alpha^{L_1}_{min} = -0.366$ provided convergence to the solution. Based on these results, in our work we consider stencils with $\alpha^{L_1}_{min} \sim -1.0$ to be strongly nonpositive.

Our first numerical experiment is to verify that the corrected scheme exhibits better measures of positivity. A number of grids with various geometries of grid cells have been generated to test suggested correction algorithms. Generating these "bad" grids, the main requirement was to produce the most possible number of non-Delaunay cells, where a standard FV scheme is nonpositive. The grids are shown in Figures 4 and 5.

To calculate the positivity measures, the Laplace equation is discretized in the unit square with the following boundary conditions:

$$u(x,0) = x^2, \quad u(0,y) = y^2, \quad u(x,1) = x^2 - 1, \quad u(1,y) = 1 - y^2. \tag{18}$$

The values of $\alpha^{L_1}_{min}$ and α^{C}_{min} for the standard FV scheme and the corrected scheme calculated on the generated grids are shown in Table 1. It can be seen from the table that the corrected scheme is much more positive than the standard one. Examples of function $\alpha_{min}(x_i, y_i)$ for the standard and the corrected schemes on some of generated grids are shown in Figure 5.

Grid G2 (see Figure 4) gives us a nice example of how the GC algorithm treats stretched cells. Fans on the grid are generated using the "torture test" idea (GGNS

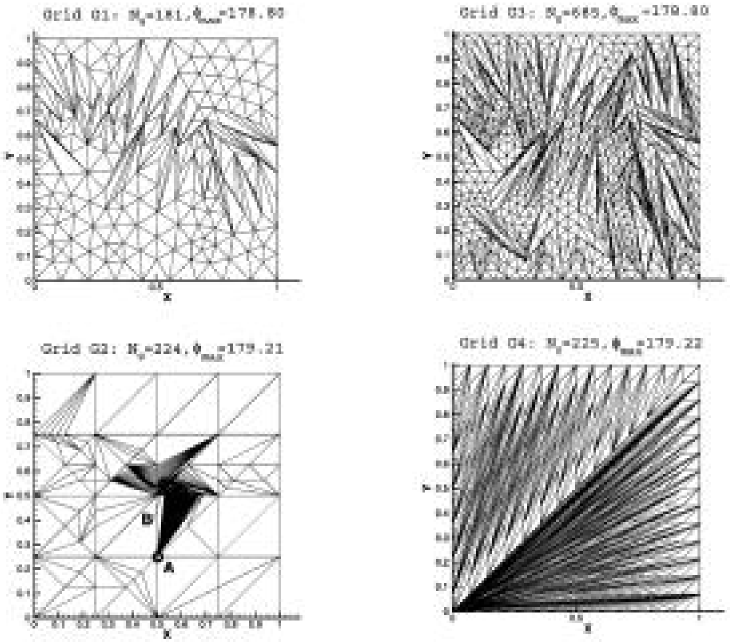

Grid G1: N_g=181, φ_max=178.80
Grid G3: N_g=685, φ_max=179.80
Grid G2: N_g=224, φ_max=179.21
Grid G4: N_g=225, φ_max=179.22
A
B
x
Y

FIG. 4. *Examples of "bad" grids used to estimate positivity measures of the corrected FV scheme; N_g is the number of the grid nodes, ϕ_{max} is the maximum grid angle (in degrees).*

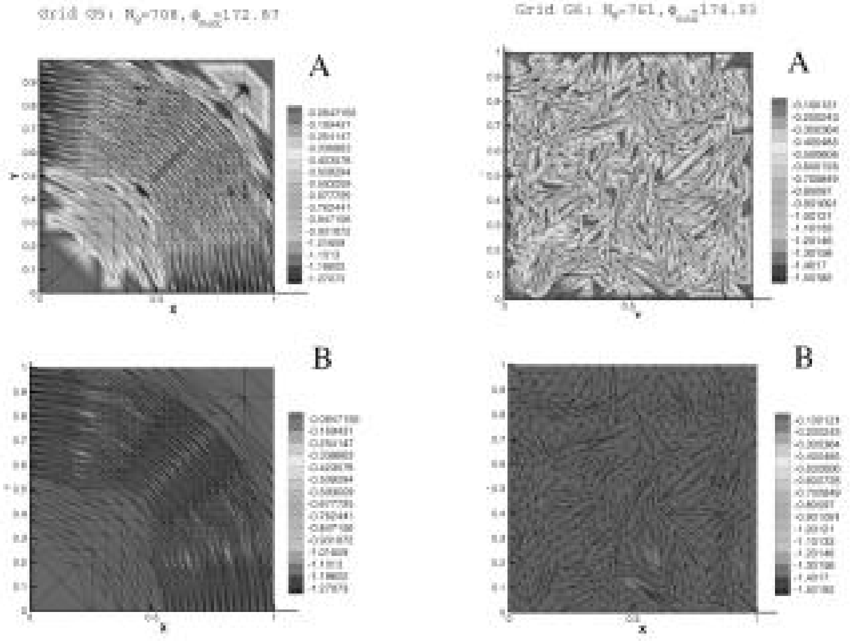


FIG. 5. *Function $\alpha_{min}(x, y)$ for (*A*) the standard and (*B*) the corrected FV scheme with the linear reconstruction.*

team [9]). In this test the following procedure has been implemented to generate a grid with large angles. At each step of the grid generation, the boundary problem (e.g., the convection-diffusion equation with Dirichlet boundary conditions in the considered example) is solved numerically. Given the numerical solution, a new grid node is always placed in the cell with the maximum solution error and the old triangle is subdivided into three triangles to increase the maximum angle in the new triangles. Then the boundary problem is solved again over the new grid, and the above procedure is repeated until the required value of the maximum angle is reached.

As a result of the grid generation, each of the triangulations with central nodes A and B shown in Figure 4 comprises 85 nodes. The standard FV discretization of the boundary problem (18) on median control volume produces the extremely nonpositive scheme on nodes A and B : $\alpha_{min}(A) = -3.498$, $\alpha_{min}(B) = -3.593$. For triangulations A and B, the gradient correction not only renders the scheme positive ($\alpha_{min}(A) = \alpha_{min}(B) = 0.0$) but also crucially transforms the scheme stencils N_s^A and N_s^B. New stencil N_s^A for the scheme in node A includes seven nodes while new stencil N_s^B includes only five nodes.

Our next test is to compare convergence to the exact solution for the standard and the corrected schemes. Due to the strong distortion of the finite volume cells the "bad" grids generated in the previous test exhibit poor approximation properties that makes it difficult to assess the convergence rate. That is why for the convergence test we generate a sequence of model meshes as follows. First, cells of a uniform Cartesian grid are cut by two diagonals. Then, the central node in each Cartesian cell is moved down vertically to increase the angle ϕ_{max} corresponding to the central node. Thus a parametric family of meshes with different geometry of stencils can be obtained, the maximum grid angle ϕ_{max} ($\pi/2 \leq \phi_{max} < \pi$) being a controlling parameter. The angle $\phi_{max} = \pi/2$ determines the standard grid considered in the previous test and provides positive standard FV discretization. For any angle $\phi_{max} > \pi/2$ the standard FV scheme is nonpositive.

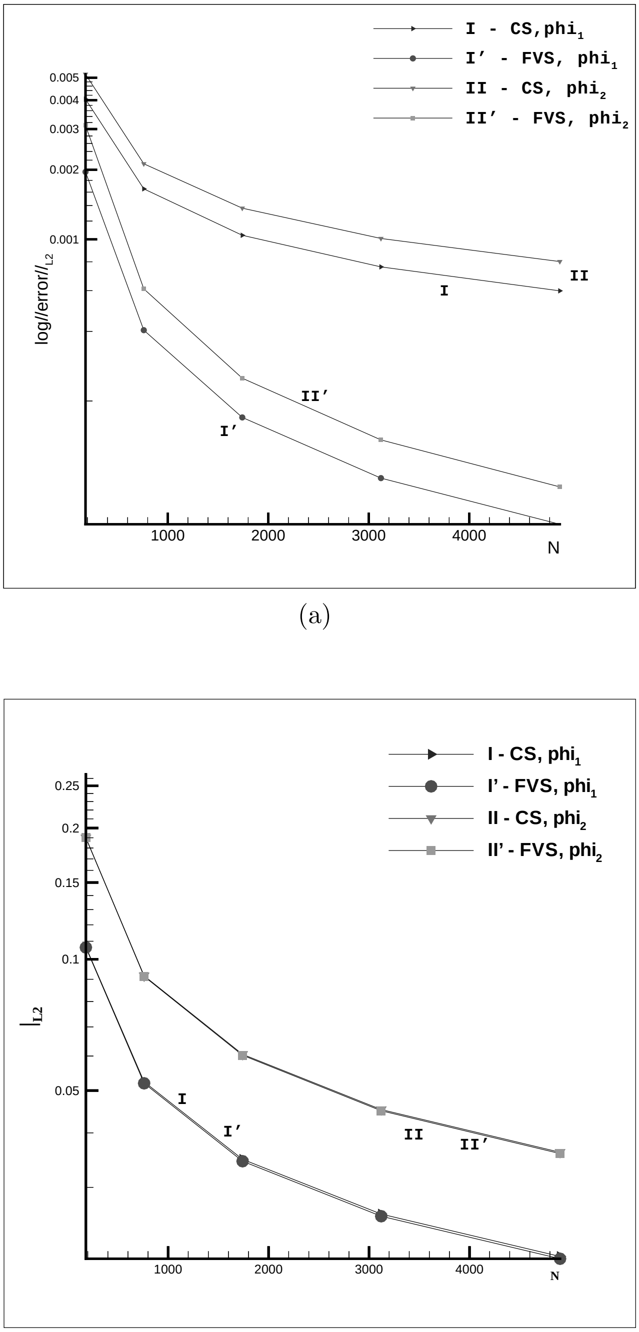
For the convergence test we solve the following boundary problem in the unit square:

$$u(x,0) = \cos(\omega x), u(0,y) = \exp(\omega y), u(x,1) = \cos(\omega x)\exp(\omega), u(1,y) = \cos(\omega)\exp(\omega y). \tag{19}$$

The analytical solution to the problem is

$$u(x,y) = \cos(\omega x)\exp(\omega y). \tag{20}$$

The numerical solution is calculated for the value $\omega = -5.0$. The convergence results for both the standard FV scheme (FVS) and the corrected scheme (CS) are plotted in Figure 6. The error measured in the L^2-norm is shown in the semilogarithmic scale. The value of ϕ_{max} has been varied to study how the convergence rate depends on the maximum grid angle. Figure 6(a) shows the convergence history for the solution on grids with $\phi_{max} \equiv \phi_1 = \frac{2}{3}\pi$ (curves I and I' in the figure for the CS and FVS, respectively). Curves II and II' in the figure present the convergence results for the value $\phi_{max} \equiv \phi_2 = \frac{5}{6}\pi$. In both cases the corrected scheme converges, although the rate of the convergence is slower in comparison with that for the standard scheme. Let us note that for the corrected scheme the approximation over nonsymmetric stencils may impact on the convergence rate as well as geometric degeneration of the finite volume cells. These two factors may slow down the convergence rate on the grids with large values of angle ϕ_{max}.



(a)

(b)

FIG. 6. *Convergence test problem* (19), (20). *The convergence history of* (a) *the solution and* (b) *the gradient for the corrected (curves* I, II*) and the standard (curves* I', II'*) schemes.*

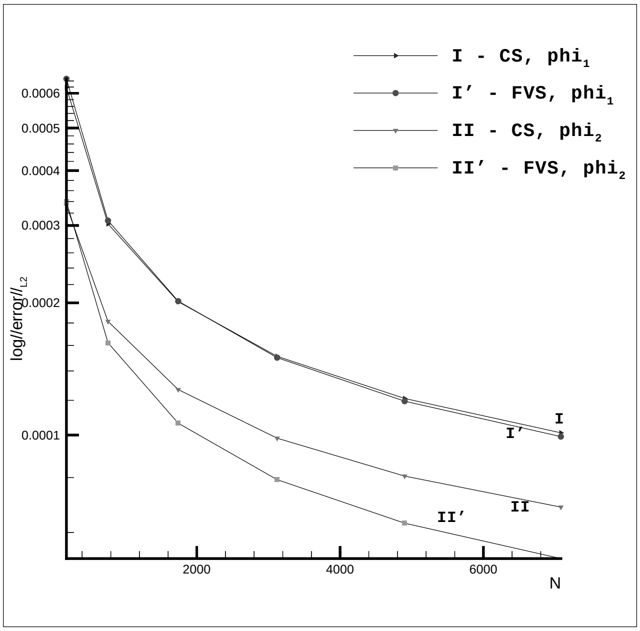


FIG. 7. *Convergence test problem with discontinous boundary conditions* (21), (22). *The convergence history of the solution for the corrected (curves* I, II*) and standard (curves* I', II'*) schemes.*

The convergence history for the gradient obtained on grids with the same values of ϕ_{max} is shown in Figure 6(b). As can be seen from the figure, the gradient convergence results are almost the same for both schemes.

For the function $u(x, y)$ considered in the test, the effect of the triangle geometry on approximation is that increasing the angle ϕ_{max} leads to the slower convergence rate for both standard and corrected schemes. In our numerical experiments we have obtained the slowest convergence rate for the extreme case of $\phi_{max} = 0.99\pi$.

Now we consider the problem with discontinuous boundary conditions

$$(21)\ u(x,0) = 0, \quad u(0,y) = 0, \quad u(x,1) = u_0, \quad u(1,y) = \frac{2u_0}{\pi} arctg\left(th\frac{\pi}{2} tg\frac{\pi y}{2}\right).$$

The analytical solution to the problem is given by function

$$(22) \qquad u(x,y) = \frac{2u_0}{\pi} arctg\left(th\frac{\pi x}{2} tg\frac{\pi y}{2}\right).$$

Convergence to the exact solution for the FVS and the CS is shown in Figure 7. The plots are obtained for the same values of angle ϕ_1 (curves I and I' in the figure for the CS and FVS, respectively) and ϕ_2 (curves II and II'), the parameter $u_0 = 0.1$. As one can see from the figure, the convergence rate is slightly different for stencils with $\phi_{max} = \phi_1$, while for $\phi_{max} = \phi_2$ the standard scheme converges noticeably better. The dependence of the convergence rate on the maximum grid angle is not so trivial as in the previous example; for a wide range of ϕ_{max} the greater value of the maximum grid angle provides the better convergence for both schemes. However, for angles ϕ_{max} close to π the degeneration of FV cells becomes as strong as to make the convergence rate slower.

Table 2
GMRES convergence test for standard (FVS) and corrected (CS) FV schemes.

N	FVS	CS	N_{st}	FVS_{st}
181	29	14	181	10
685	95	28	761	20
2665	251	69	3121	43

In our code we use GMRES algorithm (Saad [15]) to solve the algebraic system of equations obtained as a result of the discretization. The convergence of GMRES depends on the condition number of the system matrix A. A low rate of the convergence corresponds to a poorly conditioned matrix, while positive definite matrix A provides the best rate of convergence [15]. That is why the study of the convergence rate of GMRES may be considered as stability test for the corrected scheme.

To assess the GMRES convergence rate, the convergence test has been taken from the PETSc library (Balay et al. [2]). In this test the number of iterations necessary to meet the convergence is counted, the other GMRES parameters being fixed.

The sequence of “bad” grids with the number N of grid nodes, where the standard scheme produces a poorly conditioned matrix A, is generated by isotropic refinement of grid G3 shown in Figure 4. For each “bad” grid, the standard grid with the similar number N_{st} of nodes, where the standard FV scheme produces the positive definite matrix A, is generated by cutting cells of a uniform Cartesian grid by two diagonals. The convergence rate of GMRES for both the standard (FVS) and the corrected (CS) schemes on “bad” grids is then compared with the results obtained for the FV scheme on the corresponding standard grid (FVS_{st}). Table 2 reports the number of GMRES iterations needed for the convergence. The test matrix A is generated for the boundary problem (18).

As one may expect, the “quasi-positive” corrected scheme produces a well-conditioned system matrix even on “bad” grids and, therefore, requires essentially fewer number of GMRES iterations than the standard scheme. The GMRES convergence rate for the corrected scheme is close to that obtained for the positive defined matrix A.

5. The three-dimensional case. In this section, we briefly discuss whether it is possible to extend the obtained results to the three-dimensional case. Let T be the volume formed by the union of all tetrahedra t_i, $i = 1, \dots, N$, which have the central node 0 as a common vertex. Consider a sphere S_R of radius R with the center at the point $\mathbf{r}_0$. As in the two-dimensional case, it is possible to introduce a control volume V_L as a convex polyhedron with faces created by setting in each tetrahedron t_i the plane passing through the points of intersection between the sphere S_R and the edges of the tetrahedron. Let us calculate the solution on the central node of the triangulation by using the formula

$$u(\mathbf{r}_0) = \frac{1}{4\pi R^2} \int_{S_R}\!\!\int u(x, y, z) ds. \tag{23}$$

We assume the linear reconstruction of the solution in each tetrahedron t_i:

$$u^i(x, y, z) = a_0^i + a_1^i x + a_2^i y + a_3^i z = a_0^i + (\boldsymbol{\nabla} u^i, \mathbf{r}). \tag{24}$$

For integration over the sphere the vector $\mathbf{r}$ in (24) is

$$\mathbf{r}(x, y, z) = \mathbf{r}_0 + R\mathbf{n}_s, \tag{25}$$

where $\mathbf{n}_s$ is the unit vector normal to the sphere surface. Substitution of (24), (25) into (23) and summation over all tetrahedra yields

$$u(\mathbf{r}_0) = \frac{1}{4\pi R^2}\sum_{i=1}^{N}\int_{S_R^i}\!\!\int \left[a_0^i + (\boldsymbol{\nabla} u^i, \mathbf{r}_0 + R\mathbf{n}_s)\right] ds = u(\mathbf{r}_0) + \frac{1}{4\pi R}\sum_{i=1}^{N}\int_{S_R^i}\!\!\int (\boldsymbol{\nabla} u^i, \mathbf{n}_s) ds. \tag{26}$$

The equality (26) gives us the following condition which may be considered as a finite volume discretization of the Laplace equation over the sphere S_R:

$$\sum_{i=1}^{N}\int_{S_R^i}\!\!\int (\boldsymbol{\nabla} u^i, \mathbf{n}_s) ds = 0. \tag{27}$$

It can be seen from (23), (27) that the discretization over the sphere provides us with the positive scheme. The sufficient condition, which allows us to consider in (27) the polyhedron V_L instead of the sphere S_R as a control volume providing the positive scheme, is that

$$\int_{S_R^i}\!\!\int (\boldsymbol{\nabla} u^i, \mathbf{n}_s) ds \equiv \int_{V_L^i}\!\!\int (\boldsymbol{\nabla} u^i, \mathbf{n}_s) ds \quad \forall i = 1, \ldots, N, \tag{28}$$

where the face V_L^i of V_L belongs to the tetrahedron t_i. However, unlike the two-dimensional case, this condition does not hold for the inscribed polyhedron V_L. Let S_{side}^i be the union of the three plane segments of the tetrahedron faces in the tetrahedron t_i, each segment being bounded by the edge of V_L^i, the circle arc of S_R^i, and the edges of t_i. To estimate the integral over S_R^i in (28), we consider for each tetrahedron t_i the auxiliary closed surface S_{aux}^i which comprises S_{side}^i, the face V_L^i, and the part S_R^i of the sphere. Since the gradient is a constant vector in each tetrahedron, the integral (28) may be transformed as

$$\int_{S_R^i}\!\!\int (\boldsymbol{\nabla} u^i, \mathbf{n}_s) ds = \left(\boldsymbol{\nabla} u^i, \int_{S_R^i}\!\!\int d\mathbf{s}\right),$$

where a vector elemental area $d\mathbf{s} = \mathbf{n}_s ds$. According to the gradient theorem,

$$\oiint_{S_{aux}^i} d\mathbf{s} \equiv \int_{S_R^i}\!\!\int d\mathbf{s} + \int_{V_L^i}\!\!\int d\mathbf{s} + \int_{S_{side}^i}\!\!\int d\mathbf{s} = 0.$$

It is not difficult to see that $\int_{S_{side}^i}\!\int d\mathbf{s} \neq 0$; therefore, the weight coefficients of the discretization over the control volume V_L are different from those in (27). Thus, in the three-dimensional case the condition (28) sufficient for the positive discretization does not hold.

Now we consider a discretization over the prescribed control volume V_{dual} defined as a polyhedron with faces e_i constructed under some geometric conditions (i.e., median or centroid dual). As in the two-dimensional case, we suggest that including close nodes into the reconstruction stencil provides us with a more positive scheme. Since the results of [13] demonstrate that it is possible to obtain the positive scheme for the Laplace equation on three-dimensional Delaunay meshes, we believe that a circumsphere criterion may be used to find the nodes close to the given tetrahedron t_i. If node n_c belonging to T lies inside the circumsphere S_i , it may be considered as being close to t_i and included into the reconstruction stencil for the given tetrahedron.

To analyze the behavior of the fluxes in the three-dimensional case we calculate the directional derivative $\frac{\partial u}{\partial \mathbf{n}}$ across the face e_i in the given tetrahedron t_i as

$$\frac{\partial u}{\partial \mathbf{n}} \approx \frac{u(\mathbf{r}^*) - u(\mathbf{r}_0)}{|\mathbf{r}^* - \mathbf{r}_0|}, \tag{29}$$

where $\mathbf{r}^* = (x^*, y^*, z^*)$ is defined as the point of intersection between the plane e_i and the perpendicular dropped to this plane from the central node of the triangulation. Since the components of the unit vector normal to the plane e_i may be calculated as

$$n_1^i = \frac{x^* - x_0}{|\mathbf{r}^* - \mathbf{r}_0|}, \quad n_2^i = \frac{y^* - y_0}{|\mathbf{r}^* - \mathbf{r}_0|}, \quad n_3^i = \frac{z^* - z_0}{|\mathbf{r}^* - \mathbf{r}_0|},$$

the expression (29) is transformed as

$$\frac{\partial u}{\partial \mathbf{n}} \approx a_1^* n_1^i + a_2^* n_2^i + a_3^* n_3^i = \text{const} = C^*,$$

provided the linear reconstruction of the solution $u(\mathbf{r}^*) - u(\mathbf{r}_0) = a_1^*(x^* - x_0) + a_2^*(y^* - y_0) + a_3^*(z^* - z_0)$ is used. The result of calculation is then compared with the formula

$$\frac{\partial u}{\partial \mathbf{n}} \approx a_1^i n_1^i + a_2^i n_2^i + a_3^i n_3^i = \text{const} = C^i.$$

If the value C^i is different from C^*, then we consider t_i as the tetrahedron where the false gradient appears. In this case it is possible to include nodes of the tetrahedron t_j which contains point r^* into the reconstruction stencil for the given tetrahedron t_i by using the least squares method. This correction may decrease the false gradient.

6. Conclusions. In the present work the analysis of how a discretization of Laplace's equation depends on grid geometry has been made. We have demonstrated the way to construct a control volume for a positive FV scheme with a linear reconstruction on any two-dimensional unstructured grid. The important result obtained here is that on arbitrary grids the Laplace operator requires a convex control volume to provide a positive discretization. This result indicates that grids with highly stretched cells are not appropriate for constructing the positive scheme, since gaps (or overlappings) may appear in the convex dual mesh.

For the prescribed geometry of the control volume, we have investigated whether it is possible to improve the positivity measures of the linear FV scheme on arbitrary grids by using extended stencils. Although having the empirical nature, the suggested approach to the stencil correction allows us to treat stretched cells effectively. Numerical experiments show that the developed technique produces a "quasi-positive" scheme. However, in spite of giving us all advantages of the positive discretization, the practical applicability of the corrected scheme is restricted. The produced stencils are nonsymmetric which may lead to the loss of conservativity of the scheme.

The property of positivity is very important and can be considered as a criterion of a proper discretization of Laplace's equation as it expresses a maximum principle which is an inherent feature of the Laplacean. On the other hand, being a natural property of FV schemes conservativity makes them attractive for many practical applications. The results obtained in this paper seem to indicate that the properties of positivity and conservativity are incompatible with each other on arbitrary grids. This fact demonstrates how far grid quality is crucial for the discretization.

In our opinion, in order to overcome the incompatibility between these two basic requirements, one should admit that due to space isotropy of the Laplace equation its discretization needs the grid to be isotropic, in a certain sense. A mesh with all edges of the same length gives us the simplest example, while a Delaunay triangulation can be considered as a more general kind of grid with space isotropy.

Grids with stretched cells are "alien" for the Laplace equation. For those problems, where stretched grids arise as a result of grid adaptation, it may be better to make a discretization of the full problem operator rather than discretize diffusion terms separately. Meanwhile, the further development of algorithms of fully automatic Delaunay grid generation is strongly needed to provide us with a positive discretization which holds the property of conservativity.

Acknowledgments. The author is grateful to two anonymous referees for their valuable comments which helped to improve the paper. The author appreciates the help of colleagues from the BOIM CFD research team, especially A.A. Martynov, S. Yu. Medvedev, and A.M. Sorokin, who provided the grids used in the work.

REFERENCES

[1] I. Babuška and A.K. Aziz, *On the angle condition in the finite element method*, SIAM J. Numer. Anal., 13 (1976), pp. 214–226.

[2] S. Balay, W. Gropp, L.C. McInnes, and D. Smith, *PETSc* 2.0 *Users Manual*, Tech. Report ANL 95/11, Argonne National Laboratory, Argonne, IL, 1997; also available online from http://www.mcs.anl.gov/petsc/petsc.html.

[3] T.J. Barth, *Numerical Aspects of Computing High-Reynolds Number Flows on Unstructured Meshes*, AIAA Paper 91-0721, 29th Aerospace Science Meeting, Reno, NV, 1991.

[4] T.J. Barth, *Aspects of Unstructured Grids and Finite-Volume Solvers for the Euler and Navier-Stokes Equations*, Computational Fluid Dynamics von Karman Lecture Series 1994-05, 1994.

[5] T.J. Barth and S.W. Linton, *An Unstructured Mesh Newton Solver for Compressible Turbulent Flows and Its Parallel Implementation*, AIAA Paper 95-0221, 33rd Aerospace Science Meeting and Exhibit, Reno, NV, 1995.

[6] F. Brezzi, L.D. Marini, and P. Pietra, *Two-dimensional exponential fitting and applications to drift-diffusion models*, SIAM J. Numer. Anal., 26 (1989), pp. 1342–1355.

[7] W.J. Coirier, *An Adaptively Refined, Cartesian, Cell-Based Scheme for the Euler and Navier-Stokes Equations*, Ph.D. thesis, University of Michigan, Ann Arbor, MI, 1994.

[8] M. Delanaye, M.J. Aftosmis, M.J. Berger, Y. Liu, and T.H. Pulliam, *Automatic Hybrid-Cartesian Grid Generation for High-Reynolds Number Flows around Complex Geometries*, AIAA Paper 99-0777, 37th Aerospace Science Meeting and Exhibit, Reno, NV, 1999.

[9] GGNS Project Research Team, *private communication*, The Boeing Company, M/S 67-LF, Seattle, WA, 1999.

[10] B. Joe, *Delaunay triangular meshes in convex polygons*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 514–539.

[11] P. Markowich and M. Zlamal, *Inverse-average-type finite element discretizations of self-adjoint second order elliptic problems*, Math. Comp., 51 (1989), pp. 431–449.

[12] N.B. Petrovskaya, *Large Angle Tolerant Discretization*, Boeing Tech. Report 104R, Boeing Operations International, Inc., Moscow, 1999.

[13] M. Putti and C. Cordes, *Finite element approximation of the diffusion operator on tetrahedra*, SIAM J. Sci. Comput., 19 (1998), pp. 1154–1168.

[14] S. Rippa, *Long and thin triangles can be good for linear interpolation*, SIAM J. Numer. Anal., 29 (1992), pp. 257–270.

[15] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS, Kent, UK, 1995.

[16] V.S. Sakovich, *Multiple-grid solution of Euler's equations on unstructured grids*, Comput. Math. Math. Phys., 34 (1994), pp. 1603–1616.

[17] J. Xu and L. Zikatanov, *A monotone finite element scheme for convection-diffusion equations*, Math. Comp., 68 (1999), pp. 1429–1446.

SIAM J. SCI. COMPUT.
Vol. 23, No. 3, pp. 910–939

ADAPTIVE WAVELET SCHEMES FOR ELLIPTIC PROBLEMS—IMPLEMENTATION AND NUMERICAL EXPERIMENTS*

ARNE BARINKA†, TITUS BARSCH†, PHILIPPE CHARTON‡, ALBERT COHEN§, STEPHAN DAHLKE¶, WOLFGANG DAHMEN†, AND KARSTEN URBAN†

Abstract. Recently an adaptive wavelet scheme could be proved to be asymptotically optimal for a wide class of elliptic operator equations in the sense that the error achieved by an adaptive approximate solution behaves asymptotically like the smallest possible error that can be realized by *any* linear combination of the corresponding number of wavelets. On one hand, the results are purely asymptotic. On the other hand, the analysis suggests new algorithmic ingredients for which no prototypes seem to exist yet. It is therefore the objective of this paper to develop suitable data structures for the new algorithmic components and to obtain a quantitative validation of the theoretical results. We briefly review first the main theoretical facts, describe the main ingredients of the algorithm, highlight the essential data structures, and illustrate the results by one- and two-dimensional numerical examples including comparisons with an adaptive finite element scheme.

Key words. elliptic operator equations, multiscale methods, adaptive methods, wavelets, quasi-sparse matrices and vectors, adaptive operator application, fast matrix–vector multiplication, best N-term approximation, thresholding, Besov spaces, C++, STL

AMS subject classifications. 35B65, 41A25, 41A46, 42C15 46E35, 65F99, 65N12, 65N55

PII. S1064827599365501

1. Introduction.

The development of adaptive numerical methods is of enormous current interest. Although such concepts have not yet entered industrial applications at large, current research developments, for instance, in a finite element context, indicate their very promising potential [1, 3, 10, 12, 49]. Such hopes and numerical experiences are, however, contrasted by negative statements proved in the context of complexity theory. In fact, on a rigorous level not much has been proved about adaptive finite element schemes in comparison with a priori fixed meshes. To our knowledge, the only result in this direction is [38], where an adaptive finite element scheme for the bivariate Poisson's equation using piecewise linear elements was proven to converge without a priori assumptions on the unknown solution such as the *saturation property*. On the other hand, in the context of wavelet discretizations,

*Received by the editors January 19, 2000; accepted for publication (in revised form) January 23, 2001; published electronically August 29, 2001. This work was started during the European Summer School CEMRACS '98 that took place in Marseille. The authors have been supported by the following funds: Volkswagen–Stiftung in the priority project *Modellierung komplexer Systeme in der Verfahrenstechnik*, Deutsche Forschungsgemeinschaft (DFG) grant Da 117/13–1, project *Adaptive Multiscale Methods*, European Commission in the TMR project *Wavelets and Multiscale Methods in Numerical Analysis and Simulation*, DAAD (Deutscher Akademischer Austauschdienst) within the Procope project.

http://www.siam.org/journals/sisc/23-3/36550.html

†RWTH Aachen, Institut für Geometrie und Praktische Mathematik, Templergraben 55, 52056 Aachen, Germany (barinka@igpm.rwth-aachen.de, barsch@igpm.rwth-aachen.de, dahmen@igpm.rwth-aachen.de, urban@igpm.rwth-aachen.de).

‡IREMIA—Université de la Réunion, 15, avenue René Cassin, BP 7151, 97715 Saint Denis Messag. Cedex 9, La Réunion, France (charton@univ-reunion.fr).

§Laboratoire d'Analyse Numérique, Université Pierre et Marie Curie, 4 Place Jussieu, 75252 Paris Cedex 05, France (cohen@ann.jussieu.fr).

¶Universität Bremen, Fachbereich 3, ZeTeM, P.O. Box 330440, 28359 Bremen, Germany (dahlke@math.uni-bremen.de).

nowadays much more is known. In [22] an adaptive wavelet scheme was proved to converge for a wide class of elliptic operator equations including, in particular, differential operators as well as singular integral operators. This result was extended to saddle point problems in [24].

Moreover, quite recently substantial progress could also be accomplished in the analysis of the *speed* of convergence [17]. In [17], an adaptive wavelet scheme has been developed which is shown to be *asymptotically optimal* for the same class of elliptic operator equations referred to above in the following sense: its rate of convergence to the exact solution with respect to the number N of degrees of freedom—i.e., of wavelets which are used to describe the solution—is the same as the rate of convergence of the *best N-term approximation* which would typically be obtained by retaining the N largest wavelet coefficients of the exact solution. Moreover, the number of floating point operations required to compute the approximate solution stays proportional to the number N of wavelets needed to approximate the solution at any desired level of accuracy. In addition, sorting requires at most the order of $N \log(N)$ operations. The proof of the latter fact is constructive in the sense that the algorithm is described to the level of detail that the number of arithmetic operations can be rigorously estimated. To our knowledge, these are the first rigorously proven convergence rates accompanied with a corresponding operations count.

The result is interesting from two points of view. First, it is known that the rate of best approximation either by N-term wavelet combinations or by optimal adaptive refinement of finite element spaces can be characterized by a certain type of Besov regularity [36], in contrast to the rate of approximation by uniform refinements which is determined by Sobolev regularity. Therefore, the above adaptive wavelet scheme provides an asymptotically better accuracy/work balance than schemes based on uniform discretizations, e.g., when the solution has isolated singularities or, more systematically, when the solution lacks Sobolev regularity relative to Besov regularity. However, since the results are asymptotic a more quantitative assessment of the performance is of equal interest in practical applications. Second, the analysis of the scheme suggests new algorithmic ingredients centering on an adaptive matrix–vector multiplication combined with sorting entries of sequences. Therefore the efficient realization of these ingredients and the development of suitable data structures that best support the conceptual strength of the scheme in practical realizations is a challenging task. In fact, the realization of that task seems to be essential for a quantitative validation of the theoretical results which after all are phrased in a necessarily simplified computational model.

This paper describes the developments of such algorithmic ingredients and corresponding data structures and reports numerical results in one and two dimensions. The paper is organized as follows. In section 2, we briefly review the main theoretical facts needed for the understanding of the algorithm and extract from theory the essential requirements on implementation. Moreover, these considerations will guide the selection of test examples. Section 3 is devoted to a brief outline of the new data structures. These structures follow the code design principle of *separating data from the algorithm*, i.e., here the adaptive algorithm is realized independently of the input parameters such as the underlying domain and the choice of the wavelet basis. In section 4, we present our numerical experiments. The examples are designed to highlight the effects of different sources of singularities whose occurrence, according to the theoretical part, makes adaptive schemes more efficient than nonadaptive ones. This covers singularities induced by the right-hand side date, by the shape of the domain, or by both, as well as the effect of large Sobolev norms versus Besov norms. To exploit theoretical knowledge for our validation we choose as a test example the classical Poisson problem on an L-shaped domain as well as a Helmholtz problem with small viscosity. In the latter case the energy space differs more and more from the Sobolev space H^1 when the viscosity decreases. This allows us to test the robustness of the scheme. Moreover, we confirm the quantitative advantage of the stronger compressibility of higher order wavelets in spite of their larger supports. Finally, we compare the wavelet scheme with an adaptive finite element scheme. Several other examples, figures, and also some movies can be obtained from the web page http://www.igpm.rwth-aachen.de/adaptive.

2. Theoretical background.

2.1. The abstract problem. The algorithms discussed in this paper apply to the following scope of problems. Suppose that H is a Hilbert space with norm $\|\cdot\|_H$ induced by the inner product $\langle\cdot,\cdot\rangle$ and that the self-adjoint operator $A : H \to H'$, where H' is the normed dual of H, is *H-elliptic*, i.e.,

$$a(v,w) := \langle Av, w\rangle \lesssim \|v\|_H\|w\|_H \quad \text{and} \quad a(v,v) \sim \|v\|_H^2. \tag{2.1}$$

Here $a \lesssim b$ means that a can be uniformly bounded by a constant multiple of b independent of any parameters on which a and b may depend. $a \gtrsim b$ is to be understood analogously and $a \sim b$ states that $a \lesssim b$ and $a \gtrsim b$. Clearly (2.1) implies that A is an isomorphism from H to H', i.e.,

$$\|Av\|_{H'} \sim \|v\|_H, \quad v \in H. \tag{2.2}$$

Thus the equation

$$Au = f \tag{2.3}$$

has for any $f \in H'$ a unique solution which will always be denoted by u. Typical examples are second order elliptic boundary value problems with homogeneous Dirichlet boundary conditions on some open domain $\Omega \subset \mathbb{R}^d$. In this case $H = H_0^1(\Omega)$ and $H' = H^{-1}(\Omega)$. Other examples are obtained by turning an exterior boundary value problem into a singular integral equation on the boundary Γ of the domain. For a formulation in terms of the single layer potential operator one obtains, for instance, $H = H^{-1/2}(\Gamma)$ and $H' = H^{1/2}(\Gamma)$; see [19, 45] for details. In the above examples H is a Sobolev space and one has either $H \subset L_2 \subset H'$ or $H' \subset L_2 \subset H$. We sometimes write $H = H^t$ to indicate the Sobolev regularity although often a closed subspace of the full Sobolev space determined by boundary conditions is meant. H^{-t} is always the dual of this particular subspace.

We hasten to add though that A need not be a scalar equation but could as well represent a system in which case H is typically a product of Sobolev spaces as long as the above assumptions are fulfilled.

We finally emphasize that H should be viewed as representing the *energy* space associated with (2.1) and therefore need not be a Sobolev space. As a typical example consider $Au = -\varepsilon\Delta u + u$ on $\Omega \subset \mathbb{R}^d$ with homogeneous Dirichlet boundary conditions. In this case H-ellipticity holds *uniformly* in $\varepsilon > 0$ for the norm $\|\cdot\|_H^2 := \|\cdot\|_{L_2(\Omega)}^2 + \varepsilon|\cdot|_{H^1(\Omega)}^2$.

We are interested in solving (2.3) approximately with the aid of a Galerkin method, i.e., we pick some finite-dimensional space $S \subset H$ and search for $u_S \in S$

such that

$$\langle Au_S, v\rangle = \langle f, v\rangle, \qquad v \in S, \tag{2.4}$$

where $\langle\cdot,\cdot\rangle$ denotes the standard L_2-inner product.

We have paused above to describe a wide class of examples for the following reason. The *separation of algorithm and data* can be realized for the scheme discussed below to a significantly higher extent than for more conventional discretizations. The algorithm remains essentially the same in all the mentioned examples. It is therefore no principal limitation to consider only numerical tests for second order elliptic boundary value problems including, however, the above example of parameter dependent energy spaces. The main point will be that for this class of examples we can resort to important analytical information that will help to validate and interpret the numerical results in a more quantitative fashion.

2.2. An equivalent ℓ_2-problem. In our context the trial spaces S in (2.4) will be spanned by elements of a *wavelet* basis $\Psi = \{\psi_\lambda : \lambda \in \mathcal{J}\}$ for H. We will postpone at this point any technical description of the basis Ψ (which necessarily depends on the particular setting at hand) but will only list those properties that will be relevant in the following. The indices $\lambda \in \mathcal{J}$ typically encode several types of information, namely, the *scale*, often denoted by $|\lambda|$, the spatial location, and also the type of the wavelet. Recall that in a classical setting a tensor product construction yields $2^d - 1$ types of wavelets [34, 46]. For instance, for wavelets on the real line, λ can be identified with (j,k), where $j = |\lambda|$ denotes the dyadic refinement level and $2^{-j}k$ signifies the location of the wavelet. In fact, we will require the wavelets to be local in the sense that $\operatorname{diam}(\operatorname{supp}\psi_\lambda) \sim 2^{-|\lambda|}$, $\lambda \in \mathcal{J}$.

These wavelets are usually normalized in L_2. What matters in the present case is that a properly scaled version of Ψ is a *Riesz basis* for the energy space H. This means that there exists a *diagonal matrix* $\mathbf{D} = \operatorname{diag}(\omega_\lambda : \lambda \in \mathcal{J}\}$ such that each $v \in H$ has a unique expansion $v = \sum_{\lambda\in\mathcal{J}} v_\lambda \omega_\lambda^{-1}\psi_\lambda =: \mathbf{v}^T\mathbf{D}^{-1}\Psi$ such that

$$\|\mathbf{v}\|_{\ell_2(\mathcal{J})} \sim \|\mathbf{v}^T\mathbf{D}^{-1}\Psi\|_H. \tag{2.5}$$

For instance, when $H = H^t$ a canonical choice is $\omega_\lambda = 2^{t|\lambda|}$. When $\|\cdot\|_H^2 := \|\cdot\|_{L_2(\Omega)}^2 + \varepsilon|\cdot|_{H^1(\Omega)}^2$ the choice $\omega_\lambda := 1 + \sqrt{\varepsilon}2^{|\lambda|} \sim a(\psi_\lambda,\psi_\lambda)^{1/2}$ ensures that (2.5) holds with constants *independently* of ε provided that $\{\psi_\lambda\}_{\lambda\in\mathcal{J}}$ and $\{2^{-|\lambda|}\psi_\lambda\}_{\lambda\in\mathcal{J}}$ are Riesz bases for L_2 and H^1, respectively. This will be seen to entail asymptotically optimal performance of the adaptive wavelet scheme uniformly in ε.

Relations similar to (2.5) are also known to hold for Sobolev spaces in L_p for $p \neq 2$. Moreover, interpolation between such spaces provides norm equivalences for a whole range of *Besov spaces* $B_q^\alpha(L_p)$ [25, 37, 39, 46]. In the present context we will have to make use of the following special case:

$$\|\boldsymbol{d}\|_{\ell_\tau(\mathcal{J})} \sim \|\boldsymbol{d}^T\Psi\|_{B_\tau^\alpha(L_\tau)}, \tag{2.6}$$

where the smoothness index α and the integrability index τ are related by $\tau^{-1} = \alpha/d + 1/2$.

Once a basis with the above properties is given, the operator equation (2.3) over a function space H can be transformed into an *equivalent matrix* equation over the corresponding *sequence space*. In fact, the representation of

$$\boldsymbol{A} := \mathbf{D}^{-1}\langle\Psi, A\Psi\rangle\mathbf{D}^{-1} := \left(\omega_\lambda^{-1}\omega_\nu^{-1}\langle\psi_\lambda, A\psi_\nu\rangle\right)_{\lambda,\nu\in\mathcal{J}} \tag{2.7}$$

of A with respect to the Riesz basis $\mathbf{D}^{-1}\Psi$ of H is, because of the norm equivalence (2.5) in conjunction with ellipticity (2.2), an automorphism on ℓ_2 [26, 28].

THEOREM 2.1. *The function $u = \boldsymbol{d}^T\Psi \in H$ solves the original operator equation* (2.3) *if and only if the sequence $\mathbf{u} := \mathbf{D}\boldsymbol{d}$ solves the matrix equation*

$$\boldsymbol{A}\mathbf{u} = \mathbf{f}, \tag{2.8}$$

where $\mathbf{f} := \mathbf{D}^{-1}\langle\Psi, f\rangle = \{\omega_\lambda^{-1}\langle\psi_\lambda, v\rangle\}_{\lambda\in\mathcal{J}}$.

Moreover, denoting by $\|\cdot\|$ the spectral norm on ℓ_2, the matrix $\boldsymbol{A}$ defined by (2.7) *satisfies*

$$\|\boldsymbol{A}\|,\ \|\boldsymbol{A}^{-1}\| < \infty. \tag{2.9}$$

As an immediate consequence there exists a finite number κ such that all finite sections $\boldsymbol{A}_\Lambda := \left(\omega_\lambda^{-1}\omega_\nu^{-1}\langle\psi_\lambda, A\psi_\nu\rangle\right)_{\lambda,\nu\in\Lambda}$, $\Lambda \subset \mathcal{J}$, have uniformly bounded condition numbers

$$\operatorname{cond}_2(\boldsymbol{A}_\Lambda) \le \kappa, \qquad \Lambda \subset \mathcal{J}. \tag{2.10}$$

Hence the original problem has been reduced to an equivalent well-posed problem in ℓ_2. This fact will be crucial in the following.

2.3. Quasi-sparse matrices. A specific advantage of wavelet discretizations is that for a large class of elliptic operators (including singular integral operators), the resulting matrix $\boldsymbol{A}$ exhibits fast decay off the diagonal. More precisely, when $H = H^t$ and $\omega_\lambda = 2^{t|\lambda|}$, this takes the form of the following estimate:

$$2^{-(|\lambda'|+|\lambda|)t}|\langle A\psi_{\lambda'}, \psi_\lambda\rangle| \ \lesssim\ \frac{2^{-||\lambda|-|\lambda'||\sigma}}{(1+d(\lambda,\lambda'))^{d+2\tilde{m}+2t}},\quad d(\lambda,\lambda') := 2^{\min(|\lambda|,|\lambda'|)}\operatorname{dist}(\Omega_\lambda,\Omega_{\lambda'}), \tag{2.11}$$

where $\Omega_\lambda := \operatorname{supp}\psi_\lambda$ and $\sigma > d/2$ is a constant depending on the regularity of the wavelets ψ_λ. The validity of (2.11) has been established for a wide range of cases, including classical pseudodifferential operators and Calderon–Zygmund operators (see, e.g., [28]).

It is important to note, however, that (2.11) is only a *sufficient* condition for the following *compression property* of $\boldsymbol{A}$ that will be needed later. $\boldsymbol{A}$ is said to belong to the class $\mathcal{A}_s$ if there exists a positive summable sequence $(\alpha_j)_{j\ge0}$ and for every $j \ge 0$ there exists a matrix $\boldsymbol{A}_j$ with at most $2^j\alpha_j$ nonzero entries per row and column such that

$$\|\boldsymbol{A}_j - \boldsymbol{A}\| \ \lesssim\ \alpha_j 2^{-sj}. \tag{2.12}$$

The following fact has been proved in [17].

PROPOSITION 1. *Let*

$$s^* := \min\left\{\frac{\sigma}{d} - \frac{1}{2}, \frac{2t+2\tilde{m}}{d}\right\}. \tag{2.13}$$

Then $\boldsymbol{A}$ belongs to $\mathcal{A}_s$ for every $s < s^*$.

For matrices with the particular decay properties (2.11) concrete truncation rules can be given [17]. Given j, set

$$\tilde{a}_{\lambda,\nu} := \begin{cases} a_{\lambda,\nu}, & ||\lambda| - |\nu|| \le j/d \text{ and } d(\lambda,\nu) \le 2^{j/d-||\lambda|-|\nu||}\,\gamma(||\lambda|-|\nu||), \\ 0, & \text{else.} \end{cases} \tag{2.14}$$

Here $\gamma(n)$ is any summable sequence, e.g., $\gamma(n) := (1+n)^{-2/d}$.

One should note that because of the locality of the wavelets the first condition in (2.14) already suffices for local operators A (such as differential operators) which will be used later in the examples.

2.4. The adaptive strategy. The practical realization of adaptive approximations to (2.3) in a finite element context is to refine and derefine step by step a given mesh according to a posteriori local error indicators. The point of view taken by wavelet schemes is somewhat different. Trial spaces are refined directly by incorporating additional basis functions whose selection depends on the previous step. Specifically, setting for any finite subset $\Lambda \subset \mathcal{J}$ our trial spaces will always be of the form $S_\Lambda := \operatorname{span}\{\psi_\lambda : \lambda \in \Lambda\}$, where the index set Λ is to be adapted to the solution. In fact, denoting by $u_\Lambda \in S_\Lambda$ always the Galerkin solution determined by (2.4), we start with some small index set Λ_0 (possibly the empty set) and proceed as follows:

> *Given* Λ_j *and* u_{Λ_j} *and some* fixed $\theta \in (0,1)$*, find* $\Lambda_{j+1} \supset \Lambda_j$ *as* small as possible *such that the new error* $u - u_{\Lambda_{j+1}}$ *in the energy norm is at most* θ *times the previous error. Obviously, iteration of this step entails convergence of the resulting sequence of approximations in the energy norm.*

Successively enlarging index sets in this way, one hopes to track the *most significant* coefficients in the true wavelet expansion $\boldsymbol{d}^T\Psi = \mathbf{u}^T\mathbf{D}^{-1}\Psi$ of the unknown solution u. It is easy to see that the function $u_\Lambda = \sum_{\lambda\in\Lambda} \omega_\lambda^{-1} (\mathbf{u}_\Lambda)_\lambda \psi_\lambda$ with coefficient vector $\mathbf{u}_\Lambda = ((\mathbf{u}_\Lambda)_\lambda)_{\lambda\in\Lambda}$ (with respect to the scaled wavelet basis) solves the Galerkin system (2.4) for $S = S_\Lambda$ if and only if $\mathbf{u}_\Lambda$ solves

$$\boldsymbol{A}_\Lambda \mathbf{u}_\Lambda = \mathbf{f}_\Lambda := \mathbf{f}|_\Lambda. \tag{2.15}$$

Moreover, Theorem 2.1 suggests working completely on the *discrete* side for both the finite- and infinite-dimensional problems. Therefore, although $\mathbf{u}_\Lambda \in \mathbb{R}^{\#\Lambda}$ is a finite vector, it will sometimes be convenient to view $\mathbf{u}_\Lambda$ as a sequence in ℓ_2, i.e., all components of $\mathbf{u}_\Lambda$ outside Λ are understood to be zero. Since it will always be clear from the context which interpretation is meant we will not introduce any notational distinction between the finite vector $\mathbf{u}_\Lambda$ and its canonical injection in ℓ_2. Likewise for $\mathbf{v} \in \ell_2$ its restriction to Λ is denoted by $\mathbf{v}|_\Lambda$.

Defining now the *discrete energy norm*

$$\|\mathbf{v}\|^2 := \mathbf{v}^T\boldsymbol{A}\mathbf{v} =: \mathbf{a}(\mathbf{v},\mathbf{v}), \tag{2.16}$$

(2.5) and equation (2.9) in Theorem 2.1 say that

$$\|\mathbf{v}^T\mathbf{D}^{-1}\Psi\|_H \sim \|\mathbf{v}\| \sim \|\mathbf{v}\|_{\ell_2} \sim \|\boldsymbol{A}\mathbf{v}\| \sim \|\boldsymbol{A}\mathbf{v}\|_{\ell_2}, \tag{2.17}$$

so that measuring errors in $\|\cdot\|$ or $\|\cdot\|_{\ell_2}$ and for the corresponding functions in the energy norm $\|\cdot\|_H$ is the same up to uniform constants.

Next note that if we can find for a given $\Lambda \subset \mathcal{J}$ an index set $\hat\Lambda \supset \Lambda$ such that for some $\beta \in (0,1)$

$$\|\mathbf{u}_{\hat\Lambda} - \mathbf{u}_\Lambda\| \geq \beta\|\mathbf{u} - \mathbf{u}_\Lambda\|, \tag{2.18}$$

Galerkin orthogonality and the Pythagoras theorem ensure the error reduction

$$\|\mathbf{u} - \mathbf{u}_{\hat\Lambda}\| \leq \theta\|\mathbf{u} - \mathbf{u}_\Lambda\| \tag{2.19}$$

with $\theta := \sqrt{1-\beta^2}$. This strategy is also the starting point in [22] and has been used even earlier in the finite element context; see, e.g., [12, 38]. There, it was usually *assumed* to hold for some *fixed* refinement of the old trial space and therefore referred to as *saturation assumption.* However, as in [22] and before for a much more specialized situation in [38] the scheme described below will *guarantee* (2.18) without an a priori assumption like the saturation property. In addition, in contrast to previous cases, concrete and in some sense asymptotically optimal *convergence rates*, relating the error $\|\mathbf{u}-\mathbf{u}_{\Lambda_j}\|$ to the number of degrees of freedom $N_j = \#\Lambda_j$, will be realized.

To this end, for any $\hat\Lambda \supset \Lambda$ one has by Galerkin orthogonality

$$\|\mathbf{u}_{\hat\Lambda}-\mathbf{u}_\Lambda\| \gtrsim \|\boldsymbol{A}(\mathbf{u}_{\hat\Lambda}-\mathbf{u}_\Lambda)\|_{\ell_2} \geq \|\boldsymbol{A}(\mathbf{u}_{\hat\Lambda}-\mathbf{u}_\Lambda)|_{\hat\Lambda}\|_{\ell_2} = \|\boldsymbol{A}(\mathbf{u}-\mathbf{u}_\Lambda)|_{\hat\Lambda}\|_{\ell_2}.$$

Thus defining the residual

$$\mathbf{r}_\Lambda := \boldsymbol{A}(\mathbf{u}-\mathbf{u}_\Lambda) = \mathbf{f} - \boldsymbol{A}\mathbf{u}_\Lambda,$$

the above estimate states that for some constant $c_1 \in (0,1)$ depending only on the constants in (2.17)

$$\|\mathbf{u}_{\hat\Lambda}-\mathbf{u}_\Lambda\| \geq c_1\|\mathbf{r}_\Lambda|_{\hat\Lambda}\|_{\ell_2}. \tag{2.20}$$

Key idea. If $\hat\Lambda$ can be chosen such that

$$\|\mathbf{r}_\Lambda|_{\hat\Lambda}\|_{\ell_2} \geq a\|\mathbf{r}_\Lambda\|_{\ell_2} \tag{2.21}$$

holds for some fixed $a \in (0,1)$, then, since (2.17) implies $\|\mathbf{r}_\Lambda\|_{\ell_2} \gtrsim \|\mathbf{u}_\Lambda - \mathbf{u}\|$, one infers from (2.20) that there exists a constant $\beta \in (0,1)$ such that (2.18) and hence also (2.19) $\|\mathbf{u}_{\hat\Lambda}-\mathbf{u}\| \leq \theta\|\mathbf{u}_\Lambda-\mathbf{u}\|$ holds.

Thus the reduction (2.19) of the error has been reduced to catching the *bulk* of the *residual* $\mathbf{r}_\Lambda$ by finding its most significant coefficients in the sense of (2.21). However, $\mathbf{r}_\Lambda$ is in general still an *infinite* array. So in practice, $\mathbf{r}_\Lambda$ has to be approximated first sufficiently well. This will be realized by exploiting the structure of $\boldsymbol{A}$ and the knowledge about the right-hand-side data $\mathbf{f}$; see [17] and the appendix below for details. Nevertheless, to make the basic mechanisms more transparent we suppress these issues for a moment and formulate a core ingredient of the refinement strategy as the following (idealized) routine:

GROW $(\Lambda, \mathbf{u}_\Lambda) \to (\hat\Lambda, \mathbf{u}_{\hat\Lambda})$. *Given* $(\Lambda, \mathbf{u}_\Lambda)$ *find the* **smallest** $\hat\Lambda \supset \Lambda$ *such that* $\|\mathbf{r}_\Lambda|_{\hat\Lambda}\|_{\ell_2} \geq a\|\mathbf{r}_\Lambda\|_{\ell_2}$.

Note, however, that even if one were able to perform **GROW** it is by no means clear whether merely iterating the procedure **GROW** leads to (in some sense) an optimal algorithm.

2.5. Best N-term approximation.

To obtain a conceptual *benchmark* it is important to clarify first what the *optimal* outcome of an adaptive scheme might be. If N is the number of degrees of freedom produced by the algorithm and if we wish to measure the error $\|u-u_\Lambda\|_H$, the optimal outcome is clearly given by a function u_N which minimizes $\|v-u\|_H$ over all v which are N-term linear combinations of wavelets. Again Theorem 2.1 and (2.17) imply that, up to a multiplicative uniform constant, this amounts to defining the corresponding vector $\mathbf{u}_N$ as the *best approximation* of $\mathbf{u}$ in ℓ_2 by a vector with N nonzero coordinates, i.e., $\mathbf{u}_N$ is obtained by retaining the N largest components of $\mathbf{u}$. Of course, since $\mathbf{u}$ is not known, $\mathbf{u}_N$ is not directly available. In summary, the best that can be achieved by an adaptive scheme is to produce approximate solutions $\mathbf{u}_\Lambda$ such that $\|\mathbf{u}-\mathbf{u}_\Lambda\|$ has the same asymptotic decay rate as $\|\mathbf{u}-\mathbf{u}_{\#\Lambda}\|_{\ell_2}$.

It has been shown in [17] that optimality in the above sense can indeed be ensured when the iteration of **GROW** is every so often interrupted by a *clean-up* step. This simply means that after several applications of **GROW** one has to discard all coefficients in the current approximation $\mathbf{u}_\Lambda$ whose modulus is below a certain threshold. This threshold is chosen so that the current error is at most multiplied by a fixed uniform constant. While thereby the error gets only worse by a little this will turn out to have an essential effect on the behavior of the residual with respect to certain norms that are somewhat stronger than the ℓ_2-norm. We summarize this clean-up or *thresholding step* as follows.

THRESH $(\Lambda,\mathbf{u}_\Lambda)\rightarrow(\tilde\Lambda,\mathbf{u}_{\tilde\Lambda})$. *If* $\|\mathbf{u}-\mathbf{u}_\Lambda\|_{\ell_2}\le\varepsilon$, *find* **smallest** $\tilde\Lambda\subset\Lambda$ *such that* $\|\mathbf{u}_\Lambda-\mathbf{u}_\Lambda|_{\tilde\Lambda}\|_{\ell_2}\le 4\varepsilon$.

Note that both routines require us to *sort* the coefficients with respect to their modulus.

2.6. An optimal (idealized) algorithm. We next give a rough idealized version of an adaptive wavelet scheme whose practical counterpart will turn out to be optimal with respect to convergence rates as well as work count.

ALGORITHM.

```
(i) $\Lambda_0=\emptyset$, $\mathbf{r}_{\Lambda_0}=\mathbf{f}$, $\varepsilon_0:=\|\mathbf{f}\|_{\ell_2}$, $j=0$, $\varepsilon$ given target accuracy.
(ii) Determine $(\Lambda_{j+1},\mathbf{u}_{\Lambda_{j+1}})$ from $(\Lambda_j,\mathbf{u}_{\Lambda_j})$ such that $\|\mathbf{u}-\mathbf{u}_{\Lambda_{j+1}}\|_{\ell_2}\le\varepsilon_j/2:=\varepsilon_{j+1}$ as follows.
        Set $\Lambda_{j,0}:=\Lambda_j$, $\mathbf{u}_{j,0}:=\mathbf{u}_j$;
            For $k=1,2,\dots,K$ apply GROW $(\Lambda_{j,k-1},\mathbf{u}_{\Lambda_{j,k-1}})\rightarrow(\Lambda_{j,k},\mathbf{u}_{\Lambda_{j,k}})$
            $(\|\mathbf{r}_{\Lambda_{j,k-1}}|_{\Lambda_{j,k}}\|_{\ell_2}\ge\frac{1}{2}\|\mathbf{r}_{\Lambda_{j,k-1}}\|_{\ell_2})$;
        Apply THRESH $(\Lambda_{j,K},\mathbf{u}_{\Lambda_{j,K}})\rightarrow(\Lambda_{j+1},\mathbf{u}_{\Lambda_{j+1}})$;
        If $\varepsilon_{j+1}\le\varepsilon$ stop; else $j+1\rightarrow j$, go to (ii).
```

The maximal (and total) number K of applications of **GROW** can be shown to be uniformly bounded, depending only on the constants in (2.17). Moreover, a detailed description of the fully computable version **ALGORITHM**^{c} of the above algorithm is given in the appendix to which the following result refers. In what follows it will always be assumed that the right-hand-side data $\langle f,\psi_\lambda\rangle$ are entirely given and that the computation of the entries of $\boldsymbol{A}$ can be computed on average at unit cost.

THEOREM 2.2 (see [17]). *The computable version* **ALGORITHM**^{c} *always produces a solution with the desired accuracy after a finite number of steps.*

Moreover, assume that $\boldsymbol{A}\in\mathcal{A}_s$ *for* $0\le s<s^*$ *(recall Proposition* 1*). If the solution* u *to the operator equation* (2.3) *has the property that for some* $s<s^*$

$$\sigma_N(u):=\inf_{d_\lambda,\lambda\in\Lambda,\#\Lambda\le N}\|u-\sum_{\lambda\in\Lambda}d_\lambda\psi_\lambda\|\ \lesssim\ N^{-s},\tag{2.22}$$

then **ALGORITHM**^{c} *generates a sequence* $u_{\Lambda_j}=\sum_{\lambda\in\Lambda_j}(\mathbf{u}_{\Lambda_j})_\lambda\omega_\lambda^{-1}\psi_\lambda$ *of Galerkin solutions to* (2.4) *satisfying*

$$\|u-u_{\Lambda_j}\|\ \lesssim\ (\#\Lambda_j)^{-s}.\tag{2.23}$$

Moreover, the number of arithmetic operations *needed to compute* u_{Λ_j} *stays* proportional *to* $\#\Lambda_j$. *The number of operations needed for* sorting *stays bounded by* $(\#\Lambda_j)\log(\#\Lambda_j)$.

It is important to note that the above algorithm does *not* require any a priori knowledge about the rate of N-term approximability of the solution. It is shown to automatically match the rate of best N-term approximation for a certain asymptotic range depending on the operator and the chosen basis.

2.7. Computational tasks. We add a few comments on the concrete computational tasks required by the computable version of **ALGORITHM**. A detailed account of these routines can be found in the appendix; see also [17], where, in particular, various parameters are identified that steer the refinement process. Of course, the central issue is to determine the bulk of $\mathbf{r}_\Lambda$ or, equivalently, to find a *good approximation* $\bar{\mathbf{r}}_\Lambda$ of finite length to $\mathbf{r}_\Lambda$ in ℓ_2. In this context one faces the following obvious obstructions. In order to stay within the promised bounds of computational complexity, one has to employ iterative solvers to determine an approximation $\bar{\mathbf{u}}_\Lambda$ to $\mathbf{u}_\Lambda$. One then still faces the problem of approximating the application of the *infinite* matrix $\boldsymbol{A}$ to the finitely supported sequence $\bar{\mathbf{u}}_\Lambda$ by some finite vector $\mathbf{w}_\eta$ within a suitable tolerance η. Approximating also $\mathbf{f}$ within that tolerance by a finite vector $\mathbf{f}_\eta$, one has to deal with the following perturbations of the true residual:

$$\mathbf{r}_\Lambda - \underbrace{(\mathbf{f}_\eta - \mathbf{w}_\eta)}_{\bar{\mathbf{r}}_\Lambda} = \underbrace{\mathbf{f} - \mathbf{f}_\eta + \boldsymbol{A}(\bar{\mathbf{u}}_\Lambda - \mathbf{u}_\Lambda) + \mathbf{w}_\eta - \boldsymbol{A}\bar{\mathbf{u}}_\Lambda}_{\text{error}}. \tag{2.24}$$

In the computable version **ALGORITHM**^{c} the routine **GROW** works with approximate residuals $\bar{\mathbf{r}}_\Lambda$ corresponding to suitable dynamic tolerances η, specified in [17]; see also the appendix below. This amounts to the following tasks:

(1) Determine a sufficiently good approximation $\mathbf{f}_\eta$.
(2) Determine $\bar{\mathbf{u}}_\Lambda$ by an iterative scheme. This requires repeated matrix–vector multiplications.
(3) Compute an approximation $\mathbf{w}_\eta$ to $\boldsymbol{A}\bar{\mathbf{u}}_\Lambda$. This requires an (approximate) application of the (infinite) matrix $\boldsymbol{A}$.
(4) Find a best N-term approximation to the resulting approximation (2.24) (or keep it).
(5) Threshold the current approximate Galerkin solution.

2.8. Adaptive matrix–vector multiplication. Clearly (3) reveals that a core requirement is the application of the *infinite*-dimensional operator $\boldsymbol{A}$ to a *finite* vector (formally extended by zero entries). Since $\boldsymbol{A}$ is an infinite matrix we also call such an application of $\boldsymbol{A}$ *matrix–vector multiplication.* An essential ingredient of the scheme is therefore the following *approximate* adaptive matrix–vector multiplication from [17] (see the appendix).

Proposition 2. *Let* $\mathbf{v}$ *be any finitely supported vector and suppose that* $\boldsymbol{A} \in \mathcal{A}_s$*; recall* (2.12)*. Defining* $\mathbf{v}_{[j]} := \mathbf{v}_{2^j}$ *(best N-term approximation in ℓ_2 for $N = 2^j$) and*

$$\mathbf{w}_j := \boldsymbol{A}_j \mathbf{v}_{[0]} + \boldsymbol{A}_{j-1}(\mathbf{v}_{[1]} - \mathbf{v}_{[0]}) + \cdots + \boldsymbol{A}_0(\mathbf{v}_{[j]} - \mathbf{v}_{[j-1]}), \tag{2.25}$$

then

$$\|\boldsymbol{A}\mathbf{v} - \mathbf{w}_j\|_{\ell_2} \lesssim 2^{-sj}, \tag{2.26}$$

uniformly in $\mathbf{v}$ *provided that* $\|\mathbf{v} - \mathbf{v}_{[j]}\|_{\ell_2} \lesssim 2^{-sj}$ *holds uniformly in* $\mathbf{v}$.

As a consequence, under the above assumptions on $\mathbf{v}$, the computational work $\mathbf{CW}(\eta)$ needed to realize an approximation $\mathbf{w}_\eta$ to $\boldsymbol{A}\mathbf{v}$ such that $\|\boldsymbol{A}\mathbf{v} - \mathbf{w}_\eta\|_{\ell_2} \le \eta$ is of the order

$$\mathbf{CW}(\eta) \sim \#\mathrm{supp}\, \mathbf{w}_\eta \lesssim \eta^{-1/s}. \tag{2.27}$$

It can be shown that only a finite uniformly bounded number of such approximate matrix–vector multiplications of the form (2.25) will be needed at each stage to fulfill the accuracy requirements for the next step. In particular, (2.27) will guarantee that the computational work stays in the desired bounds.

Note next that (1), (4), and (5) involve *thresholding* of a known array. The way this is performed here is to discard the largest possible number of small entries so that a desired accuracy is preserved by this perturbation. The core task there is to first *sort* the arrays and then sum successively entries in increasing order. This is also used in the error control of the fast matrix–vector multiplication (2.25) because the algorithm should at no stage use any a priori assumption about $\mathbf{u}$.

The above remarks explain the pivoting role of *sorting* and of the *fast approximate matrix–vector multiplication.* We will discuss later some consequences of these facts for the implementation. In particular, it will be seen that most of the data structures needed here can be designed *independently* of the particular application and even of the particular wavelet basis. The special application (e.g., whether $\boldsymbol{A}$ represents a differential or integral operator) enters primarily through calling the significant entries in the columns of $\boldsymbol{A}$ when performing (2.25).

2.9. When does adaptivity pay? Before discussing the implementation of the above scheme, we have to address some issues that will later explain the selection of test examples and subsequent numerical experiments. The guiding questions can be formulated as follows: (a) Theorem 2.2 asserts *asymptotic optimality.* It will be important to *quantify* how much the Galerkin solutions differ from the true best N-term approximations. (b) The range of decay rates N^{-s} for which the scheme is optimal depends, in particular, on the range of compressibility of $\boldsymbol{A}$. Proposition 1 indicates that this range, in turn, depends on the *regularity* of the wavelets. However, higher regularity entails wavelets with larger support. What is the interplay of these opposing effects? (c) What is the nature of functions $u = \mathbf{u}^T\mathbf{D}^{-1}\Psi$ whose best wavelet N-term approximation in the energy norm $\|\cdot\|_{H^t}$, say, or equivalently for which $\|\mathbf{u} - \mathbf{u}_N\|_{\ell_2}$ decays like N^{-s}? (d) In which case is the performance of the adaptive scheme asymptotically better compared with discretizations based on *uniform* mesh refinements?

Questions (c) and (d) are closely interrelated and can be answered theoretically. These answers will help us in properly interpreting subsequent numerical tests. As for (c), the functions u in question are nearly characterized by a certain *Besov regularity.* More precisely, their best wavelet N-term approximation in $\|\cdot\|_{H^t}$ decays like $N^{-(\alpha-t)/d}$ if u belongs to $B^{\alpha}_{\tau^*}(L_{\tau^*}(\Omega))$ with $\frac{1}{\tau^*} := \frac{\alpha-t}{d} + \frac{1}{2}$, $\alpha > t$. In contrast, to obtain the same rate by uniformly refined discretizations, u would have to belong to the much smaller Sobolev space H^{α}. In fact, $B^{\alpha}_{\tau^*}(L_{\tau^*}(\Omega))$ is the largest space of smoothness α in L_{τ^*} that is still continuously embedded in the energy space H^t. Thus adaptivity retains best possible decay rates under weaker regularity requirements on the approximant. The above Besov spaces admit singularities that may not occur in corresponding Sobolev spaces. In particular, when (within the compressibility range of $\boldsymbol{A}$) the solution u has higher Besov regularity than Sobolev regularity the adaptive scheme outperforms uniform mesh schemes asymptotically.

Such results have recently motivated a general effort to revisit the regularity of elliptic boundary value problems in terms of the above scale of Besov spaces and to identify the instances where this level of smoothness is substantially higher than when measured in the classical Sobolev scale; see [23] (and also [42]) for Lipschitz domains and [21] for polygonal domains. These facts will be used to validate our numerical test and to select relevant examples. However, our algorithm provides optimal approximations *without any a priori knowledge on the nature of the singularities.*

Finally, it should be kept in mind that, aside from such asymptotic considerations, the quantitative improvement over a uniform mesh scheme, even if the solution u has arbitrarily high pointwise smoothness, is stronger the more the relevant Sobolev norm exceeds the corresponding Besov norm, a fact that will be confirmed by the numerical examples.

3. Data structures and implementation. One key ingredient for the realization of the adaptive algorithm presented above is the organization of the data, i.e., how to handle the adaptively chosen wavelets so as to best suit the requirements of the adaptive algorithm. Clearly, since for uniform discretizations the number of unknowns is a priori known and can therefore be organized in static vectors containing all coefficients while this is no longer true for adaptive schemes, a certain overhead of data management is necessary but should be minimized.

In this section, we highlight the main features of our data structures for two reasons. First, the requirements of the presented adaptive wavelet algorithm differ essentially from the demands for more conventional methods such as adaptive finite element methods. Second, to our knowledge, this is the first consistent design of appropriate data structures for adaptive wavelet methods.

The description of the method in section 2 reveals the following key requirements so as

(a) to efficiently insert a new index $\lambda \notin \Lambda$ (**GROW**),

(b) to efficiently erase an index $\lambda \in \Lambda$ (**THRESH**),

(c) for a given index λ to have fast access to the corresponding wavelet coefficient $\mathbf{u}_\lambda$,

(d) to sort the indices both with respect to the size of the wavelet coefficients and some “natural” ordering of the indices (which will be discussed in detail later),

(e) to efficiently perform a loop over all indices $\lambda \in \Lambda$ for iteratively solving the linear system $\boldsymbol{A}_\Lambda \mathbf{u}_\Lambda = \boldsymbol{f}_\Lambda$ in order to determine the Galerkin solution $\mathbf{u}_\Lambda$.

At this stage it is already clear that we cannot use any kind of (classical) matrix–vector structures since due to (a), (b), and (d) allocation and deallocation would destroy the efficiency of the method. Note that due to (d) and (e) it is not sufficient to use only a clever way of storing the coefficients as, e.g., provided by *hash tables.* We also need some kind of ordering for the indices. Hence, sophisticated sorted lists seem to suggest themselves.

Moreover, in order to keep the implementation feasible and reusable we are also interested in

(f) separating the adaptive algorithm from the particular choice of the wavelet basis Ψ on Ω. In particular, all geometric information on Ω will be stored in Ψ, whereas the adaptive algorithm uses Ψ only as a parameter (the adaptive method should work for all instances of Ψ);

(g) making the use of our data structures as easy as possible.

The issues (a)–(e) can be realized by using certain data structures that are already provided by the C++-*Standard Template Library* (STL) [47, 48]. Let us describe these structures first before we detail the application to our adaptive wavelet scheme.

3.1. Generic programming. The STL provides, among other things, a huge collection of *generic* classes. This means that these data structures may have various parameters in the form of other classes. In C++ such *container classes*, i.e., classes containing some elements of (almost) arbitrary type, are realized by *templates.* Particular examples of container classes are *key-based* data structures that seem to us best suited to our demands. Moreover, the STL provides certain algorithms that operate on the containers. These algorithms turn out to be important for task (e).

The STL classes `map` and `multimap` depend on three parameters, namely, a `key`, a `value`, and a binary relation `compare`. Both classes provide a sophisticated kind of ordered list of objects indexed by a class `key` and having a certain `value`. The sorting is done with respect to `compare`. The difference of the two classes is that the `compare` relation for `map` has to be total whereas this is not required for `multimap`.

Let us review the efficiency properties of these STL classes. Let N be the number of keys in the list and n the number of elements with the same key ($n = 1$ for `map` and $n \geq 1$ for `multimap`). Then, the average complexity to insert, find, and erase an element is at most $\mathcal{O}(\log(N))$, whereas sorting the whole map is at most $\mathcal{O}(N\ \log(N))$. This means that requirements (a)–(d) are already fulfilled as long as we are able to create appropriate instances for `key`, `value`, and `compare` for our adaptive wavelet method.

The requirement (e), namely, the efficient performance of a loop over all indices $\lambda \in \Lambda$ is guaranteed by the separation of data and algorithms, which is the second main concept of generic programming. So if we have the data structure modeling a wavelet index, a generic algorithm operating on this structure can be used.

3.2. Realization of the wavelet bases. It now remains to find an appropriate realization for the wavelet bases Ψ on Ω. The realization of the adaptive method urges us to realize wavelet bases with sophisticated properties as detailed in section 2.2. It is meanwhile understood how to construct such bases with the desired properties for essentially all cases of interest [13, 14, 18, 29, 30, 32].

We will be dealing with domains that can be written as the disjoint union of smooth parametric images $\Omega_i = \kappa_i(\Box)$ of the unit cube $\Box := (0,1)^d$, i.e., $\bar{\Omega} = \bigcup_{i=1}^M \bar{\Omega}_i$. Then, each ψ_λ, $\lambda \in \nabla$, restricted to Ω_i is the image through κ_i of a linear combination of tensor product wavelets $\hat{\psi}_\mu$ on $\Box$, i.e.,

$$\psi_\lambda|_{\Omega_i}(x) = \sum_{\mu \in S(i,\lambda)} \gamma_{\mu,i}\, \hat{\psi}_\mu((\kappa_i)^{-1}(x)),$$

where $S(i,\lambda)$ is a suitable set of indices of the form $\mu = (j, \boldsymbol{k})$, $j := |\lambda|$, and $\gamma_{\mu,i}$ are suitable (matching) coefficients independent of j (in order to ensure global continuity). To be specific, each wavelet on $\Box$ has the following representation:

$$\hat{\psi}_\mu(\hat{x}) = \prod_{\nu=1}^{d} \vartheta_{\mu_\nu}(\hat{x}_\nu),$$

where each index μ_ν is of the form $\mu_\nu = (j, e_\nu, k_\nu)$ and

$$\vartheta_{(j,e_\nu,k_\nu)} := \begin{cases} \xi_{j,k_\nu} & \text{if } e_\nu = 0, \\ \eta_{j,k_\nu} & \text{if } e_\nu = 1. \end{cases}$$

Here ξ_{j,k_ν} and η_{j,k_ν} are scaling functions and wavelets on $[0,1]$, respectively.

Data structure: `index`. From what has been said so far, it is clear that λ can be viewed as a quadruple $(j, p, \boldsymbol{e}, \boldsymbol{k})$, where p denotes the number of the patch, $j \geq j_0$ the level. The vectors $\boldsymbol{k} = (k_1, \ldots, k_d)$ and $\boldsymbol{e} = (e_1, \ldots, e_d)$ contain the indices k_ν and the types e_ν of the univariate scaling or wavelet functions, respectively, as explained above. Of course, the valid ranges for these parameters are determined by the choice of the wavelet bases on $[0, 1]$.

Next, we define the above-mentioned "natural" ordering of the indices that is adequate for handling the geometry and the multiresolution spaces. For two indices $\lambda = (j, p, \boldsymbol{e}, \boldsymbol{k})$, $\lambda' = (j', p', \boldsymbol{e}', \boldsymbol{k}')$ we define $\lambda < \lambda'$ if $(j, p, \boldsymbol{e}, \boldsymbol{k}) < (j', p', \boldsymbol{e}', \boldsymbol{k}')$ in the usual lexicographic ordering of these vectors.

Finally, the particular construction of Ψ depends on two parameters, namely, the choice of the wavelet bases on the interval and the geometry of Ω which—besides the choice of the construction method—determines the coefficients $\gamma_{\mu,i}$. Following the philosophy of separating algorithm and data, we kept these two data as parameters.

Hence, the construction of an `index` amounts to the computation of the chosen wavelet bases on the interval and then to the setup of the whole basis Ψ in the above form. It is important to stress the fact that this construction has to be done only *once and for all* in a preprocessing step and only for the minimum level, say, j_0. All information for higher levels $j > j_0$ can be deduced from that on j_0.

Particular choices and used software. As wavelet bases on Ω (which in our tests will be an interval for $d = 1$ or an L-shaped domain for $d = 2$) we choose the construction from [29], but the other variants would work in the same way. First numerical results of wavelet methods on the L-shaped domain can be found in [15]. The routines for constructing the wavelets we have used in this paper have been implemented by Vorloeper and are described in detail in [50].

For the wavelet systems on $[0, 1]$ we choose biorthogonal B-spline wavelets on the interval as constructed in [27, 31]. A detailed description of C++-software for constructing wavelets on the interval can be found in [9]. It is based on the *Multilevel Library* presented in [7] and [8].

3.3. Computation of matrix and right-hand-side entries. Based upon the above-described data structures, the adaptive algorithm in section 2 has been realized. In particular, all routines described in the appendix have been implemented.

For the fast matrix–vector multiplication, we have to identify the set

$$\Lambda(\lambda, J) := \Big\{\nu = (j', k') \in \nabla : \operatorname{supp} \psi_\lambda \cap \operatorname{supp} \psi_\nu \neq \emptyset,\ |j - j'| \leq J\Big\}, \tag{3.1}$$

which is determined by (2.14). For our example of a second order differential operator this set contains for a given wavelet ψ_λ the indices of those wavelet functions, whose support intersects the support of ψ_λ along with a cut-off criterion of level differences. These are exactly the nonvanishing entries of the stiffness matrix in the corresponding range of level differences.

In one dimension, this is a technical but straightforward task since explicit formulas for the supports of the wavelets are available. The technicalities are due to the modifications of functions near the end points of the interval. Since the supports of these functions are not given by the same formula as for the interior functions, several cases have to be considered.

In two dimensions, the situation is much more involved. This is mainly due to the fact that also wavelet functions on different patches interact. The corresponding information can be deduced from member functions of `index` so that the realization of the corresponding routine can still be done independently of these parameters.

It remains to address the following two issues:

(i) the computation of the right-hand-side data, i.e., L_2-inner products of the given function f and wavelets as in (2.4), and

(ii) the computation of the entries of the stiffness matrix. This means the evaluation of the bilinear form $a(\psi_\lambda, \psi_\nu)$ for any active indices λ, ν.

As for (i), the capability of computing all relevant wavelet coefficients with respect to the dual basis with sufficient accuracy depends on what kind of information on f is available. In fact, f as part of the model set up by the user involves in many cases simple data. Recently significant progress has been made in rendering the computation of $\mathbf{f}_\eta$ very efficient under reasonably general assumptions on f [4, 5, 11].

The second issue (ii) is practically much more delicate. When dealing with piecewise polynomial wavelets and constant coefficient differential operators as in the example below the requirement of computing the entries at unit cost can certainly be met. When nonconstant coefficients or nontrivial parametric mappings are involved this is by far less obvious. In order to be able to validate the convergence behavior of the scheme at a possibly early stage we employ here a provisional strategy of precomputing the relevant matrix entries via a high level scaling function representation. Of course this module is to be exchanged by more advanced strategies along the following possible lines. Schemes for the adaptive computation of products (of derivatives) of scaling and wavelet functions (also possibly multiplied by a given function) have recently been developed in [11].

A different strategy aiming at optimal computational complexity in the present context is proposed in [6]. There a suitable approximation process provides approximations to the whole summands in (2.25) simultaneously following the lines of [33].

3.4. Conclusions. Let us conclude by comparing the properties of our data structures with the tasks that have been identified at the beginning of this section. The use of STL data structures already ensures that the requirements (a)–(e), and also (f), are in fact met. Moreover, since we made heavy use of existing STL classes, the implementation could be done in a reasonable amount of time. Finally, since our data structures are actually special cases of STL classes, their handling coincides with the corresponding STL classes. These, however, are well documented [48] (also online [47]) so that an easy application is guaranteed. This means, in particular, that also other types of wavelet bases can easily be included only by providing the appropriate interfaces. This holds for other wavelet bases both on the interval and on Ω.

It is clear from the above discussions that a full exploitation of the conceptual power requires new algorithmic ingredients that largely had to be designed from scratch. Specifically, for the problem-dependent part, little can be borrowed from existing software. Consequently, not all parts could be brought to a mature state yet. Nevertheless, we think that the current summary of affairs will provide valuable guidelines for further developments.

4. Numerical tests. In spite of the much wider principal scope of applicability, we confine the numerical tests to second order elliptic boundary value problems in one and two spatial dimensions. In particular, this allows us to exploit special Besov regularity results for the validation of the schemes especially in situations where adaptivity is expected to pay off most. Moreover, we consider only a simple L-shaped domain and refer to [43] for further tests and examples on general two-dimensional domains with piecewise smooth boundary. However, to study the dependence on parameters we include Helmholtz-type problems for which the energy space differs more and more from H^1 when the viscosity parameter decreases.

Bearing in mind that adaptivity is expected to pay off in the presence of singularities, in section 4.1 we consider first a one-dimensional model problem where a nearly singular behavior of the solution is caused only by a strong gradient of the right-hand side. Unlike earlier studies of wavelet schemes we do not confine the discussion to periodic problems.

The second class of examples in section 4.2 is concerned with more sophisticated problems on nonsmooth domains in $\mathbb{R}^2$. In these cases, there also occur singularities which are not generated by the right-hand sides but by the shape of the domain. To our knowledge, such examples have not been studied before in the wavelet context. Therefore, the quantitative performance of the scheme should be very instructive.

4.1. One-dimensional examples. As a first simple example we consider the second order boundary value problem

$$-\frac{d^2u}{dx^2} = f \quad \text{on } \Omega = (0,1), \qquad u(0) = u(1) = 0. \tag{4.1}$$

The choice of the right-hand side will be discussed later. The variational formulation of (4.1) reads

$$\langle u', v' \rangle = \langle f, v \rangle \quad \text{for all } v \in H_0^1(0,1). \tag{4.2}$$

For the treatment of (4.2) the wavelet basis has to satisfy (2.5) at least for $H = H^1(0,1)$. Therefore we choose cardinal B-spline wavelets on the interval that are induced by the generators

$$\varphi := N_m(x), \quad \tilde{\varphi} := \tilde{N}_{m,\tilde{m}}(x), \qquad m, \tilde{m} \geq 2, \tag{4.3}$$

on the real line [16].

Due to (4.2), the trial functions have to satisfy homogeneous Dirichlet boundary conditions. To facilitate a possibly sparse representation of the right-hand side with respect to the dual basis, we employ so-called *complementary boundary conditions* (cf. [31]), i.e., the primal functions fulfill homogeneous Dirichlet boundary conditions whereas the dual functions remain unconstrained.

Since here $H = H_0^1(0,1)$, the diagonal matrix in (2.5) can be chosen as $\omega_\lambda := 2^{|\lambda|}$. For the validity of norm equivalences of the form (2.5) in the presence of boundary conditions, see [31].

4.1.1. Compressibility of the stiffness matrix. To be able to interpret the results, we have to identify first the *range of asymptotic optimality* permitted by the above choice of bases. Theorem 2.2 implies that the asymptotic behavior depends on the compressibility of the matrix $\mathbf{A}$. In particular, the parameter s^* defined in (2.13) provides a range where optimality is guaranteed. However, in the present situation this criterion is too weak. In fact, we know from [22] that the parameter σ in (2.11) must satisfy $t + \sigma < \gamma$ where γ bounds the Sobolev regularity of the wavelets. In this case we have $t = 1$ and $\gamma = m - 1/2$, which gives $\sigma = m - 3/2$, i.e., $s^* = m - 2$. Therefore, in the case $\varphi = N_2$, we end up with $s^* = 0$, which is clearly useless. Of course, the condition (2.13) is only sufficient, and in this case a more detailed analysis of the compression properties is necessary. In fact, for our special case, we have the following sharper result.

Lemma 4.1. *Let* $\mathbf{A}$ *denote the stiffness matrix to* (4.1) *obtained by B-spline wavelets of order* m *as basis functions. Then for any* $\epsilon > 0$ *the following compression estimate holds:*

$$\|\mathbf{A} - \mathbf{A}_J\| \lesssim 2^{-J(m-3/2-\epsilon)}, \quad \textit{i.e.,} \quad \mathbf{A} \in \mathcal{A}_s \textit{ for all } s < m - 3/2. \tag{4.4}$$

Proof. Equation (4.4) can be established directly by using a version of the *Schur lemma*: If for the matrix $\mathbf{B} = (b_{\lambda,\lambda'})_{\lambda,\lambda'\in\mathcal{J}}$ there is a sequence $\omega_\lambda, \lambda \in \mathcal{J}$ and a positive constant c such that

$$\sum_{\lambda'\in\mathcal{J}} |b_{\lambda,\lambda'}|\omega_{\lambda'} \leq c\omega_\lambda \quad \text{and} \quad \sum_{\lambda\in\mathcal{J}} |b_{\lambda,\lambda'}|\omega_\lambda \leq c\omega_{\lambda'}, \quad \lambda, \lambda' \in \mathcal{J}, \tag{4.5}$$

then $\|\mathbf{B}\| \leq c$. We want to use (4.5) for the sequence $\omega_\lambda = 1$ for all $\lambda \in \mathcal{J}$. Let us briefly sketch the arguments. The first step is to estimate the entries in the stiffness matrix corresponding to (4.2). Ignoring for the moment the boundary effects, recalling that derivatives of wavelets are again wavelets (see, e.g., [26]), and using the vanishing moment property of wavelets, we obtain for any polynomial $P_{\lambda'}$ on $\Omega_{\lambda'}$ of degree $< m - 1$ and $j' \geq j$

$$\begin{aligned}\left\langle \frac{d\psi_\lambda}{dx}, \frac{d\psi_{\lambda'}}{dx} \right\rangle &= \left\langle \frac{d\psi_\lambda}{dx} - P_{\lambda'}, \frac{d\psi_{\lambda'}}{dx} \right\rangle \leq \left\| \frac{d\psi_\lambda}{dx} - P_{\lambda'} \right\|_{L_2(\Omega_{\lambda'})} \left\| \frac{d\psi_{\lambda'}}{dx} \right\|_{L_2} \\ &\lesssim \left\| \frac{d\psi_\lambda}{dx} - P_{\lambda'} \right\|_{L_2(\Omega_{\lambda'})} 2^{j'}.\end{aligned}$$

Since $\frac{d}{dx}\psi_\lambda \in H^s,\ s < m - 3/2$, a classical Whitney-type estimate therefore yields

$$\begin{aligned}\left\langle \frac{d\psi_\lambda}{dx}, \frac{d\psi_{\lambda'}}{dx} \right\rangle &\lesssim 2^{j'}2^{-j'(m-3/2-\epsilon)} \left| \frac{d\psi_\lambda}{dx} \right|_{H^{m-3/2-\epsilon}} \lesssim 2^{j'}2^{-j'(m-3/2-\epsilon)}|\psi_\lambda|_{H^{m-1/2-\epsilon}} \\ &\lesssim 2^{j'}2^{-j'(m-3/2-\epsilon)}2^{j(m-1/2-\epsilon)} \lesssim 2^{(j-j')(m-3/2-\epsilon)}2^{j+j'},\end{aligned}$$

so that, taking the preconditioning matrix $\mathbf{D}$ into account, we get

$$|a_{\lambda,\lambda'}| \lesssim 2^{(j-j')(m-3/2-\epsilon)}, \qquad j' \geq j. \tag{4.6}$$

The case $j' < j$ can be treated analogously,

$$|a_{\lambda,\lambda'}| \lesssim 2^{(j'-j)(m-3/2-\epsilon)}, \qquad j' < j. \tag{4.7}$$

Moreover, the vanishing moment property of wavelets implies that the only nonvanishing entries $a_{\lambda,\lambda'}$ correspond to the wavelets $\psi_{\lambda'}$ whose supports intersect the singular support of ψ_λ. It can be shown that the number of these entries does not depend on the refinement level. Consequently, we obtain

$$\sum_{|\lambda'|=j'} |a_{\lambda,\lambda'}| \lesssim 2^{-|j-j'|(m-3/2-\epsilon)}. \tag{4.8}$$

According to (2.14) and (4.5), we have to show that

$$\sum_{|j-j'|>J} \sum_{|\lambda'|=j'} |a_{\lambda,\lambda'}| \lesssim 2^{-J(m-3/2-\epsilon)}. \tag{4.9}$$

Let us again first consider the case $j' > j$. By using (4.8), we obtain

$$\begin{aligned}\sum_{j'-j>J} \sum_{|\lambda'|=j'} |a_{\lambda,\lambda'}| &\lesssim \sum_{j'=j+J}^{\infty} 2^{(j-j')(m-3/2-\epsilon)} \lesssim 2^{j(m-3/2-\epsilon)}2^{-(J+j)(m-3/2-\epsilon)} \\ &\lesssim 2^{-J(m-3/2-\epsilon)}.\end{aligned} \tag{4.10}$$

The case $j' \leq j$ can be treated analogously. The second condition in (4.5) can be checked in a similar fashion, and hence (4.4) is established. □

4.1.2. Fast matrix–vector multiplication. In view of the pivotal role of the approximate fast matrix–vector multiplication, we present first some tests of this ingredient. The error estimate (2.26) indicates that the approximation power of the fast matrix–vector multiplication is determined by the parameter s^* which, according to (4.4), is given by $s^* = m - 3/2$. In Figure 4.1, the error $\|\mathbf{Av} - \mathbf{w}_j\|_{\ell_2}$ is plotted in a logarithmic scale for different choices of m and $\tilde{m}$ in (4.3). Since the slopes do not change when keeping m fixed and varying $\tilde{m}$, we display only one particular choice of $\tilde{m}$ for each m. In Table 4.1, the values of s^*, the observed values s of the numerical tests, and the ratio s^*/s are shown. We deduce that the estimate (4.4) derived above is in fact sharp.

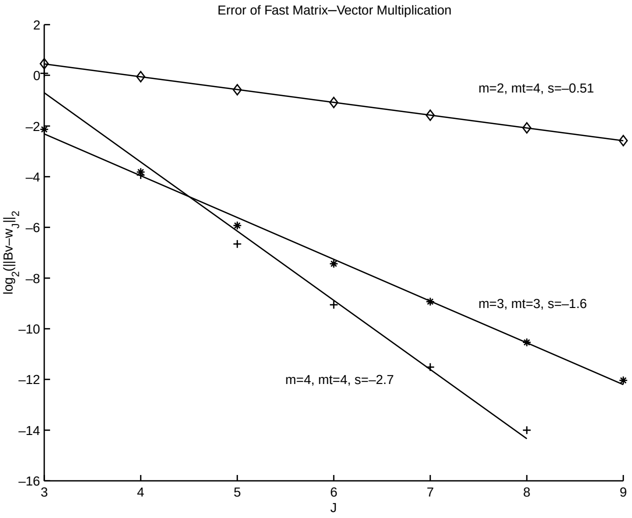


FIG. 4.1. *The slope of error reduction in the fast matrix–vector multiplication.*

TABLE 4.1
Expected and observed slopes for the error of the fast matrix–vector multiplication.

d	s^*	s	Ratio
2	0.5	0.51	0.98
3	1.5	1.60	0.94
4	2.5	2.70	0.93

4.1.3. Example 1. In our first test case, we choose f corresponding to the solution

$$u(x) = 4\frac{e^{ax}-1}{e^a-1}\left(1-\frac{e^{ax}-1}{e^a-1}\right), \tag{4.11}$$

which satisfies the boundary conditions. For our tests, we choose $a = 5.0$. The exact solution and the right-hand side are shown in Figure 4.2.

We first compare the computed Galerkin solution with the best N-term approximation. Recall that the convergence rate realized by the adaptive scheme is limited by the compressibility range of the matrix $\boldsymbol{A}$ which by Lemma 4.1 is $s^* = m - 3/2$. Note that this parameter is related to the *regularity* of the wavelets. This effect has

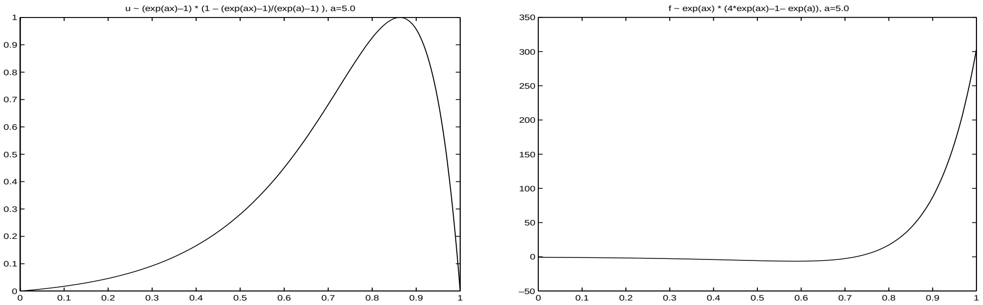


FIG. 4.2. *The exact solution and the right-hand side for the one-dimensional example.*

no counterpart in uniform discretizations. We expect that the performance of the adaptive scheme can be improved by increasing the smoothness of the wavelet basis. At this point, we want to emphasize that in contrast to linear schemes the maximal approximation order of the adaptive scheme is *not* limited by the polynomial exactness of the multiresolution analysis but apparently by its regularity. Of course, using B-spline wavelets both quantities are not far apart from each other. In Figure 4.3 we have displayed in a logarithmic scale the error for both the best N-term approximation (lines) and the adaptive algorithm (dots) as N increases. We see that both errors show almost the same behavior. We also see that the performance of the algorithm improves as the smoothness of the wavelet bases increases.

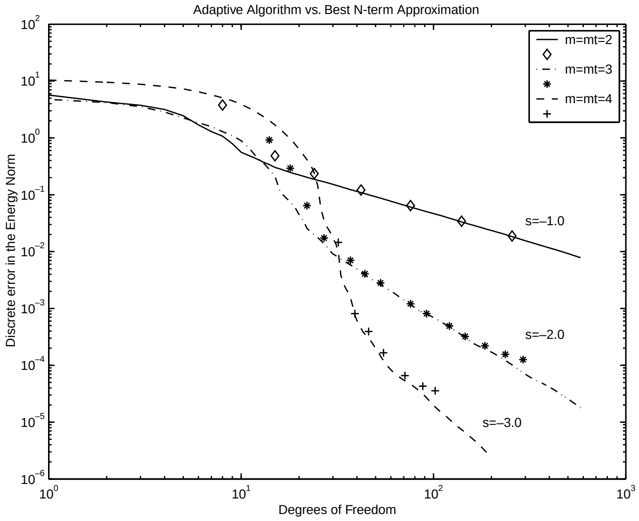


FIG. 4.3. *Comparison between best N-term approximation and adaptive algorithm for the one-dimensional example.*

Note that the solution to our test problem has pointwise smoothness of arbitrary order and therefore belongs to every Sobolev space. Thus, by the discussion in section 2.9, in principle, a uniform refinement scheme provides the same asymptotic order of approximation for functions in $H^{m-1/2}(0,1)$, i.e., in $\mathcal{O}(N^{-(m-3/2)})$. However, as pointed out at the end of section 2.9, a quantitative gain of efficiency is still possible if the $H^{m-1/2}$-norm of the solution u is large when compared with the corresponding norm in the Besov space $B^{m-1/2}_{\tau^*}(L_{\tau^*}(\Omega))$, $1/\tau^* = m - 3/2 + 1/2 = m - 1$. In our example, these norms indeed differ significantly as can be seen in Table 4.2. We have estimated the Besov norms by employing the norm equivalences (2.6), i.e., by computing weighted sequence norms of wavelet expansions. Figure 4.2 explains the difference between Sobolev and Besov norms. The boundary layer of the solution u increases the Sobolev norm but has less influence on the (weaker) Besov norm.

TABLE 4.2
Sobolev and Besov norms of the exact solution (4.11) *to the one-dimensional example.*

m	$H^{m-1/2}$	$B^{(m-1/2)}_{(m-1)^{-1}}(L_{(m-1)^{-1}})$
1.5	6.73	6.73
2	39.4	14.5
2.5	240	47.8
3	1617	275

We have compared the adaptive scheme with uniform refinement in order to get an impression of the effect of the different sizes of the above-mentioned norms. The results of these numerical tests are shown in Figure 4.4. They confirm indeed the gain of efficiency for adaptive schemes for this example as indicated by the comparison of Sobolev and Besov norms in Table 4.2. In spite of the significant quantitative gain, the slopes of the curves are indeed the same reflecting equal asymptotic behavior, as predicted by the above comments.

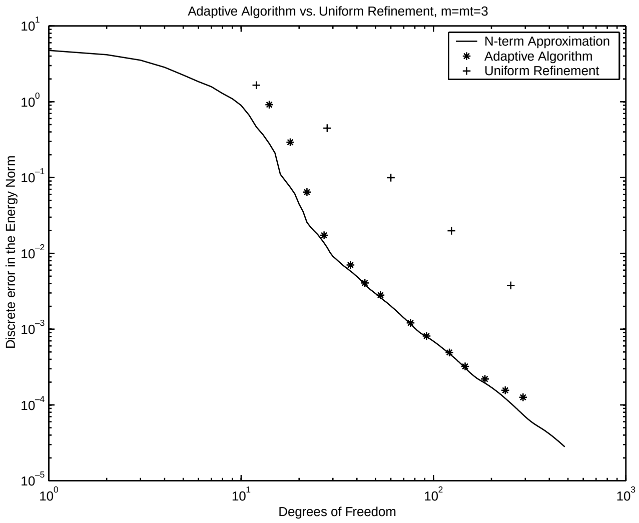


FIG. 4.4. *Error of the adaptive algorithm and of a uniform refinement for the one-dimensional example.*

In Figure 4.5, we have plotted the sets of wavelet indices that correspond to the adaptively chosen wavelets. These wavelets are sometimes called *active*. One observes that the adaptive algorithm in fact recognizes the strong gradient of the solution u according to the boundary layer and adds wavelet coefficients locally in these regions.

4.2. Two-dimensional examples. We have tested our algorithm for the Poisson and the Helmholtz equations on an L-shaped domain Ω in $\mathbb{R}^2$,

$$-\triangle u = f \quad \text{on} \quad \Omega, \quad u|_{\partial\Omega} = 0, \quad \text{and} \quad -\varepsilon\triangle u + u = f \quad \text{on} \quad \Omega, \quad u|_{\partial\Omega} = 0. \tag{4.12}$$

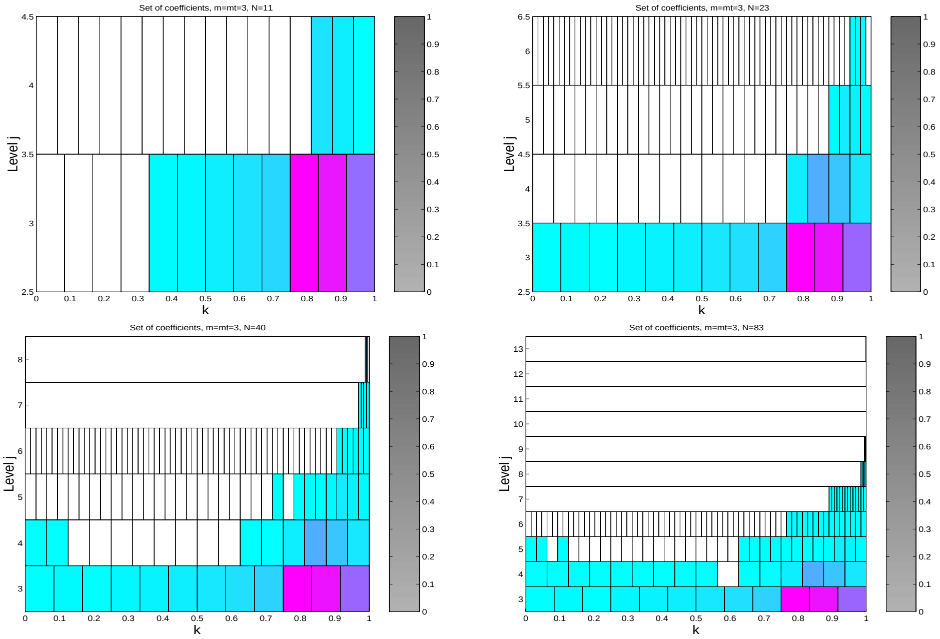


FIG. 4.5. *The sets of active indices* Λ_1*,* Λ_3*,* Λ_4*, and* Λ_6 *for the one-dimensional example.*

As already said, these problems are interesting because the solution may exhibit singularities solely caused by the shape of the domain even for smooth right-hand sides. Moreover, we can test the robustness of the scheme with respect to decreasing viscosity. We refer to [35, 44, 40] for the classical theory of singularities on polygonal domains, and to [20, 21] for an analysis of these singularities in terms of Besov smoothness.

4.2.1. Example 1: The Poisson equation, singular solution with smooth right-hand side. Introducing polar coordinates (r, θ), we define the right-hand side f in (4.12) corresponding to the solution

$$u(r,\theta) := \zeta(r) r^{2/3} \sin(\tfrac{2}{3}\theta), \tag{4.13}$$

where $\zeta \in C^\infty(\Omega)$ is a truncation function defined by

$$\zeta(r) := \frac{w(\frac{3}{4}-r)}{w(r-\frac{1}{2})+w(\frac{3}{4}-r)} \quad \text{with} \quad w(r) := \begin{cases} e^{-1/r^2} & \text{if} \quad r>0, \\ 0 & \text{else.} \end{cases}$$

The function u is shown in Figure 4.6. Observe that u is harmonic in the vicinity of the reentrant corner. Therefore the right-hand side f is C^∞ everywhere in Ω which confirms that the singularity of u is generated by the shape of the domain; see Figure 4.6. To our knowledge so far adaptive wavelet schemes have not been applied yet to situations of this type.

Due to its singularity at the reentrant corner, the solution u is contained in $H^\alpha(\Omega)$ only for $\alpha < 5/3$. Consequently, uniform grids yield at best an H^1-convergence rate $N^{-1/3}$. On the other hand, it is well known that an optimal mesh refinement allows us to restore the rate $N^{-1/2}$ in the case of affine finite elements (which would be

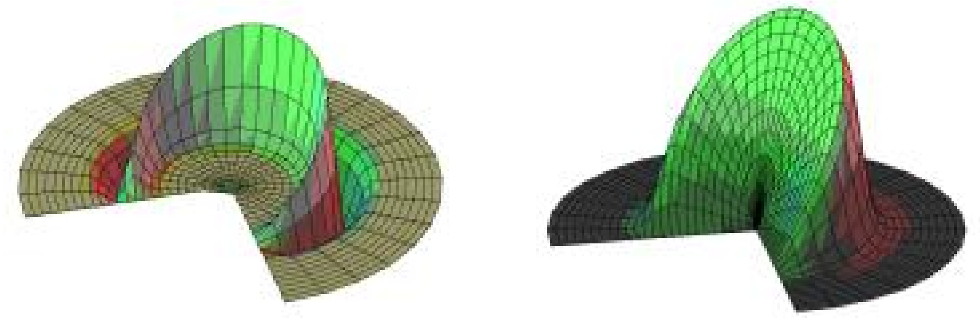


FIG. 4.6. *The right-hand side (left) and exact solution (right) to our model problem.*

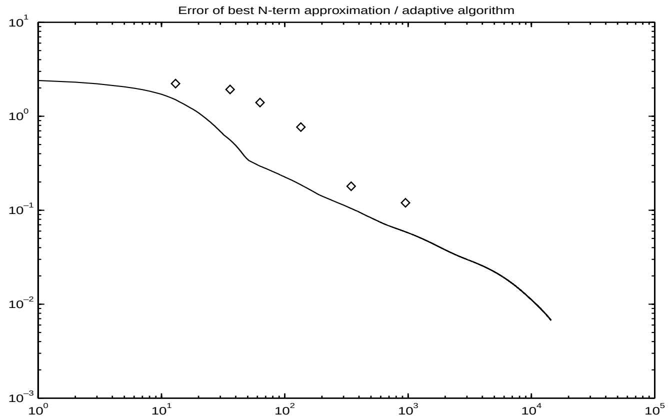


FIG. 4.7. *Comparison of best N-term approximation and adaptive algorithm for the first two-dimensional example.*

obtained by a uniform method if there were no singularity) and one can easily check that the same holds for the best N-term approximation with piecewise affine wavelets.

Again, we have tested the performance of the adaptive algorithm in this regard. In Figure 4.7, the errors are shown for increasing N. The continuous line corresponds to the best N-term approximation. Aside from the quantitatively very good matching, we observe the optimal rate $N^{-1/2}$. Similar to the one-dimensional examples it turns out that the approximation rate of the adaptive scheme can be significantly increased by using smoother and higher order wavelet bases; see section 4.2.4.

As for the quantitative refinement history, starting with the empty set, the residual in the first step is influenced only by the wavelet coefficients of the right-hand side. These wavelet coefficients are small near the vertex due to the fact that u is harmonic there. Now in the next steps the adaptive scheme has to track down the singularity at the reentrant corner and add wavelet coefficients there.

Figure 4.8 displays both the approximate solution and the error to the exact solution. At first, coefficients are added to reduce the error where strong gradients are induced by the right-hand side, whereas in the subsequent iterations the error is reduced in the vicinity of the reentrant corner, so that after five iterations the error is equally distributed. Figure 4.9 shows the sets of active wavelet coefficients

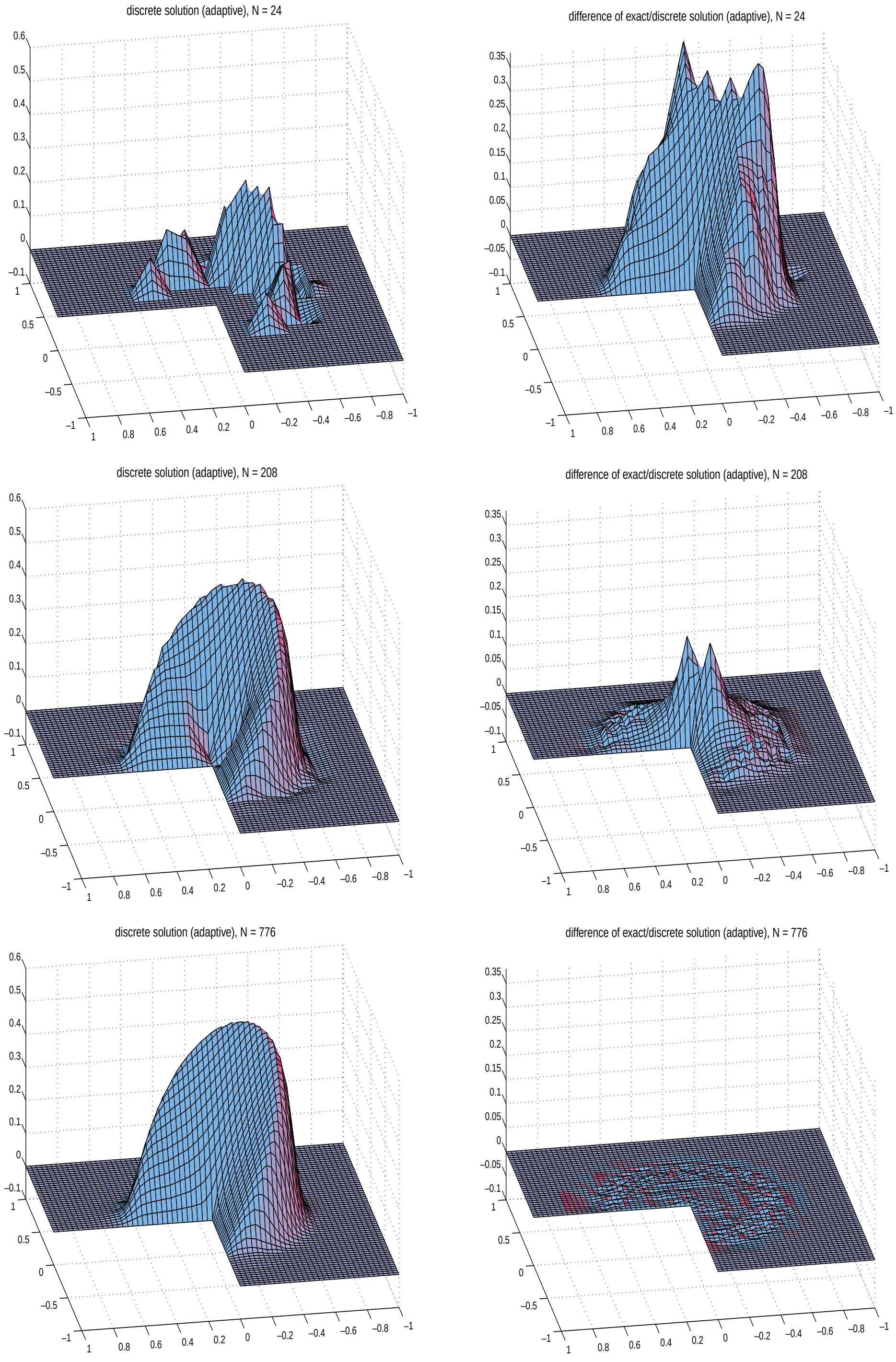


FIG. 4.8. *The first, third, and fifth approximate solutions and the differences to the exact solution for the first two-dimensional example.*

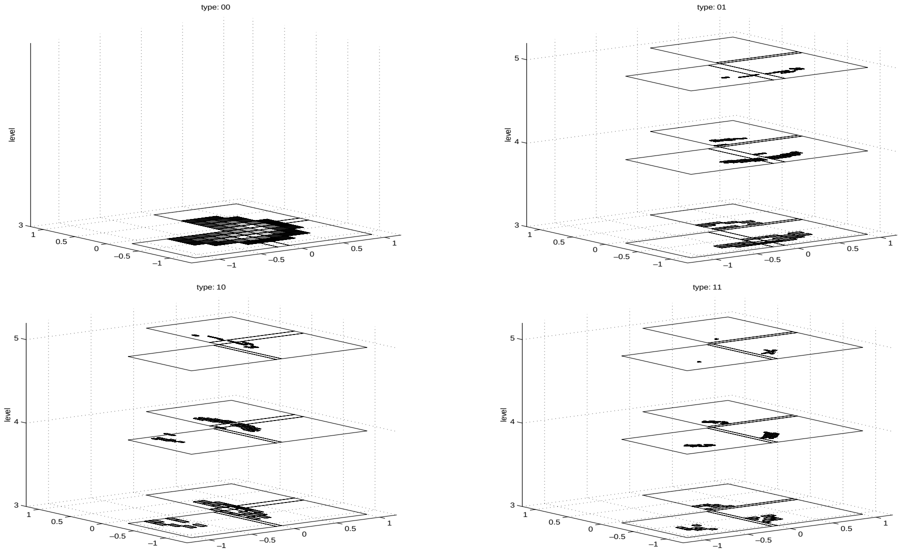


FIG. 4.9. *Index sets for the fifth iteration, first two-dimensional example.*

corresponding to the fifth iteration of the adaptive algorithm. The first picture shows the set of coefficients corresponding to the scaling functions, whereas in the remaining three pictures we have plotted the three different types of wavelets (corresponding to $\boldsymbol{e}$ in section 3.2) separately. It is shown in detail which coefficients are added on each refinement level. We see that the symmetry of the exact solution is reflected by the similarity of the pictures in the upper right and lower left corner. These two pictures correspond to tensor product functions of wavelet/generator and generator/wavelet type, respectively.

4.2.2. Example 2: The Poisson equation, singular right-hand side. In order to test the quantitative performance of the scheme in the presence of singularities that are induced by the right-hand side, we have constructed, with the aid of a dual relation of the norm equivalences (2.5), a special right-hand side with an isolated singularity "far away" from the reentrant corner. This is obtained by setting all wavelet coefficients of f equal to zero except the ones in the vicinity of one chosen point. At this specific point, we choose the coefficients as $\langle f, \psi_\lambda \rangle = 2^{(|\lambda|/2)}$. Consequently, the resulting functional f is not contained in L_2 but only in $H^s,\ s < -1/2$. An approximation of a typical example consisting of the wavelets on the first six levels is plotted in Figure 4.10.

We expect that the singularity of the right-hand side is reflected by the corresponding solution. In Figure 4.11, we have depicted the Galerkin approximations with respect to two different iterations of the adaptive algorithm. We see that the solution indeed behaves as expected, i.e., the singularity of the right-hand side shows up in a somewhat smeared shape. In this case, the right-hand side dominates the influence of the domain. Nevertheless, for the last iterations, the singularity caused by the domain (as in our first example) can again be seen, but its impact is almost negligible.

4.2.3. Example 3: The Helmholtz equation. We include a test for the Helmholtz equation (4.12) to see how the scheme copes with different small values of ε.

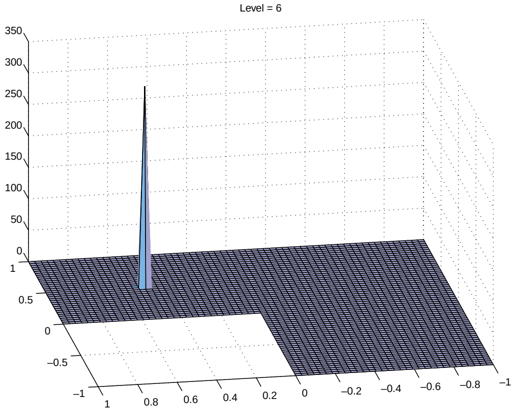


FIG. 4.10. *A right-hand side with a sharp singularity, second two-dimensional example.*

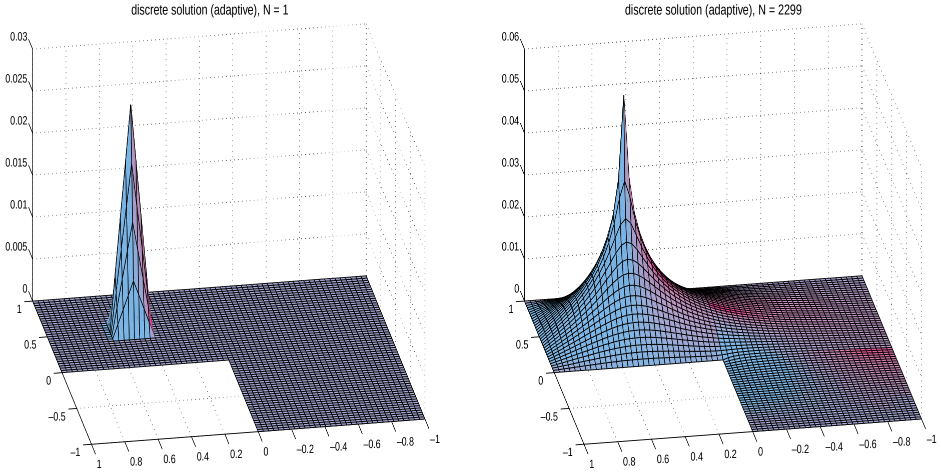


FIG. 4.11. *The first and the last computed Galerkin approximation for a right-hand side with a singularity, second two-dimensional example.*

In this case, if the right-hand side does not vanish at the boundary, the solution has a boundary layer. For $\varepsilon = 10^{-5}$ we used the sum of the singular solution and a constant as the right-hand side. The solution to this problem is displayed in Figure 4.12. The main observations can be summarized as follows: The proper ε-dependent diagonal scaling explained in section 2.2 leads to uniformly bounded condition numbers of about the same size as for the Poisson equation uniformly in ε. Note that for decreasing ε the energy space changes from H^1 toward L_2. Since we are employing Riesz bases for both spaces this has no adverse effect on the adaptive scheme, not even in quantitative terms. Again, the adaptive algorithm produces approximate solutions with an error of the same order as the best N-term approximation and very good quantitative matching.

4.2.4. Comparison with finite element schemes. We have also compared our method with a well-established finite element scheme, i.e., with the software package Differential Equation Analysis Library (`deal.II`) which was developed at the IWR in Heidelberg [2]. This package seemed to be suitable for a meaningful comparison

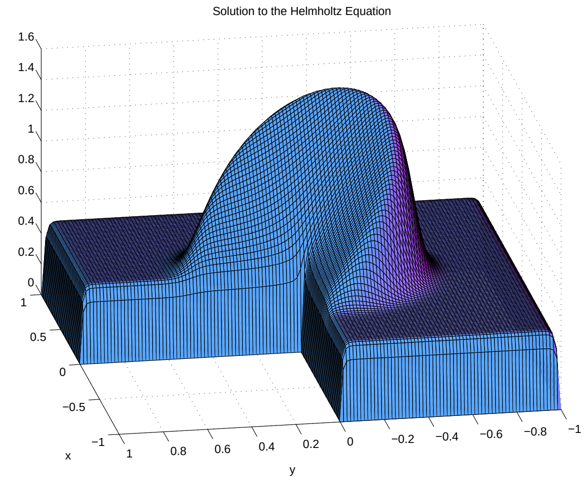


Fig. 4.12. *Solution for the third two-dimensional example.*

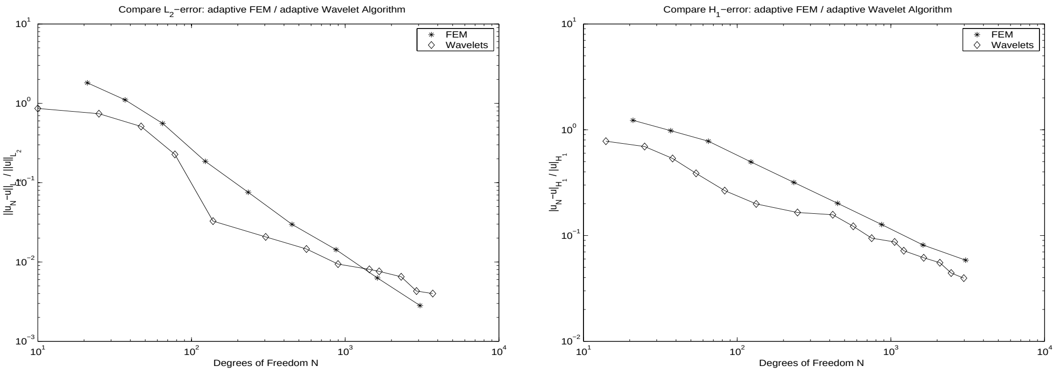


Fig. 4.13. *Rel. L_2- and H_1-error for piecewise affine functions.*

for the following reasons:

- it is written as an open source code;
- it realizes a fully adaptive finite element scheme, using a refinement strategy based on a posteriori error estimators developed by Kelly et al. [41];
- it is based on rectangular partitions.

In Figure 4.13 the results are displayed for the Poisson equation with right-hand side as in Example 1 and piecewise affine finite elements and wavelets. The wavelet scheme seems to produce a somewhat better accuracy for the same number of degrees of freedom, but the asymptotic behavior appears to be the same. In Figure 4.14, a similar comparison is shown for piecewise quadratic functions. Now the wavelet scheme exhibits a better performance in a much more pronounced way. The predicted asymptotically better compressibility of the wavelet representation starts to show up quantitatively. This difference seems to be even more enhanced for the Helmholtz problem; see Figure 4.15. Because of the different nature of the energy space, we show here both the L_2- as well as the H^1-error. This seems to confirm that finite element preconditioners are usually better suited for H^1 problems than for L_2-like problems, while the wavelet scheme shows the same robust behavior in both cases.

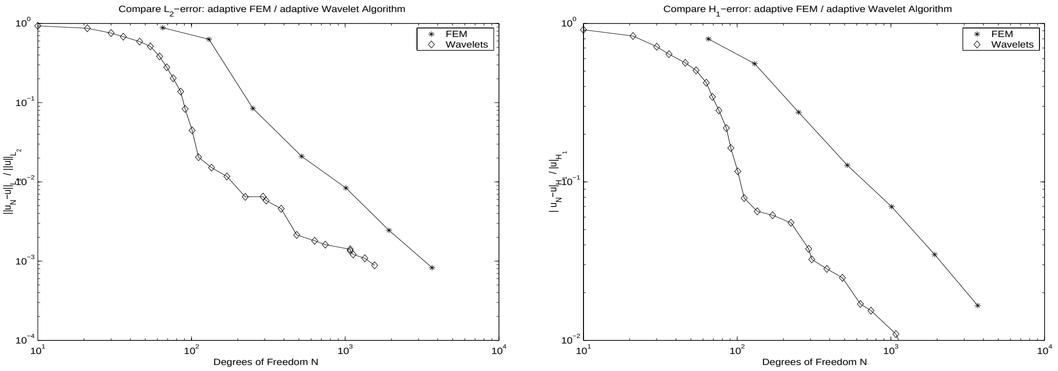


FIG. 4.14. *Rel. L_2- and H_1-error for piecewise quadratic functions.*

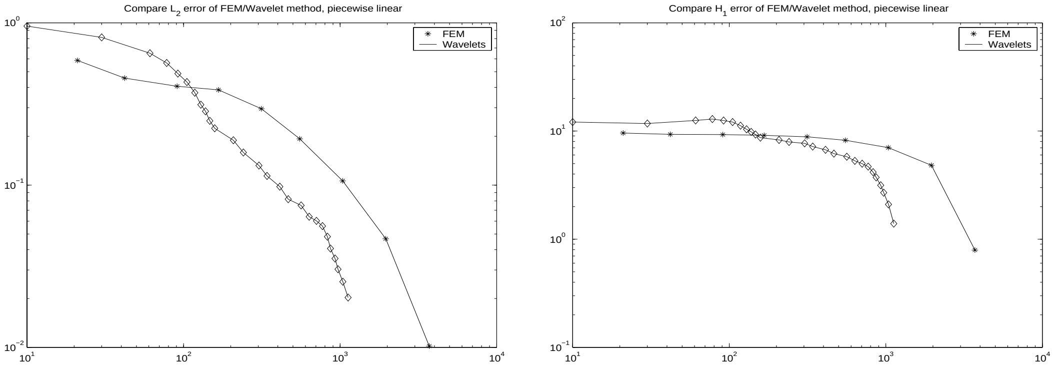


FIG. 4.15. *Rel. L_2- and H_1-error for the Helmholtz example.*

Appendix. The adaptive algorithm ALGORITHM^c. In this appendix, we detail all steps that are needed for the fully computable version **ALGORITHM^c** described in section 2. Let us first recall some constants that will be needed here. Let c_1, c_2 be the constants appearing in the first equivalence of (2.17), i.e.,

$$c_1\|\mathbf{v}\|^2_{\ell_2(\nabla)} \le \|\mathbf{v}\|^2 \le c_2\|\mathbf{v}\|^2_{\ell_2(\nabla)}, \tag{A.1}$$

and let $\kappa \ge \|\boldsymbol{A}\|\,\|\boldsymbol{A}^{-1}\|$ be an estimate for the condition number of $\boldsymbol{A}$, e.g., $\kappa := c_2c_1^{-1}$.

We start with an initialization step that also sets global constants and parameters:

INIT.

(i) *Fix some γ in $(0,1]$.*

(ii) *Determine F according to $\|\mathbf{f}\|_{\ell_2} \le F$.*

(iii) *Determine q_1, q_2, q_3, and q_4 such that*

$$\left(\frac{q_3}{c_2} + \frac{2(1+\gamma)(q_1+q_2+q_3)}{\gamma c_1}\right) \le q_4, \quad \left(q_4\sqrt{\kappa} + \frac{q_3}{c_2}\right) \le \frac{1}{10},$$

$$(q_1+q_2+q_3+\kappa^{-1}q_3) \le \frac{c_1}{20}.$$

- *Set $q_0 := \kappa^{1/2} + q_3/c_2$, $\theta := \sqrt{1-\frac{c_1}{4c_2}}$, and $\bar{\theta} := 1 - \frac{1}{6\kappa}$.*

(iv) *Define*

$$K := K(\kappa, \theta) := \left\lceil \frac{\log 20\kappa}{|\log \theta|} \right\rceil + 1. \tag{A.2}$$

We first describe the main algorithm **ALGORITHM**^{c}. The subroutines will be detailed below.

ALGORITHM^c.

(i) **Initialization.** *Let $\varepsilon > 0$ be the target accuracy. Perform* **INIT**. *Set $\Lambda := \emptyset$, $\mathbf{v} = \mathbf{0}$, and $\delta := F$.*
(ii) *If $\delta \le c_1^{1/2}\varepsilon$, accept $\mathbf{u}(\varepsilon) := \mathbf{v}$, $\Lambda(\varepsilon) := \Lambda$ as the final solution and STOP. Otherwise,* **NPROG** $[\Lambda, \mathbf{v}, \delta, \mathbf{f}] \to (\hat\Lambda, \hat{\mathbf{v}}, \hat{\mathbf{r}})$.
(iii) *If $\|\hat{\mathbf{r}}\|_{\ell_2} + (q_1 + q_2 + (1 + \kappa^{-1})q_3)\delta \le c_1\varepsilon$ accept $\bar{\mathbf{u}}(\varepsilon) := \bar{\mathbf{u}}_{\Lambda^k}$, $\Lambda(\varepsilon) = \Lambda^k$ as the solution, where $\bar{\mathbf{u}}_{\Lambda^k}$, $\Lambda(\varepsilon) = \Lambda^k$ are the last outputs of* **NGROW** *in* **NPROG** *before thresholding.*
Otherwise, replace δ by $\delta/2$, $\mathbf{v}$ by $\hat{\mathbf{v}}$, and Λ by $\hat\Lambda$ and go to (ii).

We now list the routines called above in recursive order of their call.

NPROG $[\Lambda, \mathbf{v}, \delta, \mathbf{f}] \to (\hat\Lambda, \hat{\mathbf{v}}, \hat{\mathbf{r}})$. *Given a set Λ, an approximation $\mathbf{v}$ to the exact solution $\mathbf{u}$ of $\boldsymbol{A}\mathbf{u} = \mathbf{f}$ whose support is contained in Λ and such that $\|\mathbf{v} - \mathbf{u}\|_{\ell_2} \le \delta$.*

(i) *Apply* **GALERKIN** $[\Lambda, \mathbf{v}, \delta, q_3\delta/c_2] \to \bar{\mathbf{u}}_\Lambda$. *Set $\Lambda^0 := \Lambda$, $\bar{\mathbf{u}}_{\Lambda^0} := \bar{\mathbf{u}}_\Lambda$, $k := 0$.*
(ii) *Apply* **NGROW** $[\Lambda^k, \bar{\mathbf{u}}_{\Lambda^k}, q_1\delta, q_2\delta, \mathbf{f}, \gamma] \to (\Lambda^{k+1}, \mathbf{r}^k)$.
(iii) *If $\|\mathbf{r}^k\|_{\ell_2} \le c_1\delta/20$ or $k = K$ defined in* (A.2) *go to* (iv).
Otherwise, apply **GALERKIN** $[\Lambda^{k+1}, \bar{\mathbf{u}}_{\Lambda^k}, q_0\delta, q_3\delta/c_2] \to \bar{\mathbf{u}}_{\Lambda^{k+1}}$. *Replace k by $k+1$, Λ^k by Λ^{k+1}, $\bar{\mathbf{u}}_{\Lambda^k}$ by $\bar{\mathbf{u}}_{\Lambda^{k+1}}$, and go to* (ii).
(iv) *Apply* **NCOARSE** $[\bar{\mathbf{u}}_{\Lambda^k}, 2\delta/5] \to (\hat\Lambda, \hat{\mathbf{v}})$, *set $\hat{\mathbf{r}} := \mathbf{r}^k$ and STOP.*

GALERKIN $[\Lambda, \mathbf{v}, \delta, \eta] \to \bar{\mathbf{u}}_\Lambda$.

(i) *Apply* **INRESIDUAL** $[\mathbf{v}, \Lambda, \mathbf{f}, \frac{c_1\eta}{6}, \frac{c_1\eta}{6}] \to \mathbf{r}$. *If* $\min\left\{\bar\theta\delta, c_1^{-1}\|\mathbf{r}\|_{\ell_2} + \eta/3\right\} \le \eta$, *define the output $\bar{\mathbf{u}}_\Lambda$ to be $\mathbf{v}$ and STOP, else go to* (ii).
(ii) *Set $\bar{\mathbf{v}}' := \mathbf{v} - \frac{1}{c-2}\mathbf{r}$. Replace $\mathbf{v}$ by $\bar{\mathbf{v}}'$, δ by $\bar\theta\delta$, and go to* (i).

INRESIDUAL $[\mathbf{v}, \Lambda, \mathbf{f}, \eta_1, \eta_2] \to \mathbf{r}$.

(i) *Apply* **APPLY** $\boldsymbol{A}_\Lambda\,[\mathbf{v}, \eta_1] \to \mathbf{w}$;
(ii) *Apply* **NCOARSE** $[\boldsymbol{P}_\Lambda\mathbf{f}, \eta_2] \to \mathbf{g}$.
(iii) *Set $\mathbf{r} := \mathbf{g} - \mathbf{w}$.*

APPLY A $[\eta, \mathbf{v}] \to (\mathbf{w}, \Lambda)$.

(i) *Sort the nonzero entries of the vector $\mathbf{v}$ and form the vectors $\mathbf{v}_{[0]}$, $\mathbf{v}_{[j]} - \mathbf{v}_{[j-1]}$, $j = 1, \dots, \lfloor \log N \rfloor$ with $N := \#\mathrm{supp}\,\mathbf{v}$. Define $\mathbf{v}_{[j]} := \mathbf{v}$ for $j > \log N$.*
(ii) *Compute $\|\mathbf{v}\|^2_{\ell_2(\nabla)}$, $\|\mathbf{v}_{[0]}\|^2_{\ell_2}$, $\|\mathbf{v}_{[j]} - \mathbf{v}_{[j-1]}\|^2_{\ell_2(\nabla)}$, $j = 1, \dots, \lfloor \log N \rfloor + 1$.*
(iii) *Set $k = 0$.*
  (a) *Compute $R_k := c_2\|\mathbf{v} - \mathbf{v}_{[k]}\|_{\ell_2(\nabla)} + a_k\|\mathbf{v}_{[0]}\|_{\ell_2(\nabla)} + \sum_{j=0}^{k-1} a_j\|\mathbf{v}_{[k-j]} - \mathbf{v}_{[k-j-1]}\|_{\ell_2(\nabla)}$.*
  (b) *If $R_k \le \eta$ stop and output k; otherwise replace k by $k+1$ and return to* (a).
(iv) *For the output k of* (iii) *and for $j = 0, 1, \dots, k$, compute the nonzero entries in the matrices $\boldsymbol{A}_{k-j}$ (see Proposition* 1*) which have a column index in common with one of the nonzero entries of $\mathbf{v}_{[j]} - \mathbf{v}_{[j-1]}$.*
(v) *For the output k of* (iii), *compute $\mathbf{w}_k$ as in* (2.25) *and take $\mathbf{w}(\mathbf{v}, \eta) := \mathbf{w}_k$ and $\Lambda = \mathrm{supp}\,\mathbf{w}$.* □

NCOARSE $[\mathbf{w}, \eta] \to (\Lambda, \bar{\mathbf{w}})$.

(i) *Define $N := \#(\mathrm{supp}\,\mathbf{w})$ and sort the nonzero entries of $\mathbf{w}$ into decreasing order. Thereby one obtains the vector $\lambda^* := \lambda^*(\mathbf{w}) = (\lambda_1, \lambda_2, \dots, \lambda_N)$ of indices* *which gives the decreasing rearrangement* $\mathbf{w}^* = (|\mathbf{w}_{\lambda_1}|, |\mathbf{w}_{\lambda_2}|, \ldots, |\mathbf{w}_{\lambda_N}|)$ *of the nonzero entries of* $\mathbf{w}$*; then compute* $\|\mathbf{w}\|_{\ell_2}^2 = \sum_{i=1}^N |\mathbf{w}_{\lambda_i}|^2$.

(ii) *For* $k = 1, 2, \ldots$, *form the sum* $\sum_{j=1}^k |\mathbf{w}_{\lambda_j}|^2$ *in order to find the smallest value* k *such that this sum exceeds* $\|\mathbf{w}\|_{\ell_2}^2 - \eta^2$. *For this* k, *define* $\bar{K} := k$ *and set* $\Lambda := \{\lambda_j\ ;\ j = 1, \ldots, \bar{K}\}$*; define* $\bar{\mathbf{w}}$ *by* $\bar{w}_\lambda := w_\lambda$ *for* $\lambda \in \Lambda$ *and* $\bar{w}_\lambda := 0$ *for* $\lambda \notin \Lambda$.

NGROW $[\Lambda, \bar{\mathbf{u}}_\Lambda, \xi_1, \xi_2, \mathbf{f}, \gamma] \to (\tilde{\Lambda}, \mathbf{r})$. *Given an initial approximation* $\bar{\mathbf{u}}_\Lambda$ *to the Galerkin solution* $\mathbf{u}_\Lambda$ *supported on* Λ.

(i) *Apply* **NRESIDUAL** $[\bar{\mathbf{u}}_\Lambda, \Lambda, \mathbf{f}, \xi_1, \xi_2] \to (\Lambda^r, \mathbf{r})$.

(ii) *Apply* **NCOARSE** $[\mathbf{r}, \sqrt{1-\gamma^2}\|\mathbf{r}\|_{\ell_2}] \to (\Lambda^c, \boldsymbol{P}_{\Lambda^c}\mathbf{r})$ *and define* $\tilde{\Lambda} := \Lambda \cup \Lambda^c$.

NRESIDUAL$[\mathbf{v}, \Lambda, \mathbf{f}, \eta_1, \eta_2] \to (\mathbf{r}, \tilde{\Lambda})$.

(i) *Apply* **APPLY A** $[\mathbf{v}, \eta_1] \to (\mathbf{w}, \Lambda_1)$.

(ii) *Apply* **NCOARSE** $[\mathbf{f}, \eta_2] \to (\mathbf{g}, \Lambda_2)$.

(iii) *Set* $\mathbf{r} := \mathbf{w} - \mathbf{g}$ *and* $\tilde{\Lambda} := \operatorname{supp} \mathbf{r} \subseteq \Lambda_1 \cup \Lambda_2$.

Acknowledgments. The authors feel grateful to Yvon Maday for giving them the opportunity to participate in the European Summer School. They also want to thank Jürgen Vorloeper for putting his software concerning the composite wavelet basis at their disposal. Furthermore, they want to thank Nicole Mulders, who helped them produce some of the pictures. This paper was partially written when the last author was in residence at the Istituto di Analisi Numerica del C.N.R. in Pavia, Italy.

REFERENCES

[1] I. Babuška and W. C. Rheinboldt, *A posteriori error estimates for finite element methods*, Int. J. Numer. Math. Engrg., 12 (1978), pp. 1597–1615.

[2] W. Bangert and G. Kanschat, *Concepts for Object-Oriented Finite Element Software—The* `deal.II` *Library*, Preprint 99–43 (SFB 359), IWR, Heidelberg, Germany, 1999.

[3] R. E. Bank and A. Weiser, *Some a posteriori error estimators for elliptic partial differential equations*, Math. Comp., 44 (1985), pp. 283–301.

[4] A. Barinka, T. Barsch, S. Dahlke, and M. Konik, *Some remarks on quadrature formulas for refinable functions and wavelets*, ZAMM Z. Angew. Math. Mech., to appear.

[5] A. Barinka, T. Barsch, S. Dahlke, M. Konik, and M. Mommer, *Quadrature formulas for refinable functions and wavelets* II: *Error analysis*, J. Comput. Anal. Appl., to appear.

[6] A. Barinka, S. Dahlke, and W. Dahmen, *Adaptive Application of Operators in Standard Representation*, in preparation.

[7] A. Barinka, T. Barsch, K. Urban, and J. Vorloeper, *The Multilevel Library: Software Tools for Multiscale Methods and Wavelets, Version* 1.0*, Documentation*, RWTH Aachen, IGPM Preprint 156, 1998.

[8] T. Barsch, A. Kunoth, and K. Urban, *Towards object oriented software tools for numerical multiscale methods for PDEs using wavelets*, in Multiscale Methods for Partial Differential Equations, W. Dahmen, A. Kurdila, and P. Oswald, eds., Academic Press, New York, 1997, pp. 383–412.

[9] T. Barsch and K. Urban, *Software tools for using wavelets on the interval for the numerical solution of operator equations*, in Concepts of Numerical Software, W. Hackbusch and G. Wittum, eds.,University of Kiel, Germany, 2000, pp. 13–25.

[10] R. Becker, C. Johnson, and R. Rannacher, *Adaptive error control for multigrid finite element methods*, Computing, 55 (1995), pp. 271–288.

[11] S. Bertoluzza, C. Canuto, and K. Urban, *On the adaptive computation of integrals of wavelets*, Appl. Numer. Math., 34 (2000), pp. 13–38.

[12] F. A. Bornemann, B. Erdmann, and R. Kornhuber, *A posteriori error estimates for elliptic problems in two and three space dimensions*, SIAM J. Numer. Anal., 33 (1996), pp. 1188–1204.

[13] C. Canuto, A. Tabacco, and K. Urban, *The wavelet element method, part* I: *Construction and analysis*, Appl. Comput. Harmon. Anal., 6 (1999), pp. 1–52.

[14] C. Canuto, A. Tabacco, and K. Urban, *The wavelet element method, part* II: *Realization and additional features in* 2*d and* 3*d*, Appl. Comp. Harmon. Anal., 8 (2000), pp. 123–165.

[15] C. Canuto, A. Tabacco, and K. Urban, *Numerical solution of elliptic problems by the wavelet element method*, in ENUMATH 1997, H. G. Bock et al., eds., World Scientific, Singapore, 1998, pp. 17–37.

[16] A. Cohen, I. Daubechies, and J.-C. Feauveau, *Biorthogonal bases of compactly supported wavelets*, Comm. Pure Appl. Math., 45 (1992), pp. 485–560.

[17] A. Cohen, W. Dahmen, and R. DeVore, *Adaptive wavelet methods for elliptic operator equations: Convergence rates*, Math. Comp., 70 (2001), pp. 22–75.

[18] A. Cohen and R. Masson, *Wavelet adaptive method for second order elliptic problems, boundary conditions and domain decomposition*, Numer. Math., 86 (2000), pp. 193–238.

[19] M. Costabel, *Boundary integral operators on Lipschitz domains: Elementary results*, SIAM J. Math. Anal., 19 (1988), pp. 613–626.

[20] S. Dahlke, *Wavelets: Construction Principles and Applications to the Numerical Treatment of Operator Equations*, Habilitation thesis, Shaker Verlag, Aachen, 1997.

[21] S. Dahlke, *Besov regularity for elliptic boundary value problems on polygonal domains*, Appl. Math. Lett., 12 (1999), pp. 31–36.

[22] S. Dahlke, W. Dahmen, R. Hochmuth, and R. Schneider, *Stable multiscale bases and local error estimation for elliptic problems*, Appl. Numer. Math., 23 (1997), pp. 21–47.

[23] S. Dahlke and R. DeVore, *Besov regularity for elliptic boundary value problems*, Comm. Partial Differential Equations, 22 (1997), pp. 1–16.

[24] S. Dahlke, R. Hochmuth, and K. Urban, *Adaptive wavelet methods for saddle point problems*, M2AN Math. Model. Numer. Anal., 34 (2000), pp. 1003–1022.

[25] W. Dahmen, *Stability of multiscale transformations*, J. Fourier Anal. Appl., 2 (1996), pp. 341–361.

[26] W. Dahmen, *Wavelet and multiscale methods for operator equations*, in Acta Numer. 6, Cambridge University Press, Cambridge, UK, 1997, pp. 55–228.

[27] W. Dahmen, A. Kunoth, and K. Urban, *Biorthogonal spline-wavelets on the interval—stability and moment conditions*, Appl. Comp. Harmon. Anal., 6 (1999), pp. 132–196.

[28] W. Dahmen, S. Prössdorf, and R. Schneider, *Multiscale methods for pseudo-differential equations on smooth manifolds*, in Wavelets: Theory, Algorithms, and Applications, C. K. Chui, L. Montefusco, and L. Puccio, eds., Academic Press, New York, 1994, pp. 385–424.

[29] W. Dahmen and R. Schneider, *Composite wavelet bases for operator equations*, Math. Comp., 68 (1999), pp. 1533–1567.

[30] W. Dahmen and R. Schneider, *Wavelets on manifolds.* I. *Construction and domain decomposition*, SIAM J. Math. Anal., 31 (1999), pp. 184–230.

[31] W. Dahmen and R. Schneider, *Wavelets with complementary boundary conditions—function spaces on the cube*, Results Math., 34 (1998), pp. 255–293.

[32] W. Dahmen and R. Stevenson, *Element-by-element construction of wavelets satisfying stability and moment conditions*, SIAM J. Numer. Anal., 37 (1999), pp. 319–325.

[33] W. Dahmen, R. Schneider, and Y. Xu, *Nonlinear functionals of wavelet expansions—adaptive reconstruction and fast evaluation*, Numer. Math., 86 (2000), pp. 49–101.

[34] I. Daubechies, *Ten Lectures on Wavelets*, CBMS-NSF Regional Conference Series in Applied Math. 61, SIAM, Philadelphia, 1992.

[35] M. Dauge, *Elliptic Boundary Value Problems on Corner Domains*, Lecture Notes in Math. 1341, Springer-Verlag, Berlin, 1988.

[36] R. DeVore, *Nonlinear approximation*, in Acta Numer. 7, Cambridge University Press, Cambridge, UK, 1998, pp. 51–150.

[37] R. DeVore and V. Popov, *Interpolation spaces and nonlinear approximation*, in Function Spaces and Approximation, M. Cwikel, J. Peetre, Y. Sagher, and H. Wallin, eds., Lecture Notes in Math. 1302, Springer-Verlag, New York, 1988, pp. 191–205.

[38] W. Dörfler, *A convergent adaptive algorithm for Poisson's equation*, SIAM J. Numer. Anal., 33 (1996), pp. 1106–1124.

[39] M. Frazier and B. Jawerth, *Decomposition of Besov spaces*, Indiana Univ. Math. J., 34 (1985), pp. 777–799.

[40] P. Grisvard, *Singularities in Boundary Value Problems*, Res. Notes in Appl. Math. 22, Springer-Verlag, New York, 1992.

[41] D. W. Kelly, J. R. Gago, O. C. Zienkiewicz, and I. Babuška, *A posteriori error analysis and adaptive processes in the finite element method*, J. Numer. Methods Engrg., 19 (1983), pp. 1593–1619.

[42] D. Jerison and C. E. Kenig, *The inhomogeneous Dirichlet problem in Lipschitz domains*, J. Funct. Anal., 130 (1995), pp. 161–219.

[43] M. Jürgens, *Adaptive Wavelet–Verfahren auf allgemeinen Gebieten*, Master's thesis, RWTH Aachen, 2001.

[44] V. A. Kondrat'ev, *Boundary-value problems for elliptic equations in domains with conical or angular points*, Trans. Moscow Math. Soc., 16 (1967), pp. 227–313.

[45] V. G. Mazya, *Boundary integral equations*, in Analysis IV, Encyclopaedia Math. Sci. 27, G. Mazya and S. M. Nikol'skii, eds., Springer-Verlag, Berlin, Heidelberg, 1991.

[46] Y. Meyer, *Wavelets and Operators*, Cambridge Stud. Adv. Math. 37, Cambridge University Press, Cambridge, UK, 1992.

[47] Silicon Graphics Inc., *Standard Template Library Programmer's Guide*, 1996, http://www.sgi.com/Technology/STL/.

[48] B. Stroustup, *The C++–Programming Language*, 3rd ed., Addison–Wesley, Reading, MA, 1997.

[49] R. Verfürth, *A posteriori error estimation and adaptive mesh-refinement techniques*, J. Comput. Appl. Math., 50 (1994), pp. 67–83.

[50] J. Vorloeper, *Multiskalenverfahren und Gebietszerlegungsmethoden*, Master's thesis, RWTH Aachen, 1999.

SIAM J. Sci. Comput.
Vol. 23, No. 3, pp. 940–960

A GLOBALLY CONVERGENT NEWTON-GMRES SUBSPACE METHOD FOR SYSTEMS OF NONLINEAR EQUATIONS[*]

STEFANIA BELLAVIA[†] AND BENEDETTA MORINI[†]

Abstract. Newton–Krylov methods are variants of inexact Newton methods where the approximate Newton direction is taken from a subspace of small dimension. Here we introduce a new hybrid Newton-GMRES method where a global strategy restricted to a low-dimensional subspace generated by GMRES is performed. The obtained process is consistent with preconditioning and with matrix-free implementation. Computational results indicate that our proposal enhances the classical backtracking inexact method.

Key words. nonlinear systems, Krylov subspace methods, generalized minimal residual, inexact Newton, backtracking

AMS subject classification. 65H10

PII. S1064827599363976

1. Introduction.

We consider the numerical solution of systems of nonlinear equations

$$F(x) = 0, \tag{1.1}$$

where $F : \mathbb{R}^n \mapsto \mathbb{R}^n$ is continuously differentiable. Inexact Newton methods [6] are iterative processes for solving (1.1) that result in the following scheme:

Let x_0 be given.
For $k = 0$ until "convergence" do:
    Find some $\bar{\eta}_k \in [0, 1)$ and $\bar{s}_k$ that satisfy

$$\|F(x_k) + F'(x_k)\bar{s}_k\| \le \bar{\eta}_k \|F(x_k)\|, \tag{1.2}$$

    Set $x_{k+1} = x_k + \bar{s}_k$,

where $F'(x)$ is the Jacobian matrix and the *forcing term* $\bar{\eta}_k$ is used to control the level of accuracy. Clearly, these methods are variants of Newton's method in which at each iteration the Newton equation

$$F'(x_k)s = -F(x_k), \qquad k \ge 0, \tag{1.3}$$

is solved only approximately. Local convergence analysis for inexact Newton methods [6] shows that if x_0 is sufficiently close to a solution x^* of (1.1) and the η_k's are uniformly bounded away from one, then the sequence $\{x_k\}$ converges to x^*.

When the dimension n of the problem is large, a relevant class of inexact Newton methods is constituted by Newton–Krylov methods [1], [2], [3], [4], [6], [13]: a Krylov subspace projection method is used to compute $\bar{s}_k$ satisfying (1.2). An attractive

[*] Received by the editors November 16, 1999; accepted for publication (in revised form) April 5, 2001; published electronically August 29, 2001. This work was supported by Istituto Nazionale di Alta Matematica (INDAM) "F. Severi" and by MURST, Rome, Italy, through "Cofinanziamenti Programmi di Ricerca Scientifica di Interesse Nazionale."
http://www.siam.org/journals/sisc/23-3/36397.html

[†] Dipartimento di Energetica "S. Stecco," University of Florence, via C. Lombroso 6/17, 50134 Florence, Italy (bellavia@ciro.de.unifi.it, morini@ciro.de.unifi.it).

feature of Krylov methods is that they require only the action of the Jacobian F' on a vector v. Moreover, for an appropriately chosen scalar ϵ, this action can be approximated by finite differences [1], [4], [13]

$$F'(x)v \sim \frac{F(x+\epsilon v) - F(x)}{\epsilon}, \tag{1.4}$$

giving rise to a process that is referred to as "matrix-free."

In order to enhance convergence from arbitrary starting points, hybrid globally convergent modifications of Newton–Krylov methods have been considered by several authors (see, e.g., [3], [4], [8], [9], [13], [20]): to a Newton–Krylov method one adds either a linesearch procedure or a trust-region technique. Among the inexact Newton methods where a linesearch procedure is used, an inexact Newton backtracking (INB) method was proposed in [8]. It performs backtracking along the inexact Newton step $\bar{s}_k$, and computational results on a large set of test problems have shown its robustness and efficiency [9], [20]. It is worth noting that inexact Newton methods that incorporate a linesearch procedure based on the Armijo condition [14, p. 40] are a special case of INB (see [8]).

Also, in the numerical solution of large-scale nonlinear systems and bound constrained minimization problems by Newton–Krylov methods there has been much research in using the information concerning the low-dimensional subspaces that are built into the process of solving the linear system (see, e.g., [3], [5], [10], [11], [15]).

Here we are concerned with a *Newton-GMRES* method: a Newton–Krylov method where GMRES [21] is used to solve (1.3) approximately. We propose a new hybrid method where a global strategy restricted to a suitable Krylov subspace is performed. Our globalization strategy consists of two parts. The first is the backtracking procedure of the INB method. The second is a backtracking technique along a piecewise linear curve that involves the current search direction $\bar{s}_k$ and an additional direction selected in a properly chosen subspace, which is itself obtained using the information provided by GMRES. Our method is an extension of Newton-GMRES backtracking techniques designed to improve performance when $\bar{s}_k$ is a poor descent direction. We give extensive numerical tests in support of this claim. Further, we point out that our method is consistent with efficient matrix-free implementation and with preconditioning.

The paper is organized as follows. In section 2 the Newton-GMRES algorithm and the INB method are reviewed. In section 3 a hybrid Newton-GMRES paradigm with globally convergent strategy restricted to a subspace is introduced. Further, the new hybrid method is proposed and the convergence analysis is given. Its consistency with restarting and preconditioning procedures is also investigated. Finally, in section 4 we make some comments on the implementation of the new method and give some numerical results.

2. Preliminaries. Throughout this paper, given a smooth real function $f : \mathbb{R}^n \mapsto \mathbb{R}$, its gradient is denoted by ∇f; the vector $\bar{x} = \text{argmin}_{x \in S} f(x)$ represents a minimizer of f in the set $S \subset \mathbb{R}^n$, and the vector (matrix) norm is always the 2-norm. Further, e_j denotes the jth unit vector, with its dimension inferred from the context; $(x)_i$ represents the ith component of the vector x, and the closed ball with center y and radius δ is indicated by $N_\delta(y)$. Finally, concerning iterative methods for solving nonlinear systems (1.1), the term *breakdown* refers to the case in which an iterate can not be determined, and the term *nonlinear iteration* denotes the sequence of steps that are performed to update the current iterate.

2.1. Newton-GMRES method.

GMRES (generalized minimal residual) is an iterative method for solving indefinite systems of linear equations [21]. When used to solve the Newton equation (1.3) approximately, the resulting method is called Newton-GMRES. The application of GMRES to the Newton equation (1.3) is outlined in the following algorithm.

ALGORITHM 1. GNE (GMRES for the kth Newton equation).

```
Let $x_k$, $\bar{\eta}_k$ be given.
1. Choose $s_k^0$. Set $m = 0$.
   Compute $r_k^0 = -F'(x_k)s_k^0 - F(x_k)$, $\beta_k = \|r_k^0\|$, $v_1 = r_k^0/\beta_k$.
2. While $\|r_k^m\| > \bar{\eta}_k\|F(x_k)\|$ do
   GMRES iteration:
      2.1. Set $m = m + 1$.
      2.2. Compute $F'(x_k)v_m$ and
              $h_{i,m} = (F'(x_k)v_m)^T v_i, \quad i = 1, 2, \ldots, m,$
              $v_{m+1} = F'(x_k)v_m - \sum_{i=1}^m h_{i,m}v_i,$
              $h_{m+1,m} = \|v_{m+1}\|,$
              $v_{m+1} = v_{m+1}/h_{m+1,m}.$
           Let $\bar{H}_m \in \mathbb{R}^{(m+1)\times m}$ be the upper Hessenberg matrix whose nonzero
           entries are the coefficients $h_{i,j}$, $i = 1, \ldots, j+1$, for $j = 1, \ldots, m$.
      2.3. Find the vector $y_m \in \mathbb{R}^m$ that solves the least-squares problem
              $\min_{y\in\mathbb{R}^m} \|\beta_k e_1 - \bar{H}_m y\|.$
      2.4. Set $\|r_k^m\| = \|\beta_k e_1 - \bar{H}_m y_m\|$.
3. Define $V_m \equiv [v_1, v_2, \ldots, v_m] \in \mathbb{R}^{n\times m}$ and form
      $s_k^m = s_k^0 + V_m y_m .$
4. Let $\bar{s}_k = s_k^m$.
```

Note that a GMRES iteration corresponds to a single value of m in step 2 of the above algorithm. At each GMRES iteration, s_k^m solves the least-squares problem

$$\min_{s\in s_k^0+K_m} \|F'(x_k)s + F(x_k)\|, \tag{2.1}$$

where K_m is the Krylov subspace:

$$K_m = span\{r_k^0, F'(x_k)r_k^0, (F'(x_k))^2 r_k^0, \ldots, (F'(x_k))^{m-1} r_k^0\}. \tag{2.2}$$

Step 2.2 is the Arnoldi process [13] for the construction of an orthonormal basis $v_1, \ldots, v_m$ of K_m. From this process it follows that [21]

$$F'(x_k)V_m = V_{m+1}\bar{H}_m, \tag{2.3}$$

and so the least-squares problem (2.1) reduces to

$$\min_{y\in\mathbb{R}^m} \|\beta_k e_1 - \bar{H}_m y\|. \tag{2.4}$$

One iterates until the residual vector

$$r_k^m = -F'(x_k)s_k^m - F(x_k)$$

satisfies $\|r_k^m\| \leq \bar{\eta}_k \|F(x_k)\|$, i.e., until $\bar{s}_k = s_k^m$ satisfies the stopping criterion (1.2). Then, the vector $\bar{s}_k$ is used to form the Newton iterate $x_{k+1} = x_k + \bar{s}_k$.

Although from (2.1) the residual norm $\|r_k^m\|$ is nonincreasing with m, GMRES may stagnate, i.e., the residual norm may remain constant for a number of iterations. However, in theory GMRES finds the solution of the linear system in at most n iterations [21]. In practice, because of rounding errors, GMRES is best thought of as an iterative method. In fact, the vectors v_j computed by the Arnoldi process in step 2.2 may become nonorthogonal as a result of accumulated roundoff errors. Consequently, if a large number of GMRES iterations must be performed, more complex implementations of step 2.2 are necessary (see [13] and the references cited therein). A further aspect of the above algorithm we have not yet considered is that the basis of the Krylov subspace K_m must be stored. Namely, in order to perform m iterations one must store m vectors of length n. This may become prohibitive when n is large. Typically a maximum value m_{max} of GMRES iterations is fixed. If after m_{max} GMRES iterations the norm of the residual is still greater than $\bar{\eta}_k \|F(x_k)\|$, GMRES is *restarted* with initial guess s_k^0 equal to $s_k^{m_{max}}$. The convergence of such a procedure is not always guaranteed, but the idea works well in practice [21].

2.2. Backtracking strategies. To enlarge the convergence basin of a locally convergent method, linesearch techniques are often used [7, section 6.5], [18, section 11.2]. They are based upon a globally convergent method for a problem of the form $\min_{x \in \mathbb{R}^n} M(x)$, where M is an appropriately chosen merit function whose global minimum is a zero of F. In these cases, for p a given direction in $\mathbb{R}^n$, the iterate x_{k+1} has the form $x_{k+1} = x_k + \lambda p$, where $0 < \lambda \leq 1$ is such that $M(x_k + \lambda p) < M(x_k)$. The existence of such a λ is ensured if p is a descent direction for M at x_k, i.e., if there exists a $\lambda_0 > 0$ such that $M(x_k + \lambda p) < M(x_k)$ for all $\lambda < \lambda_0$. When M is differentiable this is equivalent to the following condition [7]:

$$\nabla M(x_k)^T p < 0.$$

Proposition 3.3 of [4] shows that if $M = \|F\|^2/2$ or $M = \|F\|$, a vector p is a descent direction for M at x_k if $\|F'(x_k)p + F(x_k)\| < \|F(x_k)\|$. Thus, from (1.2) $\bar{s}_k$ is a descent direction for these choices of M. Also, note that each GMRES iterate s_k^m, $m \geq 0$, satisfying $\|F'(x_k)s_k^m + F(x_k)\| < \|F(x_k)\|$, is a descent direction for M at x_k even if the residual violates $\|r_k^m\| \leq \bar{\eta}_k \|F(x_k)\|$.

To make the linesearch method succeed, the simple decrease $M(x_k + \lambda p) < M(x_k)$ is not sufficient. In fact, to ensure the convergence of $\{x_k\}$ to a minimum of M, at each iteration a "sufficient decrease" of M must be required (see, e.g., [7, pp. 116–120], [8], [19, pp. 36–43]).

In typical linesearch strategies, the step length λ is chosen by using a so-called backtracking approach. The usual convention is to start with $\lambda = 1$ and then to backtrack, i.e., to reduce λ, until M is sufficiently decreased.

Among the backtracking methods, INB [8] is a globally convergent process where the kth iteration of an inexact Newton method is embedded in a backtracking strategy. In what follows, we will restrict our attention to the INB method where GMRES is used to solve the Newton equations approximately. The kth nonlinear iteration of this method can be sketched as follows.

Algorithm 2. NGB (Newton-GMRES with backtracking).

```
Let $x_k$, $\eta_{max} \in (0,1)$, $t \in (0,1)$, $0 < \theta_l < \theta_u < 1$ be given.
1. Choose $\bar{\eta}_k \in [0, \eta_{max}]$.
```

```
2. Apply Algorithm GNE to compute $\bar{s}_k = s_k^m$ such that
        $\|F(x_k) + F'(x_k)\bar{s}_k\| \leq \bar{\eta}_k \|F(x_k)\|$.
3. Perform the **Backtracking Loop (BL)**, i.e.,
   3.1. Set $s_k = \bar{s}_k$, $\eta_k = \bar{\eta}_k$.
   3.2. While $\|F(x_k + s_k)\| > (1 - t(1 - \eta_k))\|F(x_k)\|$ do:
            Choose $\theta \in [\theta_l, \theta_u]$.
            Update $s_k = \theta s_k$ and $\eta_k = 1 - \theta(1 - \eta_k)$.
4. Set $x_{k+1} = x_k + s_k$.
```

In step 3, the following sufficient decrease in the merit function $\|F(x)\|$ is provided:

$$\|F(x_k + s_k)\| \leq (1 - t(1 - \eta_k))\|F(x_k)\|. \tag{2.5}$$

In particular, moving along the direction of the inexact Newton step $\bar{s}_k$, successively shorter steps $s_k = s_k(\eta)$ of the form $s_k = (1 - \eta)\bar{s}_k/(1 - \bar{\eta}_k)$ are selected.

3. NGECB: A hybrid Newton-GMRES method.

Here we present a hybrid inexact method based on NGB. To provide a thorough description of it, we first introduce a new general algorithm that serves as paradigm. In this, the backtracking strategy is augmented with a second global strategy. In particular, we know from [8] that the backtracking loop BL in step 3 of Algorithm NGB terminates after a finite number of steps. However, if for reasonable values of the constants t, θ_l, θ_u, relatively few steps do not suffice to decrease $\|F\|$, instead of insisting on the direction $\bar{s}_k$ we select different directions and try to decrease the value of an appropriate merit function M moving from x_k along these directions. To be more specific, in our proposed method, the number of backtracking steps is limited to a number, N_b, that is fixed in advance. If the linesearch terminates successfully within N_b steps, the method proceeds as usual. Otherwise, a new procedure is invoked to find a different search direction; the intention is to avoid repeated backtracking steps along a poor direction. The relevant aspect of our proposal is that such direction is chosen in a low-dimensional subspace S which involves information already generated during the GMRES iterations. We shall deliberately not specify the form of this set S. In fact, the subspace S depends on whether or not restarting and/or preconditioning are employed in GMRES. Our intention here is to provide a general paradigm; in the following subsections we will specify our choices for the set S.

The kth nonlinear iteration of the hybrid method we propose belongs to the following framework.

General paradigm.

```
Let $x_k$, $\eta_{max} \in (0,1)$, $t \in (0,1)$, $0 < \theta_l < \theta_u < 1$, $N_b \geq 0$ be given.
1. Choose $\bar{\eta}_k \in [0, \eta_{max}]$.
2. Apply Algorithm GNE to compute $\bar{s}_k = s_k^m$ such that
        $\|F(x_k) + F'(x_k)\bar{s}_k\| \leq \bar{\eta}_k \|F(x_k)\|$.
3. Apply the backtracking loop BL and perform at most $N_b$ iterations.
4. If
        $\|F(x_k + s_k)\| \leq (1 - t(1 - \eta_k))\|F(x_k)\|$,
   set $\Delta_k = s_k$. Go to step 6.
5. Form the set $S$ by using information provided by Algorithm GNE and
```

compute $\Delta_k \in S$ such that

$$\nabla M(x_k)^T \Delta_k < 0 \qquad \text{and} \qquad M(x_k + \Delta_k) < M(x_k).$$

6. Set $x_{k+1} = x_k + \Delta_k$.

The above scheme can be viewed as a paradigm for hybrid Newton-GMRES processes since step 5 does not specify the form of the subspace S, the choice of the vector Δ_k, or the merit function M. Note that the simple condition $M(x_{k+1}) < M(x_k)$ is imposed for each k. However, a "sufficient decrease" of M has to be required in order to ensure the convergence of $\{x_k\}$ to a minimum of M. Specific implementations of step 5 in the given paradigm lead to different procedures.

Here we present a Newton-GMRES method where step 5 is performed by using $M = \|F\|$ and a new globalization strategy that will be called ECB (equality curve backtracking) strategy. The resulting method will be denoted the NGECB (Newton-GMRES with ECB) method.

In the presentation of the NGECB method, first we take $s_k^0 = 0$ in GNE and select Δ_k in the Krylov subspace K_m defined by (2.2). Namely, we set $S = K_m$ in step 5 of the general paradigm. The convergence analysis of the new process is given and the modifications needed in the case $s_k^0 \neq 0$ are discussed. Then, the consistency with preconditioning will be analyzed.

3.1. A globalization strategy restricted to $\boldsymbol{K_m}$. We assume that at most m_{max} GMRES iterations are performed in step 2 of Algorithm GNE and that restarting is not employed. Further, we assume that the initial guess $s_k^0 = 0$ is used. If within $m = m_{max}$ GMRES iterations, Algorithm GNE fails to provide a vector $\bar{s}_k = s_k^m$ satisfying (1.2), we continue using the last GMRES iterate. This is done letting $\bar{s}_k = s_k^m$ and $\bar{\eta}_k = \|F(x_k) + F'(x_k)\bar{s}_k\| / \|F(x_k)\|$ after step 2 of the general paradigm. It is worth noting that when $s_k^0 = 0$ is chosen, in Algorithm GNE we have

$$\beta_k = \|F(x_k)\| \qquad \text{and} \qquad F(x_k) = -\beta_k v_1.$$

To devise a reliable global strategy restricted to a subspace S, first we need to identify in S descent directions for $M = \|F\|$ at x_k. In our approach, the choice $S = K_m$ plays a central role since descent directions can be identified in K_m at no additional cost, as follows. Step 3 of Algorithm GNE provides the orthonormal basis V_m for K_m and clearly each vector $\delta \in K_m$ has the form $\delta = V_m y$ for some $y \in \mathbb{R}^m$. As a consequence, the problem of decreasing $M : \mathbb{R}^n \to \mathbb{R}$ is replaced by the problem of decreasing the function $g : \mathbb{R}^m \to \mathbb{R}$ defined by

$$g(y) = M(x_k + V_m y). \tag{3.1}$$

It is easy to determine if $g(y)|_{y=0} = M(x_k)$ can be reduced. In fact, using (2.3) we have

$$F(x_k)^T F'(x_k) V_m = -\beta_k v_1^T V_{m+1} \bar{H}_m = -\beta_k e_1^T \bar{H}_m, \tag{3.2}$$

and

$$\nabla g(y)|_{y=0} = (F'(x_k) V_m)^T F(x_k) / \beta_k = -\bar{H}_m^T e_1.$$

Further, the inexact Newton step $\bar{s}_k = V_m y_m$ is the solution of the least-squares problem (2.4), so that

$$y_m = -\beta_k (\bar{H}_m^T \bar{H}_m)^{-1} \nabla g(y)|_{y=0}. \tag{3.3}$$

Hence, $\nabla g(y)|_{y=0} = 0$ if and only if $y_m = 0$. Since we need a nonzero inexact Newton step, it is crucial to require m_{max} large enough so that $y_m \neq 0$. This way, the existence of descent directions for M at x_k within the subspace K_m is guaranteed. Therefore, we will assume $\bar{s}_k \neq 0$ in the following description of the ECB-strategy.

Finally, it is worth noting that the first row of the Hessenberg matrix $\bar{H}_m$ gives the directional derivatives of M at x_k along the directions v_j. Thus, we determine if v_j is a descent direction at no additional cost. In fact, for any $\delta \in K_m$ the directional derivative $\nabla M(x_k)^T\delta$ is such that

$$\nabla M(x_k)^T\delta = \nabla M(x_k)^T V_m y = -e_1^T \bar{H}_m y, \qquad y \in \mathbb{R}^m.$$

Consequently, recalling that $v_j = V_m e_j$, we get

$$\nabla M(x_k)^T v_j = \nabla g(y)|_{y=0}^T e_j = -e_1^T \bar{H}_m e_j = -h_{1,j}, \quad j = 1, \ldots, m. \tag{3.4}$$

Clearly, v_j is a descent direction if and only if $h_{1,j} > 0$.

We are now ready to specify the strategy we use to form the vector Δ_k in step 5 of the general paradigm. In fact, the merit function $M = \|F\|$ will be used and a sufficient decrease in M will be ensured applying a backtracking strategy along a curve $\sigma_k(\eta)$. The curve $\sigma_k(\eta)$ is such that $\sigma_k(1) = 0$ and each trial step satisfies (1.2) with equality, i.e.,

$$\|F(x_k) + F'(x_k)\sigma_k(\eta)\| = \eta\|F(x_k)\|. \tag{3.5}$$

In order to construct $\sigma_k(\eta)$, we model $g(y) = M(x_k + V_m y)$ at $y = 0$ by the following function:

$$\hat{g}(y) = \|F'(x_k)V_m y + F(x_k)\|,$$

whose steepest descent direction d_m at $y = 0$ is given by

$$d_m = -\nabla\hat{g}(y)|_{y=0} = -\nabla g(y)|_{y=0} = \bar{H}_m^T e_1. \tag{3.6}$$

It is important to note that, by definition of GMRES, $\bar{s}_k = s_k^m$ solves (2.1). Namely, letting $s_k^m = V_m y_m$, the vector $y_m \in \mathbb{R}^m$ minimizes the quadratic model $\hat{g}$ in K_m.

Algorithm 3. ECB-strategy.

```
Let $x_k$, $t \in (0, 1)$ be given. Let $\bar{s}_k$ be the inexact Newton step and $\bar{H}_m$ and $V_m$ be
the matrices defined, respectively, in steps 2.2 and 3 of Algorithm GNE.
1. Choose $\epsilon > 0$, $h_{min} > 0$ and set $\beta_k = \|F(x_k)\|$.
2. Compute $d_m$ using (3.6).
3. Compute $\alpha_d = \text{argmin}_{\alpha\in\mathbb{R}}\ \hat{g}(\alpha d_m)$:   $\alpha_d = \beta_k\,\|d_m\|^2/\|\bar{H}_m d_m\|^2$.
4. Set $z = \alpha_d V_m d_m$.
5. Form $J = \{1 \le j \le m \text{ s.t. } h_{1,j} \ge h_{min}\}$,  $D = \{e_j \text{ s.t. } j \in J\}$.
6. If $D \neq \emptyset$
      Let $e_{j^*}$ be such that $\|F(x_k + \epsilon V_m e_{j^*})\| = \min_{e_j\in D} \|F(x_k + \epsilon V_m e_j)\|$.
      Set $v_{j^*} = V_m e_{j^*}$.
      Compute $\alpha_e = \text{argmin}_{\alpha\in\mathbb{R}}\ \hat{g}(\alpha e_{j^*})$:   $\alpha_e = -F^T(x_k)F'(x_k)v_{j^*}/\|F'(x_k)v_{j^*}\|^2$.
      If $\|F(x_k + \alpha_e v_{j^*})\| \le \|F(x_k + z)\|$,  set $z = \alpha_e v_{j^*}$.
7. Compute
```

$$\begin{aligned} \eta_1 = {} & \|F'(x_k)z + F(x_k)\|/\|F(x_k)\|, \\ \eta_2 = {} & \|F'(x_k)\bar{s}_k + F(x_k)\|/\|F(x_k)\|. \end{aligned}$$

```
   Set $\eta = \eta_2$.
8. Repeat
      Choose $\theta \in [\theta_l, \theta_u]$. Set $\eta = 1 - \theta(1 - \eta)$.
      If $\eta < \eta_1$
            Let $\tau = (\frac{\eta^2-\eta_2^2}{\eta_1^2-\eta_2^2})^{1/2}$, $\hat{z} = \tau z + (1 - \tau)\bar{s}_k$.
      Else
            Let $\tau = (\frac{\eta^2-\eta_1^2}{1-\eta_1^2})^{1/2}$, $\hat{z} = (1 - \tau)z$.
   Until $\|F(x_k + \hat{z})\| \leq (1 - t(1 - \eta))\|F(x_k)\|$.
9. Set $\Delta_k = \hat{z}$, $\eta_k = \eta$.
```

In step 8 of the above algorithm, the following decrease condition in $\|F\|$ is imposed:

$$\|F(x_k + \sigma_k(\eta))\| \leq (1 - t(1 - \eta))\|F(x_k)\|, \tag{3.7}$$

where $t \in (0, 1)$ is a given constant. The backtracking is performed along the piecewise linear curve $\sigma_k(\eta)$ connecting the point zero, the point $z = \sigma_k(\eta_1) \in K_m$, and the inexact Newton step $\bar{s}_k = \sigma_k(\eta_2) \in K_m$. This curve has the form

$$\sigma_k(\eta) = \begin{cases} \tau(\eta)z + (1 - \tau(\eta))\bar{s}_k, & \tau(\eta) = (\frac{\eta^2-\eta_2^2}{\eta_1^2-\eta_2^2})^{1/2} \quad \text{if } \eta_2 \leq \eta < \eta_1, \\ (1 - \tau(\eta))z, & \tau(\eta) = (\frac{\eta^2-\eta_1^2}{1-\eta_1^2})^{1/2} \quad \text{if } \eta_1 \leq \eta \leq 1. \end{cases} \tag{3.8}$$

Some relevant features characterize z and $\bar{s}_k$. First, $\bar{s}_k$ minimizes $\|F'(x_k)s + F(x_k)\|$ in K_m due to the properties of GMRES and z minimizes $\|F'(x_k)s + F(x_k)\|$ along an appropriately chosen direction $V_m\bar{u} \in K_m$, i.e.,

$$z = \bar{\alpha}V_m\bar{u}, \qquad \bar{\alpha} = \text{argmin } \hat{g}(\alpha\bar{u}). \tag{3.9}$$

Second, the vector $\bar{u}$ is selected from d_m and the coordinate vectors e_j in $\mathbb{R}^m$ as the most promising descent direction for $\hat{g}$ at $y = 0$. This is done taking into account the value of $\|F\|$ according to the following strategy. From (3.4), the set D defined in step 5 of the ECB-strategy contains the coordinate vectors $e_j \in \mathbb{R}^m$ that are descent directions for $\hat{g}$ at $y = 0$. If $D = \emptyset$, we set $\bar{u} = d_m$. Otherwise, we consider the values $\|F(x_k + \epsilon V_m e_j)\|$, $e_j \in D$, and choose the vector e_{j^*} that gives the minimum value. Then, we form $\alpha_d V_m d_m$ and $\alpha_e V_m e_{j^*}$, i.e., the minimizers of $\hat{g}$ along d_m and e_{j^*}, respectively. Finally, $\bar{u}$ is chosen as either d_m or e_{j^*} depending on whether $\|F(x_k + \alpha_d V_m d_m)\|$ or $\|F(x_k + \alpha_e V_m e_{j^*})\|$ is smaller. Namely, $\bar{u} = d_m$ if $\|F(x_k + \alpha_d V_m d_m)\| < \|F(x_k + \alpha_e V_m e_{j^*})\|$, and $\bar{u} = e_{j^*}$ otherwise.

We remark that the path $\sigma_k(\eta)$ is not well defined if $\eta_1 = 1$. This occurrence means that there exist no descent directions for $\hat{g}$ at $y = 0$ in K_m and this case was previously excluded in the presentation of the ECB-strategy. Moreover, the path $\sigma_k(\eta)$ is not well defined if $\eta_1 = \eta_2$. This means that $\bar{s}_k$ is parallel to z. In this case, $\sigma_k(\eta)$ becomes a straight line and the ECB-strategy reduces to a backtracking technique along $\bar{s}_k$. However, in order to avoid the situation where $\sigma_k(\eta)$ becomes a straight line, a modification of the ECB-strategy can be considered. If $z = \alpha_e V_m e_{j^*}$, z can be set equal to $\alpha_d V_m d_m$; on the contrary, if $z = \alpha_d V_m d_m$ and $D \neq \emptyset$, one can take $z = \alpha_e V_m e_{j^*}$. Summarizing, the ECB-strategy cannot be used as an alternative strategy to the backtracking along $\bar{s}_k$ only either if $\bar{s}_k$ is parallel to both $V_m d_m$ and $V_m e_{j^*}$ or if $\bar{s}_k$ is parallel to $V_m d_m$ and the set D is empty. Also, it should be mentioned that in the context of globally convergent Newton-GMRES methods, a relevant trust-region model restricted to K_m was proposed in [3]. It employs a dogleg path based on

$\bar{s}_k$ and $\alpha_d V_m d_m$. We note that when $\bar{s}_k$ is parallel to $V_m d_m$, this dogleg path reduces to a straight line, while the outlined modification of the ECB-strategy prevents this degeneration of $\sigma_k(\eta)$ if $D \neq 0$ and $\bar{s}_k$ is not parallel to $V_m e_{j^*}$. In view of all above considerations, we will exclude the case $\eta_1 = \eta_2$ for the remainder of the paper.

The theorem below shows that along the curve $\sigma_k(\eta)$ we satisfy (3.5). Consequently, moving from $x_k + \bar{s}_k$ to x_k along $\sigma_k(\eta)$, the relative residual norm $\eta = \|F(x_k) + F'(x_k)\sigma_k(\eta)\|/\|F(x_k)\|$ is monotone increasing.

THEOREM 3.1. *If $m > 1$ and $\bar{s}_k$ provided by Algorithm GNE is such that $\bar{s}_k \neq 0$, along the piecewise curve $\sigma_k(\eta)$, the equality condition* (3.5) *is satisfied.*

Proof. By construction, the vectors $\bar{s}_k$ and z satisfy

$$\bar{s}_k = \underset{s \in K_m}{\operatorname{argmin}} \|F'(x_k)s + F(x_k)\|, \qquad z = \underset{s \in \bar{K}}{\operatorname{argmin}} \|F'(x_k)s + F(x_k)\|,$$

where $\bar{K} = span\{V_m d_m\}$ or $\bar{K} = span\{v_{j^*}\}$. Hence, if $m > 1$ and $\bar{s}_k \neq 0$, $\bar{K} \subset K_m$ and $\eta_1 > \eta_2$. Using [8, Lemma 8.1] the result follows. □

Remark 3.1. The proposed NGECB method is consistent with a matrix-free implementation. To see why, we turn our attention to the given general paradigm and consider the implementation of Algorithm GNE in step 2 and the implementation of the ECB-strategy in step 5. Step 2 can be implemented by using (1.4) to approximate the products $F'(x_k)v_i$, $i = 1, \dots, m$. Moreover, considering steps 6 and 7 of the ECB-strategy, α_e and η_1 can be computed without explicitly forming the Jacobian $F'(x_k)$. In particular, if $z = \alpha_e v_{j^*}$, the vector $F'(x_k)v_{j^*}$ has already been computed in Algorithm GNE. If $z = \alpha_d V_m d_m$, from (2.3) we have

$$\alpha_d F'(x_k) V_m d_m + F(x_k) = \alpha_d V_{m+1} \bar{H}_m d_m + F(x_k),$$

and then we have to perform only two matrix-vector products to compute η_1. Further, the scalar ϵ used in the ECB-strategy can be chosen equal to the constant ϵ previously used to form the products $F'(x_k)v_i$, $i = 1, 2, \dots, m$, by means of (1.4). This way, vectors of the form $F(x_k + \epsilon v_j)$ are available and the NGECB method can be advantageously implemented at a low computational cost.

3.2. Convergence analysis. To analyze the NGECB method, we observe that it belongs to the following framework:

```
Given $x_0$ and $t \in (0, 1)$.
For $k = 0$ until "convergence" do:
    Find a scalar $\eta_k \in [0, 1)$ and a vector $\Delta_k = \Delta_k(\eta_k)$ that satisfy
    (3.10)    $\|F(x_k) + F'(x_k)\Delta_k(\eta_k)\| \leq \eta_k \|F(x_k)\|$
    and
    (3.11)    $\|F(x_k + \Delta_k(\eta_k))\| \leq (1 - t(1 - \eta_k))\|F(x_k)\|$.
    Set $x_{k+1} = x_k + \Delta_k(\eta_k)$.
```

The value of Δ_k may be $\bar{s}_k$ or, alternatively, computed either in the backtracking loop BL, $\Delta_k(\eta) = s_k$ with $s_k = (1 - \eta)\bar{s}_k/(1 - \bar{\eta}_k)$, $\bar{\eta}_k \leq \eta \leq 1$, or by the ECB-strategy, $\Delta_k(\eta) = \sigma(\eta)$, $\eta_2 \leq \eta \leq 1$.

Next, we show that if the NGECB method does not break down, and if there exists a limit point x^* of $\{x_k\}$ such that $F'(x^*)$ is invertible, then the iterates are guaranteed to remain near x^* and

- $F(x^*) = 0$ and $x_k \to x^*$;
- for k sufficiently large, x_{k+1} has the form $x_{k+1} = x_k + \bar{s}_k$. Hence, the ultimate rate of convergence depends on the choices of the $\bar{\eta}_k$'s as shown in the local convergence theory of [6].

On the other hand, the NGECB method can fail in one of the following cases:

- $\|x_k\| \to \infty$, i.e., $\{x_k\}$ has no limit points;
- $\{x_k\}$ has one or more limit points, and F' is singular at each of them;

and it breaks down if

- $F'(x_k)$ is singular for some k;
- the vector $\bar{s}_k$ computed by Algorithm GNE is such that $\bar{s}_k = s_k^0 = 0$.

In our analysis we will refer to the theoretical results of [8] for processes that fit into the above framework, i.e., that generate a sequence $\{x_k\}$ where $\{\Delta_k\}$ and $\{\eta_k\}$ satisfy (3.10) and (3.11) for all k. For the sake of completeness, we recall these facts.

THEOREM 3.2 (see [8, Theorem 3.4]). *If* $\sum_{k=0}^{\infty}(1-\eta_k)$ *is divergent, then* $F(x_k) \to 0$.

THEOREM 3.3 (see [8, Theorem 3.5]). *Let* x^* *be a limit point of* $\{x_k\}$ *for which there exists a* Γ *independent of* k *such that*

$$\|\Delta_k(\eta_k)\| \leq \Gamma(1-\eta_k)\|F(x_k)\|, \tag{3.12}$$

whenever x_k *is sufficiently near* x^* *and* k *is sufficiently large. Then,* $x_k \to x^*$.

Next, we give sufficient conditions for the NGECB method to not break down in the backtracking loop in step 8 of the ECB-strategy. Then, we give a global convergence result.

LEMMA 3.1. *Assume that* $\Delta_k(\eta) = \sigma_k(\eta)$, $F(x_k) \neq 0$, *and there exist* $\eta_\sigma \geq \eta_2$ *and a constant* Γ *such that*

$$\|\Delta_k(\eta)\| \leq \Gamma(1-\eta)\|F(x_k)\|, \quad \eta_\sigma \leq \eta \leq 1. \tag{3.13}$$

Let $\delta > 0$ *be sufficiently small so that*

$$\|F(x) - F(x_k) - F'(x_k)(x-x_k)\| \leq \frac{1-t}{\Gamma}\|x-x_k\| \tag{3.14}$$

holds whenever $x \in N_\delta(x_k)$. *Then, in step* 8 *of the ECB-strategy, the backtracking loop terminates with*

$$1-\eta_k \geq \min\left\{1-\eta_\sigma, \frac{\theta_l \delta}{\Gamma\|F(x_k)\|}\right\}.$$

Proof. The proof is a straightforward application of [8, Lemma 5.1]. □

Remark 3.2. Note that [19, Lemma 3.2.10] ensures the existence of δ such that (3.14) holds whenever $x \in N_\delta(x_k)$.

The next lemma gives conditions under which (3.13) is satisfied.

LEMMA 3.2. *Assume that* $\Delta_k(\eta) = \sigma_k(\eta)$, $\bar{s}_k \neq 0$, $F(x_k) \neq 0$, *and* $F'(x_k)$ *is invertible. Then, there exist* η_σ *and* Γ *such that* (3.13) *holds.*

Proof. If

$$\gamma_k = \inf_{\eta_2 \leq \eta < 1} \frac{|F(x_k)^T(F'(x_k)\sigma_k(\eta))|}{\|F(x_k)\|\|F'(x_k)\sigma_k(\eta)\|}$$

is positive, [8, Lemma 7.2] gives (3.13) with

$$\Gamma = \frac{2\|F'(x_k)^{-1}\|}{\gamma_k} \quad \text{and} \quad \eta_\sigma = \max\{\eta_2, (1-\gamma_k^2)^{1/2}\}.$$

We now show that $\gamma_k > 0$. If $\eta_1 \le \eta < 1$, from (3.8) we have $\sigma_k(\eta) = (1-\tau(\eta))z$, where $z = \alpha_e v_{j^*}$ or $z = \alpha_d V_m d_m$. If $z = \alpha_e v_{j^*}$, the assumption $h_{1,j^*} \ge h_{min} > 0$ in the ECB-strategy and (3.2), give

$$\begin{aligned}\gamma_k &= \frac{|F(x_k)^T(F'(x_k)V_m e_{j^*})|}{\beta_k\|F'(x_k)V_m e_{j^*}\|}\\ &= \frac{\beta_k h_{1,j^*}}{\beta_k\|F'(x_k)V_m e_{j^*}\|} \ge \frac{h_{min}}{\|F'(x_k)V_m e_{j^*}\|} > 0. \end{aligned} \tag{3.15}$$

If $z = \alpha_d V_m d_m$, from (3.2) and (3.6), it follows $F(x_k)^T F'(x_k)V_m = -\beta_k d_m^T$, and thus

$$\begin{aligned}\gamma_k &= \frac{|F(x_k)^T(F'(x_k)V_m d_m)|}{\beta_k\|F'(x_k)V_m d_m\|}\\ &= \frac{\|d_m\|^2}{\|F'(x_k)V_m d_m\|} > 0,\end{aligned} \tag{3.16}$$

where the last inequality holds since $\bar{s}_k \ne 0$ (see (3.3)).

Finally, if $\eta_2 \le \eta < \eta_1$, the equality condition (3.5) gives

$$\begin{aligned}\gamma_k &= \inf_{\eta_2\le\eta<\eta_1} \frac{|-F(x_k)^T F(x_k) + F(x_k)^T(F'(x_k)\sigma_k(\eta) + F(x_k))|}{\beta_k\|F'(x_k)\sigma_k(\eta)\|}\\ &\ge \inf_{\eta_2\le\eta<\eta_1} \frac{\beta_k^2(1-\eta)}{\beta_k(\beta_k + \|F'(x_k)\sigma_k(\eta) + F(x_k)\|)}\\ &= \inf_{\eta_2\le\eta<\eta_1} \frac{1-\eta}{1+\eta} \ge \frac{1-\eta_1}{1+\eta_1} > 0. \qquad \square\end{aligned} \tag{3.17}$$

Lemmas 3.1 and 3.2 yield the result below.

COROLLARY 3.1. *Assume that* $\Delta_k(\eta) = \sigma_k(\eta)$, $\bar{s}_k \ne 0$, $F(x_k) \ne 0$, *and* $F'(x_k)$ *is invertible. Then, the backtracking loop in step* 8 *of the ECB-strategy terminates with*

$$1 - \eta_k \ge \min\left\{1-\eta_2, 1-(1-\gamma_k^2)^{1/2}, \frac{\theta_l\delta}{\Gamma\|F(x_k)\|}\right\} \tag{3.18}$$

for any $\delta > 0$ *sufficiently small so that* (3.14) *holds whenever* $x \in N_\delta(x_k)$.

To prove the global convergence result we need an intermediate result.

THEOREM 3.4. *Assume that* $\bar{s}_k$ *satisfies*

$$\|F'(x_k)\bar{s}_k + F(x_k)\| \le \bar{\eta}_k\|F(x_k)\|, \qquad \bar{\eta}_k \le \eta_{max}, \tag{3.19}$$

for k *sufficiently large, and* $\Delta_k(\eta) = \sigma_k(\eta)$. *If* x^* *is a limit point of* $\{x_k\}$ *such that* $F'(x^*)$ *is invertible, then there exists* Γ_1 *independent of* k *for which*

$$\|\Delta_k(\eta_k)\| \le \Gamma_1(1-\eta_k)\|F(x_k)\| \tag{3.20}$$

whenever x_k *is sufficiently close to* x^* *and* k *is sufficiently large.*

Proof. Let $\delta > 0$ be sufficiently small so that $F'(x)^{-1}$ exists and $\|F'(x)^{-1}\| \le 2\|F'(x^*)^{-1}\|$ whenever $x \in N_\delta(x^*)$. Set $K = \|F'(x^*)^{-1}\|$, $K_1 = \sup_{x\in N_\delta(x^*)}\|F'(x)\|$, and suppose that $x_k \in N_\delta(x^*)$ and k is sufficiently large that (3.19) holds.

Lemma 3.2 gives (3.13) with $\Gamma = 2\|F'(x_k)^{-1}\|/\gamma_k$, dependent of k. To prove the theorem we now show that there exists $\gamma > 0$ such that $\gamma_k > \gamma$ whenever $x_k \in N_\delta(x^*)$.

First, consider the case $\eta_1 \le \eta \le 1$, i.e., $\sigma_k(\eta) = (1-\tau(\eta))z$. If $z = \alpha_e v_{j^*}$, from (3.15) we have $\gamma_k > h_{min}/K_1$. If $z = \alpha_d V_m d_m$, (3.16) yields

$$\gamma_k \ge \frac{\|d_m\|^2}{\|F'(x_k)\|\|d_m\|} = \frac{\|V_m^T F'(x_k)^T F(x_k)\|}{\beta_k \|F'(x_k)\|} \ge \frac{1}{k_2(F'(x_k))} \frac{\|V_m^T F'(x_k)^T F(x_k)\|}{\|F'(x_k)^T F(x_k)\|},$$

where $k_2(F'(x_k)) = \|F'(x_k)\|\|F'(x_k)^{-1}\|$. Recalling that in K_m there is $\bar{s}_k$ satisfying (3.19), we also have

$$\frac{\|V_m^T F'(x_k)^T F(x_k)\|}{\|F'(x_k)^T F(x_k)\|} \ge \frac{1}{k_2(F'(x_k))} \frac{1-\bar{\eta}_k}{1+\bar{\eta}_k}$$

(see [4, Corollary 3.5]), and we get

$$\gamma_k > \frac{1}{4K^2K_1^2} \frac{1-\eta_{max}}{1+\eta_{max}}.$$

In what follows, we let γ_1 be the constant

$$\gamma_1 = \min\left\{ \frac{h_{min}}{K_1}, \frac{1-\eta_{max}}{4K^2K_1^2(1+\eta_{max})} \right\}. \tag{3.21}$$

Next, we turn our attention to the case $\eta_2 \le \eta < \eta_1$. Due to (3.17), we need an upper bound, independent of k, on η_1. Using the definition of η_1 and the definition (3.9) of z we have

$$\eta_1 = \frac{\|F'(x_k)z + F(x_k)\|}{\|F(x_k)\|} = \frac{\min_\alpha \|F'(x_k)\alpha V_m \bar{u} + F(x_k)\|}{\|F(x_k)\|} = \left(1 - \frac{(F(x_k)^T F'(x_k) V_m \bar{u})^2}{\beta_k^2 \|F'(x_k) V_m \bar{u}\|^2}\right)^{1/2} = \left(1 - \inf_{\eta_1 \le \eta < 1} \frac{(F(x_k)^T F'(x_k)\sigma_k(\eta))^2}{\beta_k^2 \|F'(x_k)\sigma_k(\eta)\|^2}\right)^{1/2},$$

and from (3.17) and (3.21) we get

$$\gamma_k \ge \frac{1-\eta_1}{1+\eta_1} \ge \frac{1-\sqrt{1-\gamma_1^2}}{1+\sqrt{1-\gamma_1^2}}.$$

Hence, by setting $\gamma = \min\{\gamma_1, \frac{1-\sqrt{1-\gamma_1^2}}{1+\sqrt{1-\gamma_1^2}}\}$ and $\Gamma_1 = 4K/\gamma$, we obtain

$$\|\sigma_k(\eta)\| \le \Gamma_1(1-\eta)\|F(x_k)\|, \quad \min\{\eta_2, 1-(1-\gamma^2)^{1/2}\} \le \eta \le 1,$$

and the proof is completed. □

We are ready to state the global convergence result.

Theorem 3.5. *Assume that the NGECB method does not break down, and* (3.19) *holds for k sufficiently large. If x^* is a limit point such that $F'(x^*)$ is invertible, then $x_k \to x^*$, and $F(x^*) = 0$. Furthermore, for sufficiently large k, we have $\eta_k = \bar{\eta}_k$.*

Proof. Let $\delta > 0$ be sufficiently small so that (3.20) holds whenever $x \in N_\delta(x^*)$. Set $K_2 = \sup_{x \in N_\delta(x^*)} \|F(x)\|$. Assume that $x_k \in N_\delta(x^*)$ and k is sufficiently large that (3.19) holds.

If $\Delta_k(\eta) = s_k$, from [8, Theorem 6.1] we know that there exists a constant, say, Γ_2, independent of k such that (3.12) holds. Then, we conclude that whenever x_k is sufficiently close to x^*, the NGECB method generates a Δ_k that satisfies (3.12) with $\Gamma = \min\{\Gamma_1, \Gamma_2\}$ independent of k. Hence $x_k \to x^*$ because of Theorem 3.3.

If s_k is computed in the backtracking loop BL, the backtracking terminates with η_k such that

$$1 - \eta_k \geq \min\left\{1 - \bar{\eta}_k, \frac{\delta\theta_l}{\Gamma\|F(x_k)\|}\right\} \tag{3.22}$$

(see [8, Lemma 5.1]). Hence, in this case we have

$$1 - \eta_k \geq \min\left\{1 - \eta_{max}, \frac{\delta\theta_l}{\Gamma K_2}\right\}.$$

Further, if the ECB-strategy is used, (3.18) yields

$$1 - \eta_k \geq \min\left\{1 - \eta_{max}, 1 - (1 - \gamma^2)^{1/2}, \frac{\theta_l\delta}{\Gamma K_2}\right\}.$$

But, since $x_k \to x^*$, we have $x_k \in N_\delta(x^*)$ for k sufficiently large. Therefore, the series $\sum_{k=0}^{\infty}(1 - \eta_k)$ is divergent and Theorem 3.2 yields $F(x^*) = 0$. Finally, since $\|F(x_k)\| \to 0$, from (3.22) it follows $\eta_k = \bar{\eta}_k$ for k sufficiently large. □

3.3. Restarting and enlarged subspace ECB-strategy. In typical implementations of GMRES, a maximum value m_{max} of m is dictated by storage requirements. If a restarted GMRES is applied, GMRES is used iteratively and restarted every m_{max} iterations. At each restart the initial guess is set equal to the last computed GMRES iterate. The NGECB method studied in section 3.1 does not cover this situation, because, in general, after a restart the new initial guess s_k^0 is nonzero. We now show how to define the NGECB method in this case.

When $s_k^0 \neq 0$, the major impact on our hybrid method is that the vector $\bar{s}_k$ provided by Algorithm GNE belongs to $s_k^0 + K_m$ and $s_k^0 + K_m \neq K_m$. Then, the step in $s_k^0 + K_m$ from x_k has the form $s_k^0 + V_m y$, $y \in \mathbb{R}^m$, and $g(y)$ in (3.1) is now given by $g(y) = M(x_k + s_k^0 + V_m y)$. In general, $g(0) \neq M(x_k)$, and we cannot search for a decrease of M moving from x_k in K_m. Hence, following the approach of [3] we let the contribution of the step s_k^0 be variable, and we search for a descent direction in the subspace $S = span\{v_1, v_2, \ldots, v_m, s_k^0\}$.

Letting $W = [w_1, w_2, \ldots, w_{m+1}] \in \mathbb{R}^{n\times(m+1)}$ be the orthonormal basis of S such that $w_i = v_i$ for $i = 1, 2, \ldots, m$, we replace the function g given in (3.1) with the following function:

$$g(\tilde{y}) = M(x_k + W\tilde{y}),$$

where $g : \mathbb{R}^{m+1} \to \mathbb{R}$.

The model $\hat{g}(\tilde{y})$ for $M(x_k + W\tilde{y})$ becomes

$$\hat{g}(\tilde{y}) = \|F'(x_k)W\tilde{y} + F(x_k)\|, \tag{3.23}$$

where $F'(x_k)W = [V_{m+1}\bar{H}_m, F'(x_k)w_{m+1}]$, and the steepest descent direction for $\hat{g}$ at $\tilde{y} = 0$ is given by

$$\tilde{d}_m = -\frac{(F'(x_k)W)^T F(x_k)}{\|F(x_k)\|} = \frac{1}{\|F(x_k)\|}\begin{pmatrix} -\bar{H}_m^T V_{m+1}^T F(x_k) \\ -w_{m+1}^T F'(x_k)^T F(x_k) \end{pmatrix}. \tag{3.24}$$

Clearly, there will be descent directions for $\hat{g}$ at $\tilde{y} = 0$ in S if $F'(x_k)W$ is not singular. This is ensured if $s_k^0 \notin K_m$.

In order to define the equality curve σ_k, we need the minimizer $\hat{s}_k$ of $\|F'(x_k)s + F(x_k)\|$ in S. This vector can be easily computed in a matrix-free manner using the information provided by GMRES, as shown in [3]. Hence, we let σ_k be of the form (3.8), where $\bar{s}_k$ is replaced by $\hat{s}_k$ and z is such that

$$z = \bar{\alpha} W \tilde{u}, \qquad \bar{\alpha} = \text{argmin } \hat{g}(\alpha\tilde{u}). \tag{3.25}$$

The definition of $\tilde{u}$ can be restated along the same lines as in the ECB-strategy. In particular, $\tilde{u}$ is selected among $\tilde{d}_m$ and the vectors $e_j \in \mathbb{R}^{m+1}$ that are descent directions for $\hat{g}$ at $\tilde{y} = 0$. To establish whether the vectors e_j are descent directions, we use the relation

$$\nabla\hat{g}(y)|_{y=0}^T e_j = -(\tilde{d}_m)_j, \quad j = 1, \ldots, m+1.$$

Going on as in the proof of Theorem 3.1, it follows that the curve σ_k constructed by using $\hat{s}_k$ and the vector z defined in (3.25) satisfies (3.5). Therefore, the theoretical results of section 3.2 hold.

Based upon the preceding discussion, the ECB-strategy can be applied by replacing $\bar{s}_k$ with $\hat{s}_k$ and rephrasing steps 2–6 as follows.

```
2. Compute $\tilde{d}_m$ using (3.24).
3. Compute $\alpha_d = \text{argmin}_{\alpha\in\mathbb{R}}\ \hat{g}(\alpha\tilde{d}_m)$: $\alpha_d = \|F(x_k)\|\ \|\tilde{d}_m\|^2/\|F'(x_k)W\tilde{d}_m\|^2$.
4. Set $z = \alpha_d W\tilde{d}_m$.
5. Form $J = \{1 \le j \le m+1 \text{ s.t. } (\tilde{d}_m)_j \ge h_{min}\}$,  $D = \{e_j \text{ s.t. } j \in J\}$.
6. If $D \ne \emptyset$
     Let $e_{j^*}$ be such that $\|F(x_k + \epsilon W e_{j^*})\| = \min_{e_j\in D} \|F(x_k + \epsilon W e_j)\|$.
     Set $w_{j^*} = We_{j^*}$.
     Compute $\alpha_e = \text{argmin}_{\alpha\in\mathbb{R}}\ \hat{g}(\alpha e_{j^*})$: $\alpha_e = -F^T(x_k)F'(x_k)w_{j^*}/\|F'(x_k)w_{j^*}\|^2$.
     If $\|F(x_k + \alpha_e w_{j^*})\| \le \|F(x_k + z)\|$  set $z = \alpha_e w_{j^*}$.
```

We point out that the above computations are implementable using only matrix-vector products. In particular, the evaluation of $\tilde{d}_m$ does not need the Jacobian explicitly since the product $F'(x_k)w_{m+1}$ can be approximated by finite differences. Moreover, if $z = \alpha_e w_{j^*}$, $1 \le j^* \le m$, the matrix-vector product $F'(x_k)w_{j^*}$ has already been computed by Algorithm GNE. Similarly, the evaluation of α_d does not need the Jacobian explicitly.

Regarding the function evaluations, if in step 1 of the ECB-strategy we take the scalar ϵ equal to the one used in (1.4), the vectors $F(x_k + \epsilon We_j)$ are available.

3.4. Preconditioning.

In order to increase the convergence rate of Krylov methods, preconditioning techniques are commonly used. Given the Newton equation (1.3), preconditioning leads to the equivalent system

$$P_l F'(x_k) P_r P_r^{-1} s = -P_l F(x_k), \qquad k \ge 0.$$

The matrices P_l and P_r are chosen so that the preconditioned system is easier to solve than the original one. The system is preconditioned on the right if P_l is equal to the identity matrix I, on the left if $P_r = I$.

In our case, we need to restrict ourselves to right preconditioning in order to use the NGECB method. This is because if $P_l \neq I$, the residual of the linear system is $\|P_l F'(x_k)s + P_l F(x_k)\|$. Hence, $\hat{g}$ should be $\|P_l F'(x_k)V_m y + P_l F(x_k)\|$. In other words, $\hat{g}$ would be a model for $\|P_l F(x_k + V_m y)\|$ instead of $\|F(x_k + V_m y)\|$. On the contrary, right preconditioning does not affect the residual associated to the linear system. Thus, we consider the linear system

$$F'(x_k)P_r p = -F(x_k), \qquad p = P_r^{-1}s.$$

Without loss of generality, we concentrate on the case where a null initial guess for GMRES, $p_k^0 = 0$, is chosen and restart is not used. In this case, the Krylov space generated by GMRES has the form

$$K_m^p = \{r_k^0, (F'(x_k)P_r)r_k^0, (F'(x_k)P_r)^2 r_k^0, \ldots, (F'(x_k)P_r)^{m-1} r_k^0\},$$

where $r_k^0 = -F(x_k)$ and (2.3) becomes $F'(x_k)P_r V_m = V_{m+1}\bar{H}_m$. Clearly, the vector $\bar{p}_k$ in K_m^p such that $\|F'(x_k)P_r\bar{p}_k + F(x_k)\| \leq \bar{\eta}_k\|F(x_k)\|$ gives rise to the approximate solution $\bar{s}_k = P_r\bar{p}_k$ of (1.3). Consequently, $\bar{s}_k \in P_r K_m^p$ and this leads to the low-dimensional globalization strategy in the subspace $S = P_r K_m^p$.

Following the lines of the unpreconditioned case, since each vector $s \in P_r K_m^p$ is such that $s = P_r V_m y$, $y \in \mathbb{R}^m$, the function $g(y)$ in (3.1) becomes

$$g(y) = M(x_k + P_r V_m y).$$

Noting that for each $s \in P_r K_m^p$, the relation

$$\nabla M(x_k)^T s = \nabla g(y)|_{y=0}^T y = -e_1^T \bar{H}_m y$$

holds, then the directional derivatives have the same form of the unpreconditioned case. Finally, if we let

$$\hat{g}(y) = \|F'(x_k)P_r V_m y + F(x_k)\|,$$

the application of the ECB-strategy is straightforward.

4. Numerical tests. Our numerical experiments show that the combination of two backtracking strategies yields a significant benefit in the likelihood of convergence of an inexact method. To support this claim, we compared performance of NGB and NGECB methods on several problems.

The tests were conducted on an IBM Risc 3CT workstation using MATLAB 5.2. The machine precision is $\epsilon_m \simeq 2.10^{-16}$. In our implementation, products of the form $F'(x_k)v$ that are required by Algorithm GNE were approximated by (1.4) with $\epsilon = \sqrt{\epsilon_m}\|x_k\|/\|v\|$. Note that for $v = v_j$, $j = 1, \ldots, m$, ϵ reduces to $\sqrt{\epsilon_m}\|x_k\|$, i.e., it is independent of j. Also, in order to implement an efficient matrix-free procedure, this value of ϵ was used to construct the piecewise linear curve σ_k in step 6 of the ECB-strategy. Further, in the ECB-strategy, h_{min} was set equal to $10\epsilon_m$.

Concerning the forcing terms $\{\bar{\eta}_k\}$, we adopted one of the proposals of [9, p. 305]: we used "*Choice* 2 *safeguard*" with $\gamma = 0.9$, $\alpha = 2$, $\eta_{max} = 0.9$, $\eta_0 = \eta_{max}$.

Algorithm GNE was started with $s_k^0 = 0$. At most $m_{max} = 40$ GMRES-iterations were allowed, and restarting was not employed. If after $m = m_{max}$ steps

$\|r_k^m\| > \bar{\eta}_k \|F(x_k)\|$, NGB and NGECB algorithms continued with the vector provided in the final GMRES iteration. Namely, the inexact Newton step $\bar{s}_k$ was set equal to s_k^m and $\bar{\eta}_k = \|F'(x_k)\bar{s}_k + F(x_k)\|/\|F(x_k)\|$ was used. Right diagonal and ILU(1) preconditioners were allowed.

In the backtracking loop BL defined in step 3 of NGB and in the backtracking loop performed in step 8 of the ECB-strategy, the typical values $t = 10^{-4}$, $\theta_l = 0.1$, $\theta_u = 0.5$ were used.

In both algorithms, the iterations were halted when $\|F(x_k)\|/\sqrt{n} \leq 10^{-6}$. A failure was declared if one of the following situations occurred: the maximum of 300 nonlinear iterations was reached, or $\|F(x_{k-1})\| - \|F(x_k)\| \leq 10^{-6}\|F(x_k)\|$ was detected. This last occurrence is commonly indicated as a "stagnation" of the nonlinear iterates [10], [15], and it may indicate that the method did not manage to escape from a local minimizer of the merit function. Also, a failure of NGB was declared if 50 iterations of the backtracking loop BL failed to produce the sufficient decrease (2.5) in $\|F\|$. Analogously, if in step 8 of the ECB-strategy (3.7) was not verified within 50 backtracking steps, a failure of NGECB was declared.

In order to analyze the computational performance of NGECB, we conducted experiments with different choices of the parameter N_b. We recall that N_b is the maximum number of times the backtracking loop BL can be repeated in the NGECB algorithm, i.e., is the maximum number of backtracks allowed along $\bar{s}_k$ before switching to the ECB-strategy. For the sake of brevity, we will shorten NGECB with $N_b = p$ to NGECB(p). We emphasize that NGECB(p) reduces to NGB for all the problems that, at each iteration, require a number of backtracks less than or equal to p. Further, the choice $N_b = 0$ means that we turn off backtracking along the inexact Newton step. In other words, NGECB(0) uses a globalization strategy where only backtracking along the piecewise linear curve is performed.

In reporting the results of NGB and NGECB, the following values are given in the tables:

- `NI`, number of nonlinear iterations;
- `FE`, number of function evaluations;
- `BT`, number of backtracks;
- `SW`, number of switches to the ECB-strategy. The symbol "*" indicates a failure. We remark that, due to the backtracking strategies and the matrix-free implementation, the most significant part of the computational cost is revealed by `FE`.

To realistically assess performance of NGECB, numerical experiments were carried out on a large set of problems; see Table 4.1. For each problem, several starting guesses x_0 were used. In particular, letting x_s be the standard initial guess used in literature, *ones* be the vector in $\mathbb{R}^n$ with all components equal to 1, and *null* be the zero vector in $\mathbb{R}^n$, we used $x_0 = \pm x_s, \pm 2x_s, \pm 5x_s, ones, 2\,ones, 5\,ones, null$. However, for several problems we neglected those initial guesses x_0 for which we had $F(x_0) = 0$ or singular $F'(x_0)$. Further, the dimension n of each test was varied. Below we will discuss the results obtained with n given in Table 4.1.

We notice that our test problems display different features: problems with ill-conditioned Jacobian (P2, P14), problems with singular Jacobian at the solution (P4, P10, P11), and one parameter-dependent problem (P17).

Our experiments showed a wide array of convergence behaviors: often NGB needed few or no backtracks to enforce convergence, but at times it was unsuccessful. On a total of 143 tests, NGB failed 51 times. The most recurrent failure was stag-

Table 4.1
Nonlinear problems, dimension sizes n, and standard initial guesses x_s used in testing.

Pb #	Problem name	n	x_s
P1	Countercurrent reactor [16]	512	$0.1(1,2,3,4,5,4,3,2,\ldots,1,2,3,4,5,4,3,2)$
P2	Powell badly scaled [16]	4096	$(0,1,0,1,\ldots,0,1)$
P3	Trigonometric [16]	5000	$(0,1,0,1,\ldots,0,1)$
P4	Singular Broyden [16]	4096	$(-1,-1,-1,-1,\ldots,-1,-1)$
P5	Tridiagonal [16]	512	$(12,12,12,12,\ldots,12,12)$
P6	Five-diagonal [16]	256	$(-3,-3,-3,-3,\ldots,-3,-3)$
P7	Seven-diagonal [16]	512	$(-3,-3,-3,-3,\ldots,-3,-3)$
P8	Premultiplied diagonal ... [12, Pb. D7]	4098	$(50,0.5,-1,\ldots,50,0.5,-1)$
P9	Augmented Rosenbrock [12]	4096	$(-1.2,1,-1,20,\ldots,-1.2,1,-1,20)$
P10	Extended Powell singular [16]	4096	$(3,-1,0,1,\ldots,3,-1,0,1)$
P11	Extended Cragg and Levy [16]	4096	$(1,2,2,2,\ldots,1,2,2,2)$
P12	Broyden tridiagonal function [16]	4096	$(-1,-1,-1,-1,\ldots,-1,-1)$
P13	Broyden tridiagonal problem [16]	4096	$(-1,-1,-1,-1,\ldots,-1,-1)$
P14	Augmented Powell badly scaled [12]	4096	$(0,1,-4,\ldots,0,1,-4)$
P15	Modified Rosenbrock [12]	4096	$(-1.8,-1,-1.8,-1,\ldots,-1.8,-1)$
P16	Chemical equilibrium [17]	11000	$(3,3,3,3,\ldots,3,3)$
P17	Bratu ($\alpha=10,\ \lambda=1$) [3]	4096	$(0,0,0,0,\ldots,0,0)$

Table 4.2
NGECB(5) *performance for problems P*2 *and P*14.

	Problem P2				Problem P14			
x_0	NI	FE	BT	SW	NI	FE	BT	SW
x_s	135	1034	589	50	*	*	*	*
$2\,x_s$	133	1029	589	50	139	1166	589	48
$5\,x_s$	125	1009	591	49	123	1096	570	49
$-x_s$	*	*	*	*	*	*	*	*
$-2\,x_s$	*	*	*	*	*	*	*	*
$-5\,x_s$	*	*	*	*	*	*	*	*
ones	133	1040	599	50	136	1149	583	49
2 *ones*	134	1031	589	50	137	1147	581	47
5 *ones*	120	918	515	50	124	1098	570	49
null	*	*	*	*	*	*	*	*

nation, which occurred for 42 tests; for 10 of them, no backtracks were performed. We point out that none of these failures can be ascribed to the absence of restarts in GMRES. In fact, the stopping criterion (1.2) is satisfied within m_{max} GMRES iterations in most of the tests. For the remaining tests we verified that stagnation occurred even if a restarted version of GMRES is used to satisfy (1.2).

First, we fixed $N_b = 5$ for NGECB. We remark that there is nothing critical about this choice. However, on the basis of our numerical tests it seems to work well and serves to prove the effectiveness of our method.

For 98 tests, NGECB(5) reduced to NGB. So we now focus on the remaining 45 tests (problems P2, P4, P5, P6, P7, P8, P9, P11, P14, P15, P16): NGB and NGECB(5) solved 9 and 24 tests, respectively. Then, the number of failures dropped from 80% to 47%. In Table 4.2 we summarize results for problems P2 and P14. These are the only two problems on which NGB failed to converge for all initial guesses; the most recurrent failure was due to stagnation.

For the 21 tests that were not solved by NGECB(5), we reran NGECB varying N_b from 0 to 4. This yielded a significant benefit since we managed to solve 6 tests; see Table 4.3 for results with $N_b = 0,\ 3$. Note that, except for P14, all the tests were solved at a low computational cost.

TABLE 4.3
Tests solved by NGECB, $p < 5$, that were unsuccessful for NGB and NGECB(5).

Pb#	x_0	NGECB(3)				NGECB(0)			
		NI	FE	BT	SW	NI	FE	BT	SW
P6	$5\,x_s$	20	128	20	3	17	76	3	2
P7	$5\,x_s$	*	*	*	*	17	73	1	1
P8	$2\,ones$	12	59	16	2	*	*	*	*
P14	$-2\,x_s$	143	1117	516	58	*	*	*	*
P16	x_s	19	159	5	1	19	169	3	3
P16	$5\,x_s$	*	*	*	*	23	227	13	6

The computational experiments presented above illustrate that on a total of 143 tests NGB converged 64% of the time, NGECB 85% of the time. Since for 10 failures no backtracks were performed, and hence cannot be recovered by NGECB, our results indicate that our hybrid strategy enhances the robustness of NGB. Further, it is worth noting that only for problems P7 and P16 with $x_0 = 5\,x_s$, the use of $N_b = 0$ was crucial to obtain convergence in cases where NGB failed. This underscores that the improvement in the global convergence properties of NGB is due to the combination of the linesearch strategies.

We stress the apparent similarity of the ECB-strategy and the dogleg strategy proposed by Brown and Saad in [3] and already mentioned in section 3.1. They introduce the quadratic model $\|F'(x_k)V_m y + F(x_k)\|^2$ and minimize it along a dogleg path based on the inexact Newton step $\bar{s}_k$ and the steepest descent direction $V_m d_m$. To be more precise, this dogleg path has 3 nodes, $\bar{s}_k$, $\alpha_d V_m d_m$, and zero. On the contrary, as was seen in the detailed description of the ECB-strategy, the piecewise linear curve $\sigma_k(\eta)$ connects $\bar{s}_k$, the vector z chosen between $\alpha_d V_m d_m$ and $\alpha_e V_m e_{j^*}$, and zero. Therefore, $\sigma_k(\eta)$ differs from the dogleg curve of [3] when $z \neq \alpha_d V_m d_m$ is selected in step 6 of the ECB-strategy. Let us consider the frequency of taking $z \neq \alpha_d V_m d_m$ in NGECB. Restricting our attention to the 30 tests for which NGECB(5) differs from NGB and succeeded, we observed that NGECB selected $z \neq \alpha_d V_m d_m$ 47% of the time. To show a conclusive proof that backtracking along the curve $\sigma_k(\eta)$ of the ECB-strategy can be more robust than backtracking along the dogleg curve, we considered a modified version of NGECB(p) where only $z = \alpha_d V_m d_m$ can be selected. We will refer to this modified version as NGDOG(p) (Newton-GMRES dogleg).

Thus, we verified that for some tests, the choice $z \neq \alpha_d V_m d_m$ was crucial to obtaining convergence. For example, among the tests contained in Table 4.3, NGDOG(p), $0 \leq p \leq 5$, did not solve problems P6 and P7 with $x_0 = 5\,x_s$. Also, Table 4.4 shows NGECB(p) and NGDOG(p) applied to problem P8. For the reported initial guesses, NGB failed to converge. The symbols ND and NV denote the number of times where $z = \alpha_d V_m d_m$ and $z = \alpha_e V_m e_{j^*}$ were selected, respectively. One sees that the ECB-strategy outperformed the dogleg strategy when the standard initial guess is used, while NGECB reduced to NGDOG when $x_0 = ones$ and $x_0 = 2\,ones$.

Finally, more insight into the behavior of NGECB may be gleaned by comparing the overall computational effort of NGB and NGECB. Therefore, we extensively examined successful runs. At this regard, the results in Table 4.5 are typical: there, we summarize the results on problem P5 for the full set of starting guesses. Note that starting with $x_0 = 2\,x_s$ NGB failed, but NGECB succeeded. Otherwise, except for $x_0 = 5\,x_s$, all the methods were successful. Notice that they perform very similarly on tests requiring a few backtracks ($x_0 = -x_s$, $-5\,x_s$, *null*) as well as on tests requiring several backtracks ($x_0 = x_s$, $5\,ones$).

TABLE 4.4
Comparison between NGECB(p) and NGDOG(p) with $p = 0$, 3, 5 for problem P8.

x_0	p	NGECB(p)						NGDOG(p)					
		NI	FE	BT	SW	ND	NV	NI	FE	BT	SW	ND	NV
x_s	5	15	51	7	1	0	1	*	*	*	*	*	*
	3	15	49	5	1	0	1	*	*	*	*	*	*
	0	*	*	*	*	*	*	*	*	*	*	*	*
ones	5	16	100	51	3	3	0	16	100	51	3	3	0
	3	*	*	*	*	*	*	*	*	*	*	*	*
	0	*	*	*	*	*	*	*	*	*	*	*	*
2 *ones*	5	*	*	*	*	*	*	*	*	*	*	*	*
	3	12	59	16	2	2	0	12	59	16	2	2	0
	0	*	*	*	*	*	*	*	*	*	*	*	*

TABLE 4.5
Comparison between NGB, NGECB(p) with $p = 0$, 3, 5 for problem P5.

x_0	NGB			NGECB(5)				NGECB(3)				NGECB(0)			
	NI	FE	BT	NI	FE	BT	SW	NI	FE	BT	SW	NI	FE	BT	SW
x_s	63	547	159	58	493	136	2	64	649	215	30	70	617	132	53
$2\,x_s$	*	*	*	215	3061	1374	121	264	3631	1570	227	269	3001	898	251
$5\,x_s$	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
$-x_s$	15	74	4	15	74	4	0	18	86	5	1	18	83	2	1
$-2\,x_s$	17	70	0	17	70	0	0	17	70	0	0	17	70	0	0
$-5\,x_s$	19	85	2	19	85	2	0	19	85	2	0	21	90	3	2
2 *ones*	9	45	0	9	45	0	0	9	45	0	0	9	45	0	0
5 *ones*	25	152	21	25	152	21	0	30	193	36	2	36	236	44	22
null	10	58	3	10	58	3	0	10	58	3	0	8	54	1	1

One final aspect that should be noted is the comparison of NGECB with an "exact" version of NGB. Therefore, we ran NGB with a tight stopping criterion for GMRES, i.e.,

$$\|F'(x_k)\bar{s}_k + F(x_k)\| \le \max\{100\epsilon_m\|F(x_k)\|, 100\epsilon_m\}. \tag{4.1}$$

In our experience, NGECB proved to be more efficient than the "exact" NGB. This is due to the choice of forcing terms that gives desirable fast local convergence and also tends to minimize "oversolving." In fact, as noted in [9], these forcing terms avoid wasting effort in the initial stages of the nonlinear iterative method. Obviously, we obtained a less significant benefit when GMRES converged quickly. Figure 4.1 shows NGECB and "exact" NGB applied to problems P7 and P8. There the graphs plot the value of $\|F\|$ as a function of the number of F-evaluations required to reach that value of $\|F\|$. One sees that for problem P7, NGECB showed a significant improvement in the number of function evaluations over the "exact" NGB counterpart. On the contrary, NGECB performed slightly better on problem P8, where only a few number of GMRES iterations were necessary to satisfy (4.1).

Summarizing the overall computational experiments, we have tackled 143 commonly used tests. We have shown that the introduction of the ECB-strategy enhances global convergence for cases where NGB fails. Further, although the ECB-strategy may coincide with a backtracking strategy along the dogleg curve given in [3], we have shown that in practice NGECB benefits from the possibility of selecting a backtracking curve different from the dogleg one. Finally, regarding the computational effort, the numerical tests suggest that our approach and the classical backtracking Newton-GMRES method have very similar cost.

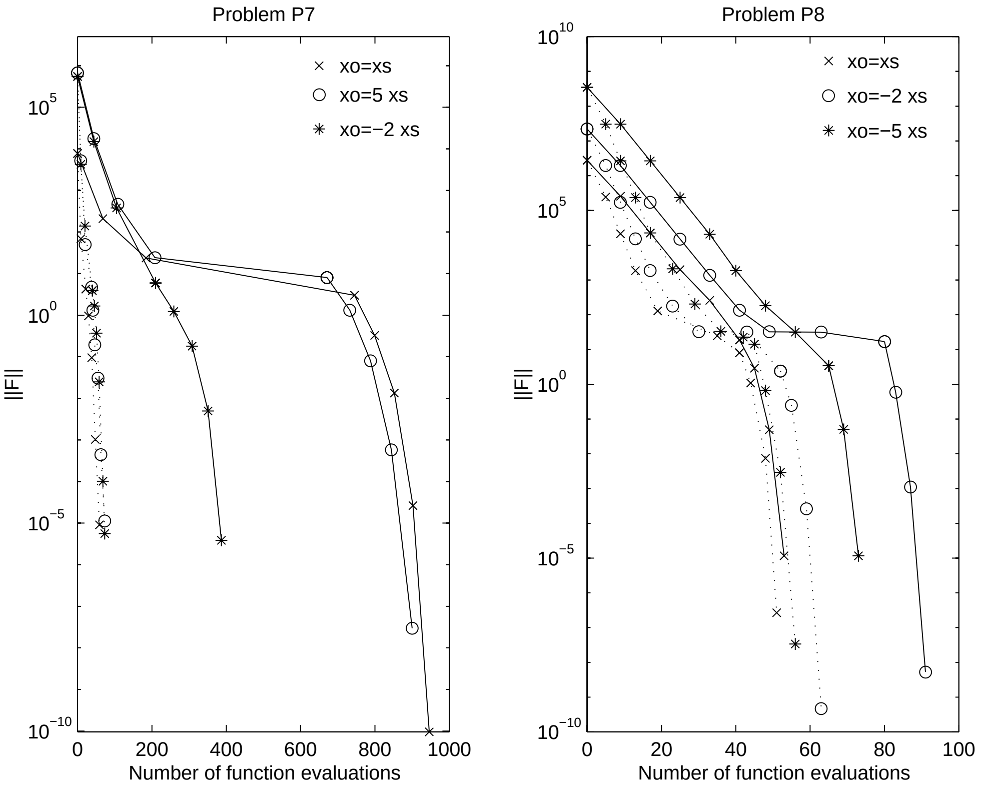


FIG. 4.1. *Comparison between NGECB(p), $p \leq 5$, and "exact" NGB. Dotted line: NGECB; solid line: "exact" NGB.*

Acknowledgments. The authors are indebted to Prof. Luca Dieci and Prof. Maria Macconi for many helpful discussions and suggestions related to this paper. Also, thanks are due to Prof. Margaret Wright for her careful reading of the manuscript and for several suggestions that improved the presentation of this work.

REFERENCES

[1] P. N. BROWN, *A local convergence theory for combined inexact-Newton/finite-difference projection methods*, SIAM J. Numer. Anal., 24 (1987), pp. 407–434.

[2] P. N. BROWN AND A. C. HINDMARSH, *Reduced storage matrix methods in stiff ODE systems*, Appl. Math. Comput., 31 (1989), pp. 40–91.

[3] P. N. BROWN AND Y. SAAD, *Hybrid Krylov methods for nonlinear systems of equations*, SIAM J. Sci. Statist. Comput., 11 (1990), pp. 450–481.

[4] P. N. BROWN AND Y. SAAD, *Convergence theory of nonlinear Newton–Krylov algorithms*, SIAM J. Optim., 4 (1994), pp. 297–330.

[5] M. A. BRANCH, T. F. COLEMAN, AND Y. LI, *A subspace, interior, and conjugate gradient method for large-scale bound-constrained minimization problem*, SIAM J. Sci. Comput., 21 (1999), pp. 1–23.

[6] R. S. DEMBO, S. C. EISENSTAT, AND T. STEIHAUG, *Inexact Newton methods*, SIAM J. Numer. Anal., 19 (1982), pp. 400–408.

[7] J. E. DENNIS AND R. B. SCHNABEL, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1983.

[8] S. C. EISENSTAT AND H. F. WALKER, *Globally convergent inexact Newton methods*, SIAM J. Optim., 4 (1994), pp. 393–422.

[9] S. C. Eisenstat and H. F. Walker, *Choosing the forcing term in an inexact Newton method*, SIAM J. Sci. Comput., 17 (1996), pp. 16–32.

[10] D. R. Fokkema, G. L. G. Sleijpen, and H. A. Van der Vorst, *Accelerated inexact Newton schemes for large systems of nonlinear equations*, SIAM J. Sci. Comput, 19 (1998), pp. 657–674.

[11] D. Feng and T. H. Pulliam, *Tensor-GMRES method for large systems of nonlinear equations*, SIAM J. Optim., 7 (1997), pp. 757–779.

[12] A. Friedlander, M. A. Gomes-Ruggiero, D. N. Kozakevich, J. M. Martinez, and S. A. Santos, *Solving nonlinear systems of equations by means of Quasi-Newton methods with nonmonotone strategy*, Optim. Methods Softw., 8 (1997), pp. 25–51.

[13] C. T. Kelley, *Iterative Methods for Linear and Nonlinear Equations*, Frontiers Appl. Math. 16, SIAM, Philadelphia, 1995.

[14] C. T. Kelley, *Iterative Methods for Optimization*, Frontiers Appl. Math. 18, SIAM, Philadelphia, 1999.

[15] I. E. Kaporin and O. Axelsson, *On a class of nonlinear equation solvers based on the residual norm reduction over a sequence of affine subspaces*, SIAM J. Sci. Comput., 16 (1995), pp. 228–249.

[16] L. Luksan, *Inexact trust region method for large sparse systems of nonlinear equations*, J. Optim. Theory Appl., 81 (1994), pp. 569–591.

[17] K. Meintjes and A. P. Morgan, *Chemical equilibrium systems as numerical tests problems*, ACM Trans. Math. Softw., 16 (1990), pp. 143–151.

[18] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer Ser. Oper. Res., Springer-Verlag, New York, 1999.

[19] J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equation in Several Variables*, Academic Press, New York, 1970.

[20] M. Pernice and H. F. Walker, *NITSOL: A Newton iterative solver for nonlinear systems*, SIAM J. Sci. Comput., 19 (1998), pp. 302–318.

[21] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

SIAM J. SCI. COMPUT.
Vol. 23, No. 3, pp. 961–975

THE LINEAR RATIONAL PSEUDOSPECTRAL METHOD WITH ITERATIVELY OPTIMIZED POLES FOR TWO-POINT BOUNDARY VALUE PROBLEMS*

JEAN-PAUL BERRUT† AND HANS D. MITTELMANN‡

Abstract. An algorithm is proposed which improves upon the polynomial pseudospectral method for solving linear two-point boundary value problems. In the latter, the collocation points are the vertical projection onto the interval of points equidistant or nearly equidistant on the circle, and they therefore accumulate in the vicinity of the extremities of the interval. Thus the method is well-suited for solving problems whose solutions have boundary layers but not as good at approximating solutions with large gradients (shocks) away from the extremities of their domain of definition. Our idea is to modify the polynomial ansatz by attaching a denominator so as to make it a *rational* interpolant. The denominator is then successively optimized in an iterative procedure with each step consisting of the solution of two problems: an optimization of the denominator for *given* values of the approximation to the solution u at the interpolation points and a collocation in the linear space of the rational interpolants with the just obtained *fixed* denominator to obtain new approximate values of u. We show the efficiency of a Galerkin version of the method and discuss the power of the collocation version with several numerical examples.

Key words. two-point boundary value problems, linear rational collocation, pole optimization

AMS subject classifications. Primary, 65L10; Secondary, 41A20, 65D05

PII. S106482750036615X

1. Introduction.

The problem. Our aim in the present work is to improve upon the polynomial pseudospectral method for solving linear two-point boundary value problems (BVPs)

$$u''(x) + p(x)u'(x) + q(x)u(x) = f(x), \qquad x \in (-1,1), \tag{1.1a}$$

$$u(-1) = u_\ell, \quad u(1) = u_r, \tag{1.1b}$$

where all arising functions belong to $C^\infty[-1,1]$ and u_ℓ and u_r are given real numbers. We assume that p, q, and f are such that the problem is well-posed. (For conditions guaranteeing the latter, see, e.g., [Kel, p. 12], [As-Ma-Ru, p. 88], or [Sto-Bul].) The generalization of the method to be presented below to nonlinear problems is straightforward but does not bring more insight.

The version of the pseudospectral method we have in mind is that consisting of replacing the solution in (1.1a) by an interpolating polynomial in Lagrangian form between well-chosen points and collocating at those same points. Since such points are vertical projections on the interval of points (nodes) equidistant or nearly equidistant on the circle, they accumulate in the vicinity of the extremities of the interval. As a consequence, the method is well-suited for solving problems whose solutions have boundary layers. (This property can even be accentuated by conformally shifting the points to make them more tilted toward the extremities, e.g., by a sine-map

*Received by the editors February 3, 2000; accepted for publication (in revised form) January 25, 2001; published electronically September 26, 2001.

http://www.siam.org/journals/sisc/23-3/36615.html

†Département de Mathématiques, Université de Fribourg, CH–1700 Fribourg/Pérolles, Switzerland (jean-paul.berrut@unifr.ch). The work of this author was supported by Swiss National Science Foundation grant 21–59272.99.

‡Department of Mathematics, Arizona State University, Tempe, AZ 85287-1804 (mittellman@asu.edu).

[Tan-Tru].) Correspondingly, however, the center nodes are about equidistant and further from each other than the same number of nodes equidistant on the whole interval. As a consequence, the pseudospectral method has a hard time approximating solutions with large gradients (shocks) away from the extremities of their domain of definition.

Several methods can be used to cope with this difficulty. The most efficient ones, as measured by the size of the error, are probably methods where the solution is not approximated by a global but by a piecewise interpolant (see, e.g., [As-Ch-Ru, Bad-Asc, Lee-Gre]). Here we are interested only in methods which preserve the infinite differentiability of the solution. One of them consists of constructing and switching in an analytic function that displaces the interpolation points so as to have them concentrate close to the abscissae where the shocks arise [Mu-Hu-Sl]. To obtain this function, the method solves a sequence of problems for values of a parameter corresponding to stiffer and stiffer problems.

We will instead develop a method that improves on the pseudospectral (polynomial) method without any change in the problem nor in the interpolation/collocation points. The idea is to modify the ansatz by attaching a denominator to the polynomial so as to make it a *rational* interpolant, as in [Ber-Mit2] and [Ba-Be-Du]. The denominator is then successively optimized in an iterative procedure with each step consisting of the solution of two problems: an optimization of the denominator for *given* values of the approximation to u at the interpolation points and a collocation in the linear space of the rational interpolants with the just obtained *fixed* denominator to obtain new approximate values of u.

We will describe the methods for solving these two problems in section 2. Section 3 introduces the then very simple full algorithm, whose viability is motivated in section 4 by a theorem proving that a corresponding Galerkin algorithm yields a sequence of approximations with a certainly nonincreasing, and in most cases decreasing, error in the energy norm. Finally, in section 5, we report on numerical experiments that demonstrate the efficiency of the algorithm.

2. The ingredients of the solution.

As just mentioned in the introduction, the method presented here consists of the iterative application of two algorithms which we describe in this section.

2.1. The linear rational collocation method for boundary value problems.

This generalization, suggested in [Ber-Bal] and [Ba-Be-Du], of the now classical polynomial pseudospectral method is the application to BVPs of the corresponding method for time evolution problems [Bal-Ber2]. It is based on the fact [Ber2, Ber-Mit1] that every rational function $r \in \mathcal{R}_{N,N}$ interpolating a continuous function f between interpolation points $x_0, \dots, x_N$ can be written in its *barycentric form*

$$r(x) = \sum_{j=0}^{N} \frac{\beta_j}{x - x_j} f(x_j) \Bigg/ \sum_{j=0}^{N} \frac{\beta_j}{x - x_j} \tag{2.1}$$

for some (nonunique) numbers β_j, one per node, called *weights* of the interpolant. Here $\mathcal{R}_{m,n}$ denotes the set of all rational functions with numerator degree $\leq m$ and denominator degree $\leq n$. The polynomial interpolant is the special case of (2.1) in which the β_j's are proportional to the barycentric weights

$$w_j := 1 \Big/ \prod_{k \neq j} (x_j - x_i)$$

of polynomial interpolation [Hen]. For instance, for equidistant points the w_j's are proportional to $(-1)^j\binom{N}{j}$, for Čebyšev points $\cos\phi_j$ of the first kind they are proportional to $(-1)^j\sin\phi_j$ and for Čebyšev points of the second kind they are proportional to $(-1)^j\eta_j$, with $\eta_j = 1$ for all j but at the boundary points, where $\eta_0 = \eta_N = 1/2$. In view of the presence in the problem of the boundary values (1.1b) we will restrict ourselves here to sets of nodes containing the extremities -1 and 1, i.e., Lobatto points (and in particular to Čebyšev points of the second kind in numerical computations).

For fixed $\beta := [\beta_0, \ldots, \beta_N]^T$ the set of all interpolants (2.1) is a linear space, which we denote by $\mathcal{R}_N^{(\beta)}$. The functions

$$L_j^{(\beta)}(x) := \frac{\beta_j}{x - x_j}\Bigg/\sum_{k=0}^{N}\frac{\beta_k}{x - x_k}, \quad j = 0, 1, \ldots, N,$$

make up a basis for this space and they satisfy the Lagrange property

$$L_j^{(\beta)}(x_i) = \delta_{ij}. \tag{2.2}$$

In the *linear rational collocation method* (in barycentric form) for the nodes x_j, one tries to find u as an interpolant

$$\widetilde{u}(x) = \sum_{j=0}^{N}\widetilde{u}_j L_j^{(\beta)}(x) \in \mathcal{R}_N^{(\beta)}$$

for some *given* weights β and some unknown values $\widetilde{u}_j$ at the x_j's, one inserts $\widetilde{u}$ into (1.1a), and one collocates at the same interior x_j's for simplicity. (Collocation points different from the interpolation points, as in [Fun], are equally possible.) This yields the following linear system of equations for the $\widetilde{u}_j$:

$$\sum_{j=0}^{N}\widetilde{u}_j L_j^{(\beta)''}(x_i) + p(x_i)\sum_{j=0}^{N}\widetilde{u}_j L_j^{(\beta)'}(x_i) + q(x_i)\sum_{j=0}^{N}\widetilde{u}_j L_j^{(\beta)}(x_i) = f(x_i), \tag{2.3}$$
$$i = 1, \ldots, N-1, \qquad \widetilde{u}_0 = u_r, \widetilde{u}_N = u_\ell.$$

In order to write this expression in a more concise way, we introduce the following vectors and matrices in $\mathbb{R}^{N-1}$ (resp., $\mathbb{R}^{(N-1)\times(N-1)}$):

$$\begin{aligned}
&\widetilde{\mathbf{u}} := [\widetilde{u}_1, \widetilde{u}_2, \ldots, \widetilde{u}_{N-1}]^T,\\
&\mathbf{D}^{(1)} = \big(D_{ij}^{(1)}\big), \quad D_{ij}^{(1)} := L_j^{(\beta)'}(x_i),\\
&\mathbf{D}^{(2)} = \big(D_{ij}^{(2)}\big), \quad D_{ij}^{(2)} := L_j^{(\beta)''}(x_i),\\
&\mathbf{P} := \operatorname{diag}\big(p(x_i)\big), \quad \mathbf{Q} := \operatorname{diag}\big(q(x_i)\big),\\
&\mathbf{f} := [f(x_i) - u_r\big(L_0^{(\beta)''}(x_i) + p(x_i)L_0^{(\beta)'}(x_i)\big) - u_\ell\big(L_N^{(\beta)''}(x_i) + p(x_i)L_N^{(\beta)'}(x_i)\big)]^T,\\
&\qquad\quad i,\ j = 1, \ldots, N-1.
\end{aligned}$$

In view of (2.2), the system (2.3) for the unknown values $\widetilde{\mathbf{u}}$ of the approximant then reads $\mathbf{A}\widetilde{\mathbf{u}} = \mathbf{f}$ with

$$\mathbf{A} := \mathbf{D}^{(2)} + \mathbf{P}\mathbf{D}^{(1)} + \mathbf{Q}.$$

Despite its large condition number for N large, it can be solved very precisely by Gaussian elimination [Ber1, Tan-Tru], for only the differentiation operator $\mathbf{A}$ is ill-conditioned, not the integration operator $\mathbf{A}^{-1}$. The system can often also be solved efficiently via iterative methods [Ber1], although the ill-conditioned $\mathbf{A}$ then slows down the convergence. (A conformal shift of the points can improve on this; see [Ber-Bal].) In our calculations we have alleviated the instability by using the modified Schneider–Werner formulae [Bal-Ber1, Bal]

$$D_{ij}^{(1)} = \begin{cases} \dfrac{\beta_j/\beta_i}{x_i - x_j}, & i \neq j, \\ -\displaystyle\sum_{k \neq i} D_{ik}^{(1)}, & i = j, \end{cases}$$

and

$$D_{ij}^{(2)} = \begin{cases} 2D_{ij}^{(1)}\Big(D_{ii}^{(1)} - \dfrac{1}{x_i - x_j}\Big), & i \neq j, \\ -\displaystyle\sum_{k \neq i} D_{ik}^{(2)}, & i = j \end{cases}$$

for the differentiation matrices.

In the polynomial case ($\beta_j = w_j$, all j) and with the interpolation points used here, the convergence of $\widetilde{u}$ toward the exact solution u is *exponential* if p, q, and f are analytic in an ellipse containing $[-1, 1]$. This can be seen through subtraction (and use of the exponential convergence of the interpolant of f) if p and q are constant, by more elaborate theorems [Can-Qua] in general cases. However, this fast convergence may show up only after too large an N for practical purposes if u has huge gradients (see the introduction in [Ber-Mit2]). For error bounds through estimates of the norm of the inverse operator, see the work by Wright and collaborators, from [Cru-Wri] to [Ahm-Wri].

2.2. Optimal attachment of poles to the interpolating polynomial. It is intuitively clear, and well known in practice, that rational interpolation can better accommodate large gradients. (For literature on rational interpolation, see the catalogue [Gro].) However, at least for small numbers of nodes, the classical rational interpolation problem (interpolate a given function, here $\widetilde{u}$, between N points by a $r \in \mathcal{R}_{m,n}$ with $m + n = N$) is hampered by two drawbacks: the nonexistence of the solution in certain cases, which shows itself in the occurrence of "unattainable points" [Sto, p. 56], and the possible presence of poles in the interval of interpolation, a common phenomenon for N small.

To overcome these difficulties, we have suggested in [Ber-Mit2] to complement the interpolating polynomial $\widetilde{u}$ with a denominator with, say, P poles z_ℓ, $\ell = 1, \ldots, P$ (and corresponding modification of the numerator so as to maintain interpolation). In practice, if the real part is to lie in the interval of interpolation, the poles will be chosen as pairs of conjugate complex numbers, so as to stay with real interpolants.

It has been noted in [Ber3] ([Ber-Mit2] contains a more obvious derivation) that introducing preassigned poles is easily achieved in the barycentric setting by multiplying the weights w_j of polynomial interpolation by a multiple of

$$d_j := \prod_{\ell=1}^{P} (x_j - z_\ell), \quad j = 0, \ldots, N. \tag{2.4}$$

By writing r as

$$r(x) := \sum_{j=0}^{N} \frac{w_j \prod_{\ell=1}^{P}\Bigl(1 - \frac{x_j}{z_\ell}\Bigr)}{x - x_j} \widetilde{u}_j \Bigg/ \sum_{j=0}^{N} \frac{w_j \prod_{\ell=1}^{P}\Bigl(1 - \frac{x_j}{z_\ell}\Bigr)}{x - x_j},$$

one sees that the polynomial is the case where all z_ℓ are at infinity. The suggestion in [Ber-Mit2] is to move them from there to an optimal position where they minimize some error functional, which we take here as the norm

$$J(\mathbf{z}) := \|r'' + pr' + qr - f\|_\infty, \quad \mathbf{z} := [z_1, \ldots, z_P]^T, \tag{2.5}$$

of the *residual* of the differential equation for the approximation r with given values $\widetilde{\mathbf{u}}$ of the solution u at the x_j's. *The optimization can only decrease the value of J*, since the interpolation polynomial belongs to the feasible set.

Note that the interpolated values $\widetilde{u}_j$ at the nodes are not modified as one displaces the poles: interpolation is warranted by the barycentric formula [Wer, Ber2, Ber-Mit1]—r' and r'' do change, however, so that r no longer satisfies (2.3).

Optimizing the poles z_ℓ is a nonlinear problem to be solved by iteration. There is always an optimal $\mathbf{z}$, but, at least in special cases, there can be several of them. (Think of the case in which all functions arising in (1.1a) are constant.) Whether the optimal r is unique is an open question [Ber-Mit2]. Nevertheless, in every undetermined case among our many tests (there were very few such cases, and none for N large enough), the optimal set was a continuum and the multiplicity could easily be detected from the divergence of the optimization procedure.

3. The linear rational pseudospectral method with iteratively optimized poles. The algorithm we suggest here for solving (1.1a) improves iteratively upon the polynomial pseudospectral method. It consists of recursively performing the two methods described in section 2.

Let the $N+1$ interpolation points $x_0, \ldots, x_N$ be given, as well as the number P of poles to be optimized, which are first supposed at infinity (if no information on their final location is known at the onset). For $k = 1, 2, \ldots$, repeat the following steps.

Step 1. Compute the approximate solution $\widetilde{\mathbf{u}}^{(k)} = [\widetilde{u}_1^{(k)}, \ldots, \widetilde{u}_{N-1}^{(k)}]^T$ of (1.1a) by the linear rational collocation method with $\beta_j = w_j d_j$, d_j from (2.4) ($d_j \equiv 1$ for $k = 1$). This modifies $\widetilde{\mathbf{u}}$ (for $k > 1$) but not the poles $\mathbf{z}$ nor the weights β.

Step 2. For the $\widetilde{\mathbf{u}}^{(k)}$ inherited from Step 1, optimize the location of the poles $\mathbf{z}$ by minimizing $J(\mathbf{z})$. This changes β, but not $\widetilde{\mathbf{u}}^{(k)}$, and yields a new interpolant $\widehat{u}^{(k)}$ of the latter values.

When do we stop? Roughly speaking, we stop when the decrease in J becomes too small in comparison with the cost of one more step of the algorithm.

The recurrence of Step 2 makes the algorithm costly. However, for a given problem and a given N, the cost of the optimized solution is a constant multiple of the cost of the polynomial method without memory increase, whereas reaching a better precision by increasing N requires at least a quadratic increase in computing time (when using iteration methods for solving the systems of equations) and memory. Moreover, at the outcome, when β and $\widetilde{\mathbf{u}}$ have been computed, evaluating $\widetilde{u}$ by the formula (2.1) is exactly as expensive as evaluating the polynomial solution. The algorithm presented here may therefore be especially interesting in cases in which the time for finding the solution is not very relevant, but the latter must be evaluated a great many times, as in animated graphics or in the use of $\widetilde{u}$ as the reference solution in optimal control problems.

Also notice that in certain cases the location of the poles of u may be directly read from the equations (equations belonging to the Fuchsian class) or approximated with the WKB theory [Wei]. In which cases the β_j are known, there is no need for optimization and Step 1 yields $\widetilde{u}$: this is precisely the method introduced in [Ba-Be-Du].

4. Motivation: A corresponding Galerkin method. Why should the method work, i.e., bring improvement as compared to the classical pseudospectral solution? A corresponding Galerkin method, more complicated and computationally more expensive, gives some indication.

Denote by L the operator which to every function u in some appropriate space V associates the function on the left-hand side of (1.1a). Then a weak form of the latter consists of finding $u \in V$ for which

$$a(u,v) := (Lu, Lv) = (f, Lv) \qquad \forall\; v \in V,$$

where (,) is the L_2-scalar product in V. (Notice that, in contrast with the classical Galerkin method, we also apply L to the test functions v.) In an appropriate space, $a(u,v)$ is a symmetric positive definite form which induces the *energy norm*

$$\|v\|_a^2 := a(v,v) = \|Lv\|_2^2.$$

We may now introduce the linear rational Galerkin solution of (1.1a) as the function $\widetilde{u} \in \mathcal{R}_N^{(\beta)}$ such that

$$a(\widetilde{u}, \widetilde{v}) = (f, L\widetilde{v}) \qquad \forall\; \widetilde{v} \in \mathcal{R}_N^{(\beta)}.$$

$\widetilde{u}$ exists and it is unique because of the symmetry and V-ellipticity of a, and it notoriously possesses the important property of minimizing the norm $\| \quad \|_a$ of the error in $\mathcal{R}_N^{(\beta)}$, i.e.,

$$\|\widetilde{u} - u\|_a = \min_{v \in \mathcal{R}_N^{(\beta)}} \|v - u\|_a. \tag{4.1}$$

This Galerkin method would replace Step 1 of the algorithm to yield another $\widetilde{u}^{(k)}$. In Step 2 we would simply change the norm in (2.5) from $\| \quad \|_\infty$ to $\| \quad \|_2$ when computing $\widehat{u}^{(k)}$. Indeed, since $Lu = f$, $\|Lr - f\|_2^2 = \|L(r-u)\|_2^2 = \|r - u\|_a^2$. And, again, the optimal attachment of the poles can only decrease J, not increase it.

Because of (4.1), the next Step 1 (with the new β) can in its turn lead only to a $\widetilde{u}^{(k+1)}$ with

$$\|\widetilde{u}^{(k+1)} - u\|_a \le \|\widehat{u}^{(k)} - u\|_a \le \|\widetilde{u}^{(k)} - u\|_a,$$

and so on. We therefore have the following result.

THEOREM. *The linear rational Galerkin method with successive optimization of the poles, as described above, yields, in the energy norm* $\| \quad \|_a$*, a distance decreasing sequence of approximations to the solution of* (1.1a).

Since the minimum property (4.1) does not hold for the collocation method, the above result is merely an indication for the success of our algorithm. Still, in all problems we have solved, the successive optimal attachment of the poles has resulted in a decrease of J.

It should be noted, however, that a smaller residual does not necessarily imply an approximation $\widetilde{u}$ that is everywhere closer to the exact solution u: on rare occasions, the error becomes larger at particular points when one takes too small an N (see Example 1 with $N = 12$ in Table 1).

5. Numerical examples. We now report on computations performed on four examples, taken from the literature for the sake of comparison: three are borrowed from [Gre], the last one from [As-Ch-Ru]. Graphs of the solutions can be found in those articles.

As mentioned in section 2.1, our interpolation/collocation points have been in all examples Čebyšev points of the second kind $x_j := \cos \frac{j\pi}{N}$, $j = 0, \dots, N$. We have solved the systems arising in the linear rational collocation method by means of Gaussian elimination, since for some difficult examples the simple iterative procedure of [Ber1] did not converge. The optimization of the poles in Step 2 of the algorithm has been performed as in [Ber-Mit2] by a discrete differential correction algorithm according to [Ka-Le-Ta] for small N, by the simulated annealing method of [C-M-M-R] for larger numbers of nodes. The L_∞-norm in (2.5) has been approximated by considering the values at the 100 equally spaced points

$$\widehat{x}_k = -\frac{5}{4} + \frac{k-1}{K-1}\frac{5}{2}, \qquad k = 1(1)K, \quad K = 100,$$

on the interval $[-5/4, 5/4]$ and computing the maximal absolute value at those $\widehat{x}_k$ lying in $[-1, 1]$. (Tests with 1000 points instead of 100 have shown that the results do not depend much on this number of points.)

The computations were performed in Fortran77 on HP-workstations.

Example 1. The first example is from the classical book [Sto-Bul]. Modified by the change of variable $x = (t+1)/2$ to take place on the interval $[-1, 1]$, it reads

$$2y''(t) - 200y(t) = 200\cos^2(\pi x) + \pi^2\cos(2\pi x),$$
$$y(-1) = y(1) = 0,$$

and its exact solution is

$$y(t) = \frac{e^{-20}}{1+e^{-20}}e^{20x} + \frac{1}{1+e^{-20}}e^{-20x} - \cos^2(\pi x).$$

When applying the algorithm with an increasing number of poles P, one notices that as soon as $P \geq 4$, four of the poles have the tendency to arrange themselves symmetrically about the origin. This is not surprising, in view of the symmetry of the problem with respect to the imaginary axis. Since the difficulty of the optimization (the delicate part of the whole algorithm) grows sharply with the number of variables, it is natural to set the poles in groups of four and to diminish that way the number of variables from 8 to 2. The results we have obtained are summarized in Table 1. The first column gives N, the number of nodes minus 1, the second the number P of optimized poles, the third the residual norm (2.5) achieved by the optimized rational, the fourth the maximum error of the latter as above but with $K = 1000$ and the last the location of one of the four poles—the others being the same with the three other combinations of signs.

As in all our tests, the residual norm is significantly larger than the maximal error at the nodes. The results with $P = 0$ for increasing N document the exponential convergence of the polynomial pseudospectral method. For every fixed N the first

Table 1
Results for Example 1.

N	P	res. norm	max. error	poles
8	0	1.625e1	8.091e − 03	
	4	3.797e − 03	1.287e − 05	1.533893345756 + .2662406988685i
12	0	6.614e − 01	1.253e − 04	
	4	1.218e − 04	7.488e − 07	1.697187497475 + .3184214263790i
	8	1.307e − 05	2.572e − 06	1.979358155787 + .5540074636114i
				2.199682034625 + .3486755194479i
16	0	8.482e − 03	7.756e − 07	
	4	3.613e − 07	6.600e − 10	2.020272398591 + .3769170961355i
	8	8.918e − 08	6.132e − 10	2.378484979985 + .4356917449111i
				2.354705789043 + .6625676279678i
20	0	4.373e − 05	2.230e − 09	
	4	5.561e − 10	7.491e − 13	2.374930136687 + .4292360606798i
	8	3.351e − 11	7.139e − 14	2.808579639418 + .1118393705060i
				2.463546688447 + .8565353449894i
24	0	1.061e − 07	6.334e − 13	
	4	1.057e − 11	2.331e − 15	2.867253541946 + .4497240781784i

set of four poles improves the residual by three to five orders of magnitude and the maximal error by two to three, a very significant improvement especially spectacular for small N if one thinks in relative terms. The next four poles diminish the error by another power of 10. Also note that the poles come to lie to the left and to the right of the interpolation interval, and quite far from it, in order to best help the large gradients at the extremities.

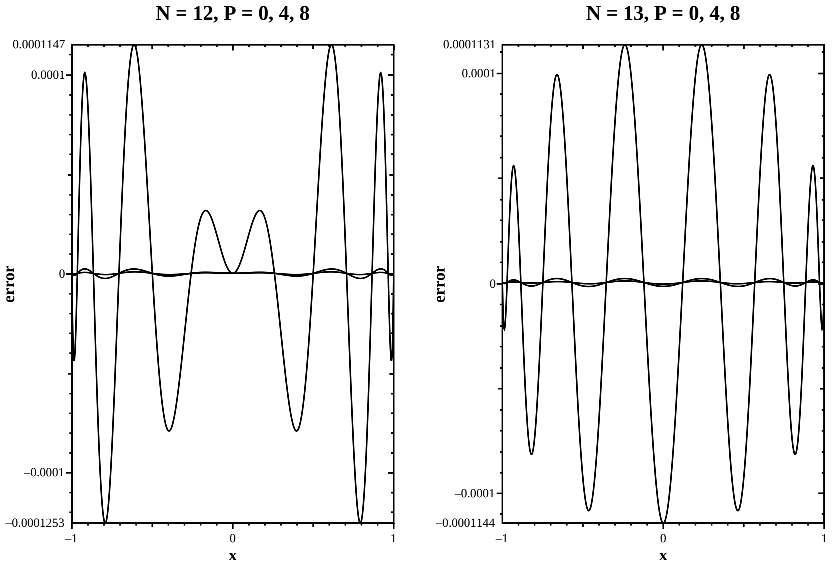


Fig. 1. *Error curves in Example* 1.

Figure 1 displays the improvement in the error curves for $N = 12$ and $N = 13$ with $P = 0$, 4, and 8 poles. The error pattern is very regular; the attached poles decrease the amplitude of the oscillations, whereas the abscissae of minimal absolute error do not move much.

Table 2
Results for Example 2.

ϵ	N	P	res. norm	max. error	poles
10	8	0	4.891e − 02	4.645e − 05	
		4	1.710e − 05	1.056e − 07	−3.822798824152 + .5226577159388i
					3.307176519618 + .8569093409655i
	16	0	8.700e − 09	1.448e − 12	
		4	1.614e − 12	4.996e − 15	−5.914776338155 + 1.410721876941i
					6.554571704828 + .4351084000440i
1’000	16	0	1.625e2	8.624e − 03	
		4	1.128e − 01	5.765e − 05	1.154539048844 + .9101294227218e − 01i
					−1.173916424929 + .9092666837900e − 01i
		8	1.718e − 02	8.555e − 06	−1.208444348538 + .8067319622931e − 01i
					1.190035589264 + .8067319622931e − 01i
					−1.175015446979 + .2416780824567i
					1.157107610750 + .2416780824567i
	32	0	4.228e − 03	7.986e − 09	
		4	1.280e − 07	4.924e − 11	−1.482150010195 + .1530297156978i
					1.459828803300 + .1533252394093i
		8	6.092e − 08	2.213e − 11	−1.443519502447 + .1348151975537i
					1.446860921493 + .1348151975537i
					−1.349189617520 + .5151973185520i
					1.388842892110 + .5151973185520i
100’000	64	0	1.243e + 03	6.303e − 04	
		4	5.896e − 02	1.841e − 06	−1.022761517226 + .1183432049352e − 01i
					1.021981828458 + .1171671875346e − 01i
		8	8.874e − 03	3.485e − 07	1.031577423248 + .1783176452066e − 01i
					1.041577580145 + .1783176452066e − 01i
					−1.033999150163 + .1828661035856e − 01i
					−1.039643789007 + .1828661035856e − 01i
	96	0	3.479e − 01	1.449e − 07	
		4	7.837e − 06	7.313e − 11	−1.049570684575 + .1712441907944e − 01i
					1.048060449714 + .1707769476758e − 01i
		8	7.785e − 07	8.839e − 12	1.060163908153 + .2324959883051e − 01i
					1.084678672960 + .2324959883051e − 01i
					−1.084874191336 + .2342311323713e − 01i
					−1.061543283858 + .2342311323713e − 01i

Example 2. The classical problem

$$u'' - \epsilon u = 0, \qquad y(-1) = 1, \quad y(1) = 2$$

displays boundary layers at the extremities of the interval. Written in such a way as to avoid overflow for large ϵ, the solution reads

$$u(x) = \frac{3}{2}\frac{e^{\delta(x-1)} + e^{-\delta(x+1)}}{1 + e^{-2\delta}} + \frac{1}{2}\frac{e^{\delta(x-1)} - e^{-\delta(x+1)}}{1 - e^{-2\delta}}, \qquad \delta := \sqrt{\epsilon}.$$

The results, as displayed in Table 2, again show that the first four poles improve the residual by three to almost five orders of magnitude and that the next four add one to two more orders. Also note that as ϵ increases and the layers become more pronounced, the poles move closer to the extremities of the interval, as could be expected.

Example 3. The third example in [Gre] is chosen in such a way that the solution is very oscillatory. Written in a more general way, the problem is

TABLE 3
Results for Example 3 with $a = 5$, $N = 16$, and various b.

b	P	res. norm	max. error	poles
1	0	7.306e − 12	1.548e − 14	
	4	1.487e − 12	7.147e − 16	−3.969061491605 + 1.199190616608i
				−3.943735599518 + 1.613384723663i
3	0	2.557e − 09	5.940e − 13	
	4	2.146e − 12	9.298e − 16	−3.426065845191 + 3.480961608887i
				−3.914409512838 + 2.082303028107i
6	0	1.958e − 07	3.597e − 11	
	4	1.522e − 11	4.958e − 14	−2.963469235131 + 2.088228035742i
				−1.811764925632 + 3.440041199296i
12	0	1.300e − 03	3.042e − 07	
	4	7.130e − 08	1.910e − 10	−1.723326736175 + 2.022955508355i
				−.5129729688333 + 2.461677514156i
25	0	2.028e1	4.226e − 03	
	4	1.390e − 02	4.670e − 05	−.2205181839708e − 01 + .9416587006262i
				−.7564105358912 + .8763852719388i
50	0	4.483e2	6.021e − 01	
	4	1.642e1	4.370e − 01	−.7940107554821 + .1643196982088i
				.1168118878478 + .6287820805492i
100	0	9.562e2	1.494e + 00	
	4	9.521e1	1.188e + 00	.2857700467848 + 1.076182707013i
				−.9531755714520 + .4590791444501e − 01i

$$u''(x) + \frac{a}{2}u'(x) + \frac{b^2}{4}u(x) = -\frac{ab}{2}\cos(bx)e^{-ax},$$
$$y(0) = 0, \qquad y(1) = \sin be^{-a}$$

and its solution is given by

$$u(x) = \sin(bx)e^{-ax},$$

where a and b are two positive real parameters. b controls the frequency of the oscillations: the bigger b, the larger the number of oscillations and the steeper the function in each of the latter. As a consequence, the number of optimized poles should be increased in step with b. This, however, is not possible, in view of the difficulty of solving optimization problems with large numbers of variables.

The same change of variable as in Example 1 must be made. In Table 3 we give the numbers obtained with $a = 5$, $N = 16$, and increasing b's. They show that up to about $b = 25$ the optimal attachment of few poles yields a very significant improvement of the solution, a surprising and heartening result. With $b = 100$, the case considered by Greengard, the improvement is no longer as pronounced.

Example 4. Finally we comment on results with a problem containing a parameter which can make for a large slope in the interior of the interval [Hem]:

$$u''(x) + \epsilon x u'(x) = -\pi^2\cos(\pi x) - \epsilon\pi x\sin(\pi x),$$
$$y(-1) = -2, \qquad y(1) = 0.$$

The solution

Table 4
Results for Example 4.

ϵ	N	P	res. norm	max. error	poles
1'000	128	0	2.296e + 00	1.993e − 05	
		2	5.904e − 03	7.310e − 07	.6062920633592e − 08 + .1395206410123i
		4	2.288e − 04	1.110e − 07	.2974300028030e − 01 + .1420450960183i
					−.2974266689472e − 01 + .1420450654309i
		6	3.603e − 05	3.771e − 08	.4448526224808e − 01 + .1458761717085i
					−.5843683359291e − 01 + .1515584585260i
					−.7845635113085e − 02 + .1483673955646i
	256	0	9.595e − 08	5.107e − 15	
5'000	128	0	7.592e + 03	4.804e − 02	
		2	5.890e + 01	3.106e − 03	−.2094778341082e − 11 + .3545251135656e − 01i
		4	3.347e + 00	1.575e − 03	.1195181672283e − 01 + .3329382120232e − 01i
					−.1195181670675e − 01 + .3329382120563e − 01i
		6	8.687e − 01	4.942e − 04	.3075858832887e − 04 + .4052163419770e − 01i
					−.2069484691270e − 01 + .3827521509881e − 01i
					.2071355783766e − 01 + .3832010156375e − 01i
	256	0	1.079e + 02	1.212e − 04	
		2	5.306e − 02	7.308e − 06	.1377516301326e − 09 + .5484520957958e − 01i
		4	2.779e − 03	1.074e − 06	−.1359393469132e − 01 + .5674245816196e − 01i
					.1359390766860e − 01 + .5674245834173e − 01i
		6	1.397e − 04	1.285e − 07	.2343737623551e − 01 + .5762784375589e − 01i
					−.8821000910677e − 05 + .5787704711284e − 01i
					−.2344946569995e − 01 + .5764373665211e − 01i
	512	0	7.823e − 07	1.019e − 13	
10'000	128	0	3.443e + 04	1.591e − 01	
		2	1.695e + 02	8.342e − 03	.3037129481257e − 08 + .2076400110120e − 01i
		4	7.610e + 01	1.579e − 02	.6528469517688e − 02 + .2011661878536e − 01i
					−.6528516151182e − 02 + .2011660618498e − 01i
		6	1.057e + 00	3.371e − 03	.1723911982815e − 01 + .2502011085089e − 01i
					−.4885557531023e − 06 + .2517679851540e − 01i
					−.1723873248392e − 01 + .2502154883629e − 01i
	256	0	5.677e + 03	5.680e − 03	
		2	2.985e + 00	4.197e − 04	.1440703847366e − 06 + .2947450734705e − 01i
		4	1.197e − 01	1.146e − 04	.9344224596270e − 02 + .2962289496016e − 01i
					−.9345160516667e − 02 + .2962287501417e − 01i
	512	0	5.254e − 01	8.860e − 08	

$$u(x) = \cos \pi x + \frac{\operatorname{erf}(\delta x)}{\operatorname{erf}(\delta)}, \qquad \delta = \sqrt{\epsilon/2},$$

becomes steeper and steeper at zero as ϵ grows larger. We have solved the problem for $\epsilon = 100$, 500, 1'000, 5'000, and 10'000.

Some of our results with the larger ϵ are summarized in Table 4. Since for too small an N the optimization procedure may fail to converge [Ber-Mit2], we give numbers only for $N \geq 128$. They share some common features. For instance, for given ϵ and N, the imaginary parts of the optimal poles are quite close to one another. Moreover, if four poles are optimized, they have the tendency to gather as the vertices of a rectangle about the origin, where the maximum gradient arises.

As for the errors, the optimization improves the residual by 4–6 digits, much more than it does with the maximum error (less than 3 digits)—see the comment on the condition in the conclusion. Nevertheless, with $\epsilon = 5'000$ and $N = 64$ or $\epsilon = 10'000$ and $N = 128$, attaching poles decreases the maximum error more than doubling N!

Our results are not quite as good as those obtained in [As-Ch-Ru] with the same example. We recall, however, that they are not comparable, for our method yields

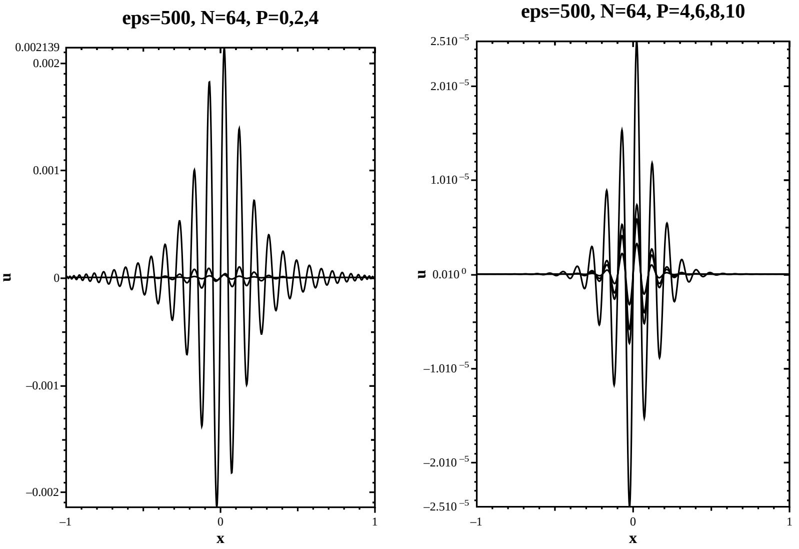


FIG. 2. *Error curves in Example* 4.

C^∞-approximations of the C^∞-solutions of the problems considered in the present work.

Finally, in Figure 2 we give error curves for a fixed ϵ-N pair and an increasing number of attached poles: even with the large gradient the error behaves nicely as P increases.

6. Conclusion.

In the present article we have applied to the solution of two-point boundary value problems the fact that rational interpolation is often more efficient than its polynomial counterpart. (We may mention in passing that this fact has been applied to the solution of Cauchy-type integral equations; see [Dri-Sri] and [Kai-Nod].) Our approach consists of an iterative improvement of the polynomial pseudospectral method, which is known to converge exponentially for good interpolation points and infinitely differentiable problems. After having obtained the solution at some (collocation) points by the polynomial method, we compute (one of the) rational interpolant(s) of these same values, with a denominator of given degree, by minimizing the residual of the differential equation. This defines the new linear space of all rationals interpolating between these same points and sharing that same denominator. We then just have to start again with the solution of the original equation in the new space, and so on. Although we can prove the effectiveness of a Galerkin version of the method, in practice we solve the problem with the much simpler collocation method.

The computed examples show the somewhat surprising result that one can usually gain between three and five digits of accuracy in comparison with classical polynomial collocation, and this almost independently of N. This is especially significant in cases where the precision obtained with the latter method is low and one does not want to increase the number of points so as to keep consequent evaluation of the solution as cheap as possible.

The placement of the poles is a very well-conditioned problem in the sense that many of their locations around the optimal one yield merely slightly larger residuals.

The tables show, however, that the gain in the residual error is usually much larger than the improvement in the precision of $\widetilde{u}$. This is probably due in part to the fact that the computation of the residual is smeared by the ill condition of the differentiation matrices, despite the improvement by the methods in [Bal-Ber1]. We hope to improve on this in the not too distant future.

Although not our purpose here, the method also seems applicable to problems whose solution u is not infinitely smooth, in fact, even when u displays discontinuities: it could then be compared with, e.g., methods which use especially constructed jump functions such as those advocated by Geer [Gee-Ban] or methods that call upon the help of several grids and fictious points, as suggested in [Dri-For].

The generalization of the method to elliptic problems in parallelpipeds seems straightforward. In the two-dimensional case, on a tensor grid in a rectangle, the ansatz would become

$$\widetilde{u}(x,y) = \sum_{j,k=0}^{N} \widetilde{u}_{jk} L_j^{(\beta)}(x) L_k^{(\gamma)}(y) \in \mathcal{R}_N^{(\beta)} \otimes \mathcal{R}_N^{(\gamma)},$$

and the new approximation of u would be obtained in Step 2 of the algorithm by minimizing $J(\mathbf{z}, \mathbf{t})$ with respect to the poles $\mathbf{z}$ and $\mathbf{t} = [t_1, \ldots, t_Q]$ in

$$r(x,y) := \frac{\displaystyle\sum_{j=0}^{N}\sum_{k=0}^{M} \frac{w_j \displaystyle\prod_{\ell=1}^{P}(x_j - z_\ell)}{x - x_j} \frac{s_k \displaystyle\prod_{m=1}^{Q}(y_k - t_m)}{y - y_k} \widetilde{u}_{jk}}{\displaystyle\sum_{j=0}^{N}\sum_{k=0}^{M} \frac{w_j \displaystyle\prod_{\ell=1}^{P}(x_j - z_\ell)}{x - x_j} \frac{s_k \displaystyle\prod_{m=1}^{Q}(y_k - t_m)}{y - y_k}}$$

(i.e., by optimizing the weights $\beta_j = w_j \prod_{\ell=1}^{P}(x_j - z_\ell)$ and $\gamma_k = s_k \prod_{m=1}^{Q}(y_k - t_m)$), where the s_k's denote the polynomial weights in the y-direction.

Another natural extension of the method is its application to time evolution partial differential equations: we intend to address the question in future work [Be-Mi-Tr].

Acknowledgment. The authors wish to thank the anonymous referees whose comments have improved the present work.

REFERENCES

[Ahm-Wri] A. H. Ahmed and K. Wright, *Error estimation for collocation solution of linear ordinary differential equations*, Comput. Math. Appl. Part B, 12 (1986), pp. 1053–1059.

[As-Ch-Ru] U. Ascher, J. Christiansen, and R. D. Russell, *A collocation solver for mixed order systems of boundary value problems*, Math. Comp., 33 (1979), pp. 659–679.

[As-Ma-Ru] U. M. Ascher, R. M. Mattheij, and R. D. Russell, *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations*, Prenctice-Hall, Englewood Cliffs, NJ, 1988.

[Bad-Asc] G. Bader and U. Ascher, *A new basis implementation for a mixed order boundary value ODE solver*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 483–500.

[Bal] R. Baltensperger, *Improving the accuracy of the matrix differentiation method for arbitrary collocation points*, Appl. Numer. Math., 33 (2000), pp. 143–149.

[Bal-Ber1] R. Baltensperger and J.-P. Berrut, *The errors in calculating the pseudospectral differentiation matrices for Čebyšev–Gauss–Lobatto points*, Comput. Math. Appl., 37 (1999), pp. 41–48.

[Bal-Ber2] R. Baltensperger and J.-P. Berrut, *The linear rational collocation method*, J. Comput. Appl. Math., to appear.

[Ba-Be-Du] R. Baltensperger, J.-P. Berrut, and Y. Dubey, *The Linear Rational Pseudospectral Method with Preassigned Poles*, in preparation.

[Ber1] J.-P. Berrut, *A Pseudospectral Čebyšev Method with Preliminary Transform to the Circle: Ordinary Differential Equations*, Report 252, Mathematisches Institut, Technische Universität München, München, Germany, 1990; revised Université de Fribourg, Fribourg/Pérolles, Switzerland, 1995.

[Ber2] J.-P. Berrut, *Linear rational interpolation of continuous functions over an interval*, in Mathematics of Computation 1943–1993: A Half–Century of Computational Mathematics, W. Gautschi, ed., Proc. Sympos. Appl. Math. 48, AMS, Providence, RI, 1994, pp. 261–264.

[Ber3] J.-P. Berrut, *The barycentric weights of rational interpolation with prescribed poles*, J. Comput. Appl. Math., 86 (1997), pp. 45–52.

[Ber-Bal] J.-P. Berrut and R. Baltensperger, *The linear rational pseudospectral method for boundary value problems*, BIT, to appear.

[Ber-Mit1] J.-P. Berrut and H. D. Mittelmann, *Lebesgue constant minimizing linear rational interpolation of continuous functions over the interval*, Comput. Math. Appl., 33 (1997), pp. 77–86.

[Ber-Mit2] J.-P. Berrut and H. D. Mittelmann, *Rational interpolation through the optimal attachment of poles to the interpolating polynomial*, Numer. Algorithms, 23 (2000), pp. 315–328.

[Be-Mi-Tr] J.-P. Berrut, H. D. Mittelmann, and M. R. Trummer, *The Linear Rational Pseudospectral Method with Optimized Poles for Time Evolution Problems*, in preparation.

[Can-Qua] C. Canuto and A. Quarteroni, *Variational methods in the theoretical analysis of spectral approximations*, in Spectral Methods for Partial Differential Equations, R. J. Voigt, D. Gottlieb, and M. Y. Hussaini, eds., SIAM, Philadelphia, 1984, pp. 55–78.

[C-M-M-R] A. Coran, M. Marchesi, C. Martini, and S. Ridella, *Minimizing multimodal functions of continuous variables with the "simulated annealing" algorithm*, ACM Trans. Math. Software, 13 (1987), pp. 262–280.

[Cru-Wri] D. M. Cruickshank and K. Wright, *Computable error bounds for polynomial collocation methods*, SIAM J. Numer. Anal., 15 (1978), pp. 134–151.

[Dri-For] T. A. Driscoll and B. Fornberg, *A block pseudospectral method for Maxwell's equations*. J. Comput. Phys., 140 (1998), pp. 47–65.

[Dri-Sri] M. A. Driscoll and R. P. Srivastav, *Rational function approximations in the numerical solution of Cauchy-type singular integral equations*, Comput. Math. Appl., 11 (1985), pp. 949–965.

[Fun] D. Funaro, *Spectral Elements for Transport-Dominated Equations*, Lecture Notes in Comput. Sci. Eng. 1, Springer-Verlag, Berlin, 1997.

[Gee-Ban] J. F. Geer and N. S. Banerjee, *Exponentially accurate approximations to piecewise smooth periodic functions*, J. Sci. Comput., 12 (1997), pp. 253–287.

[Gre] L. Greengard, *Spectral integration and two-point boundary value problems*, SIAM J. Numer. Anal., 28 (1991), pp. 1071–1080.

[Gro] E. Grosse, *A catalogue of algorithms for approximation*, in Algorithms for Approximation II, J. C. Mason and M. G. Cox, eds., Chapman and Hall, London, 1990, pp. 479–514.

[Hem] P. Hemker, *A Numerical Study of Stiff Two–Point Boundary Problems*, Mathematisch Centrum, Amsterdam, 1977.

[Hen] P. Henrici, *Essentials of Numerical Analysis*, John Wiley, New York, 1982.

[Kai-Nod] H. Kai and M. T. Noda, *Hybrid computation of Cauchy-type singular integral equations*, SIGSAM Bulletin, 32 (1998), pp. 59–60.

[Ka-Le-Ta] E. H. Kaufman Jr., D. J. Leeming, and G. D. Taylor, *Uniform rational approximation by differential correction and Remes-differential correction*, Internat. J. Numer. Methods Engrg., 17 (1981), pp. 1273–1280.

[Kel] H. B. Keller, *Numerical Methods for Two–Point Boundary–Value Problems*, Blaisdell, Waltham, MA, 1968.

[Lee-Gre] J.-Y. Lee and L. Greengard, *A fast adaptive numerical method for stiff two-point* *boundary value problems*, SIAM J. Sci. Comput., 18 (1997), pp. 403–429.

[Mu-Hu-Sl] L. S. Mulholland, W.-Z. Huang, and D. M. Sloan, *Pseudospectral solution of near-singular problems using numerical coordinate transformations based on adaptivity*, SIAM J. Sci. Comput., 19 (1998), pp. 1261–1289.

[Sto] J. Stoer, *Einführung in die Numerische Mathematik* I, 4th ed., Springer-Verlag, Berlin, 1972.

[Sto-Bul] J. Stoer and R. Bulirsch, *Numerische Mathematik* II, 3rd ed., Springer-Verlag, Berlin, 1990.

[Tan-Tru] T. Tang and M. R. Trummer, *Boundary layer resolving pseudospectral methods for singular perturbation problems*, SIAM J. Sci. Comput., 17 (1996), pp. 430–438.

[Wei] J. A. C. Weideman, *Spectral methods based on nonclassical orthogonal polynomials*, in Applications and Computation of Orthogonal Polynomials, W. Gautschi et al., eds., Internat. Ser. Numer. Math. 131, Birkhäuser, Basel, 1999, pp. 239–251.

[Wer] W. Werner, *Polynomial interpolation: Lagrange versus Newton*, Math. Comp., 43 (1984), pp. 205–217.

SIAM J. SCI. COMPUT.
Vol. 23, No. 3, pp. 976–999

A POSTERIORI ERROR CONTROL OF FINITE ELEMENT APPROXIMATIONS FOR COULOMB'S FRICTIONAL CONTACT*

PATRICE COOREVITS†, PATRICK HILD‡, AND MOHAMMED HJIAJ§

Abstract. This paper is concerned with the frictional unilateral contact problem governed by Coulomb's law. We define an a posteriori error estimator based on the concept of error in the constitutive relation to quantify the accuracy of a finite element approximation of the problem. We propose and study different mixed finite element approaches and discuss their properties in order to compute the estimator. The information given by the error estimates is then coupled with a mesh adaptivity technique which provides the user with the desired quality and minimizes the computation costs. The numerical implementation of the error estimator as well as optimized computations are performed.

Key words. Coulomb's friction law, a posteriori error estimates, finite elements, error in the constitutive relation, optimized computations

AMS subject classifications. 65N30, 74M10

PII. S1064827500375461

1. Introduction and problem setup.

The finite element method is currently used in the numerical realization of frictional contact problems occurring in many engineering applications (see [13]). An important task consists of evaluating numerically the quality of the finite element computations by using a posteriori error estimators. In elasticity, several different approaches leading to various error estimators have been developed: in particular, the error estimators introduced in [2] based on the residual of the equilibrium equations, the estimators linked to the smoothing of finite element stresses (see [22]), and the estimators based on the errors in the constitutive relation (see [14, 17]). A review of different a posteriori error estimators can be found in [21].

For frictionless unilateral contact problems, the residual based method was considered and studied in [3] (see also the references quoted therein) using a penalized approach, and the study of error in the constitutive relation was performed in [5].

In the present paper, we are interested in the more general and currently used Coulomb's frictional contact model, and we choose the estimators in the constitutive relation to quantify the accuracy of the finite element approximations. As far as we know, there is no literature concerning a posteriori error estimators for Coulomb's frictional unilateral contact model. The latter is recalled hereafter.

Let us be given an elastic body occupying a bounded domain Ω in $\mathbb{R}^2$ whose generic point is denoted $\boldsymbol{x} = (x_1, x_2)$. The boundary Γ of Ω is Lipschitz and divided as follows: $\Gamma = \overline{\Gamma_D} \cup \overline{\Gamma_N} \cup \overline{\Gamma_C}$, where Γ_D, Γ_N, and Γ_C are three open disjoint parts. We suppose that the displacement field is given on Γ_D. (To simplify, we assume afterwards that the body is clamped on Γ_D.) On the boundary part Γ_N, a density of forces denoted $\boldsymbol{F} \in (L^2(\Gamma_N))^2$ is applied. The third part is the segment Γ_C, in

*Received by the editors July 20, 2000; accepted for publication (in revised form) April 9, 2001; published electronically September 26, 2001.

http://www.siam.org/journals/sisc/23-3/37546.html

†Laboratoire de Mécanique et CAO, Université de Picardie–Jules Verne, 48 rue d'Ostende, 02100 Saint-Quentin, France (patrice.coorevits@insset.u-picardie.fr).

‡Laboratoire de Mathématiques, Université de Savoie et CNRS EP 2067, 73376 Le Bourget du Lac, France (hild@univ-savoie.fr).

§Department of Civil, Surveying, and Environmental Engineering, The University of Newcastle, University Drive, Callaghan NSW 2308, Australia (mohammed.hjiaj@newcastle.edu.au).

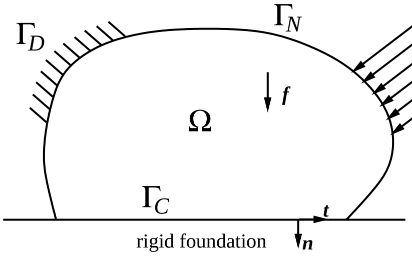


FIG. 1. *Setting of the problem.*

frictional contact with a rigid foundation (see Figure 1). The body Ω is submitted to a given density of volume forces $\boldsymbol{f} \in (L^2(\Omega))^2$. Let the notation $\boldsymbol{n} = (n_1, n_2)$ represent the unit outward normal vector on Γ and define the unit tangent vector $\boldsymbol{t} = (-n_2, n_1)$. Let us denote by $\mu > 0$ the friction coefficient on Γ_C.

The problem consists of finding the displacement field $\boldsymbol{u} : \Omega \longrightarrow \mathbb{R}^2$ and the stress tensor field $\boldsymbol{\sigma} : \Omega \longrightarrow \mathcal{S}_2$ satisfying (1.1)–(1.10):

$$\boldsymbol{\sigma}(\boldsymbol{u}) = \mathcal{C}\,\boldsymbol{\varepsilon}(\boldsymbol{u}) \quad \text{in } \Omega, \tag{1.1}$$

$$\mathbf{div}\,\boldsymbol{\sigma}(\boldsymbol{u}) + \boldsymbol{f} = 0 \quad \text{in } \Omega, \tag{1.2}$$

$$\boldsymbol{\sigma}(\boldsymbol{u})\boldsymbol{n} = \boldsymbol{F} \quad \text{on } \Gamma_N, \tag{1.3}$$

$$\boldsymbol{u} = 0 \quad \text{on } \Gamma_D, \tag{1.4}$$

where $\mathcal{S}_2$ stands for the space of second order symmetric tensors on $\mathbb{R}^2$, $\boldsymbol{\varepsilon}(\boldsymbol{u}) = \frac{1}{2}(\nabla\boldsymbol{u} + \nabla^T\boldsymbol{u})$ denotes the linearized strain tensor field, $\mathcal{C}$ is a fourth order symmetric and elliptic tensor of linear elasticity, and **div** represents the divergence operator of tensor valued functions.

In order to introduce the equations on Γ_C, let us adopt the following notation: $\boldsymbol{u} = u_n\boldsymbol{n} + u_t\boldsymbol{t}$ and $\boldsymbol{\sigma}(\boldsymbol{u})\boldsymbol{n} = \sigma_n(\boldsymbol{u})\boldsymbol{n} + \sigma_t(\boldsymbol{u})\boldsymbol{t}$. The equations modelling unilateral contact with Coulomb friction are as follows on Γ_C:

$$u_n \le 0, \tag{1.5}$$

$$\sigma_n(\boldsymbol{u}) \le 0, \tag{1.6}$$

$$\sigma_n(\boldsymbol{u})\, u_n = 0, \tag{1.7}$$

$$|\sigma_t(\boldsymbol{u})| \le \mu|\sigma_n(\boldsymbol{u})|, \tag{1.8}$$

$$|\sigma_t(\boldsymbol{u})| < \mu|\sigma_n(\boldsymbol{u})| \Longrightarrow u_t = 0, \tag{1.9}$$

$$|\sigma_t(\boldsymbol{u})| = \mu|\sigma_n(\boldsymbol{u})| \Longrightarrow \exists\lambda \ge 0 \ \text{ such that } \ u_t = -\lambda\sigma_t(\boldsymbol{u}). \tag{1.10}$$

The variational formulation of problem (1.1)–(1.10) has been obtained by Duvaut and Lions in [8]. It consists of finding $\boldsymbol{u}$ such that

$$\boldsymbol{u} \in \boldsymbol{K}_{ad}, \qquad a(\boldsymbol{u}, \boldsymbol{v} - \boldsymbol{u}) + j(\boldsymbol{u}, \boldsymbol{v}) - j(\boldsymbol{u}, \boldsymbol{u}) \ge L(\boldsymbol{v} - \boldsymbol{u}) \quad \forall \boldsymbol{v} \in \boldsymbol{K}_{ad}, \tag{1.11}$$

where

$$a(\boldsymbol{u}, \boldsymbol{v}) = \int_\Omega (\mathcal{C}\boldsymbol{\varepsilon}(\boldsymbol{u})) : \boldsymbol{\varepsilon}(\boldsymbol{v})\, d\Omega,$$

$$j(\boldsymbol{u}, \boldsymbol{v}) = \int_{\Gamma_C} \mu|\sigma_n(\boldsymbol{u})||v_t|\, d\Gamma,$$

$$L(\boldsymbol{v}) = \int_\Omega \boldsymbol{f}.\boldsymbol{v}\, d\Omega + \int_{\Gamma_N} \boldsymbol{F}.\boldsymbol{v}\, d\Gamma,$$

are defined for any $\boldsymbol{u}$ and $\boldsymbol{v}$ in

$$\boldsymbol{V} = \Big\{\boldsymbol{v} \in (H^1(\Omega))^2;\ \boldsymbol{v} = 0 \text{ on } \Gamma_D\Big\}.$$

The notation $H^1(\Omega)$ represents the standard Sobolev space; $\cdot$ and : stand for the inner product in $\mathbb{R}^2$ and $\mathcal{S}_2$, respectively. In (1.11), $\boldsymbol{K}_{ad}$ denotes the closed convex cone of admissible displacement fields satisfying the nonpenetration condition

$$\boldsymbol{K}_{ad} = \Big\{\boldsymbol{v} \in \boldsymbol{V};\ v_n \leq 0 \text{ on } \Gamma_C\Big\}.$$

The first existence result for problem (1.11) has been obtained in [20] when Ω is an infinitely long strip and if the friction coefficient of compact support in Γ_C is sufficiently small. The extension of these results to domains with smooth boundaries can be found in [12]. A recent improvement in [9] states existence when the friction coefficient μ is lower than $\frac{\sqrt{3-4\nu}}{2-2\nu}$, ν denoting Poisson's ratio in Ω $(0 < \nu < \frac{1}{2})$. When the loads $\boldsymbol{f}$ and $\boldsymbol{F}$ are not equal to zero, there is to our knowledge neither uniqueness result nor nonuniqueness example for problem (1.11). Let us mention that there exists several laws "mollifying" Coulomb's frictional contact model (see, e.g., [13, 19] and the references quoted therein) and that such regularizations lead to more existence and uniqueness properties.

Our paper is outlined as follows. In section 2, we first recall the convenient setting which consists of separating the kinematic conditions, the equilibrium equations, and the constitutive relations in order to define the error estimator and to study its properties. In section 3, we propose two mixed finite element methods for Coulomb's frictional unilateral contact problem. We prove the existence of solutions and we study the discrete frictional contact properties satisfied by such solutions. Section 4 is concerned with the practical construction of such an estimator. In section 5, several numerical studies in which we compute and couple the estimator with a mesh adaptivity procedure are performed.

2. The error estimator for Coulomb's frictional contact problem.

The aim of this section is to introduce the concept of error in the constitutive relation for the frictional unilateral contact problem. Before defining the estimator, let us begin with some useful setting and notation.

2.1. The appropriate setting for error in the constitutive relation.

To define the error in the constitutive relation, the contact part Γ_C is considered as in [15, 5] as an interface on which two unknowns $\boldsymbol{w}$ (displacement field) and $\boldsymbol{r}$ (density of surface forces due to the frictional contact with the rigid foundation) are to be found. If $\boldsymbol{n} = (n_1, n_2)$ and $\boldsymbol{t} = (-n_2, n_1)$ stand for the unit outward normal and tangent on Γ, we adopt afterwards the notation $\boldsymbol{z} = z_n\boldsymbol{n} + z_t\boldsymbol{t}$ for any vector $\boldsymbol{z}$.

The unilateral contact problem with Coulomb's friction law (1.1)–(1.10) is reformulated by using these quantities, and it consists of finding the displacement field $\boldsymbol{u}$ on Ω, the stress tensor field $\boldsymbol{\sigma}$ on Ω, and $\boldsymbol{w}$, $\boldsymbol{r}$ on Γ_C satisfying the following equations (2.1)–(2.9).

- The displacement fields $\boldsymbol{u}$ and $\boldsymbol{w}$ verify the kinematic conditions

$$\boldsymbol{u} = 0 \text{ on } \Gamma_D \quad \text{and} \quad \boldsymbol{w} = \boldsymbol{u} \text{ on } \Gamma_C. \tag{2.1}$$

- The fields $\boldsymbol{\sigma}$ and $\boldsymbol{r}$ satisfy the equilibrium equation

$$-\int_\Omega \boldsymbol{\sigma} : \boldsymbol{\varepsilon}(\boldsymbol{v})\, d\Omega + \int_\Omega \boldsymbol{f}.\boldsymbol{v}\, d\Omega + \int_{\Gamma_N} \boldsymbol{F}.\boldsymbol{v}\, d\Gamma + \int_{\Gamma_C} \boldsymbol{r}.\boldsymbol{v}\, d\Gamma = 0 \quad \forall \boldsymbol{v} \in \boldsymbol{V}. \tag{2.2}$$

- The fields $\boldsymbol{\sigma}$ and $\boldsymbol{u}$ are linked by the constitutive law of linear elasticity:

$$\boldsymbol{\sigma} = \mathcal{C}\boldsymbol{\varepsilon}(\boldsymbol{u}). \tag{2.3}$$

- The displacement field $\boldsymbol{w} = w_n\boldsymbol{n} + w_t\boldsymbol{t}$ and the density of forces $\boldsymbol{r} = r_n\boldsymbol{n} + r_t\boldsymbol{t}$ satisfy the unilateral contact conditions with Coulomb's friction law along Γ_C:

$$w_n \leq 0, \tag{2.4}$$
$$r_n \leq 0, \tag{2.5}$$
$$r_n w_n = 0, \tag{2.6}$$
$$|r_t| \leq \mu|r_n|, \tag{2.7}$$
$$|r_t| < \mu|r_n| \quad \Longrightarrow \quad w_t = 0, \tag{2.8}$$
$$|r_t| = \mu|r_n| \quad \Longrightarrow \quad \exists\lambda \geq 0 \text{ such that } w_t = -\lambda r_t. \tag{2.9}$$

Let us define the convex cones

$$K = \Big\{\boldsymbol{z}; \boldsymbol{z} = z_n\boldsymbol{n} + z_t\boldsymbol{t} \text{ such that } z_n \leq 0\Big\},$$
$$C_\mu = \Big\{\boldsymbol{s}; \boldsymbol{s} = s_n\boldsymbol{n} + s_t\boldsymbol{t} \text{ such that } s_n \leq 0 \text{ and } |s_t| \leq \mu|s_n|\Big\}.$$

Denoting by I_A the indicator function of the set A (i.e., $I_A(\boldsymbol{z}) = 0$ if $\boldsymbol{z} \in A$ and $I_A(\boldsymbol{z}) = +\infty$ if $\boldsymbol{z} \notin A$), it can easily be checked that the frictional contact conditions (2.4)–(2.9) can be also written in a more compact form:

$$I_K(\boldsymbol{w}) + I_{C_\mu}(\boldsymbol{r}) + \mu|r_n||w_t| + r_t w_t + r_n w_n = 0 \quad \text{on } \Gamma_C. \tag{2.10}$$

We begin with recalling the definition of an admissible pair.

Definition 2.1. *A pair* $\hat{s} = ((\hat{\boldsymbol{u}}, \hat{\boldsymbol{w}}), (\hat{\boldsymbol{\sigma}}, \hat{\boldsymbol{r}}))$ *is admissible if the kinematic conditions* (2.1) *and the equilibrium equations* (2.2) *are fulfilled.*

We are now in a position to define the estimator based on the error in the constitutive relation.

Definition 2.2. *Let* $\hat{s} = ((\hat{\boldsymbol{u}}, \hat{\boldsymbol{w}}), (\hat{\boldsymbol{\sigma}}, \hat{\boldsymbol{r}}))$ *be admissible. The error estimator* $e(\hat{s})$ *is as follows:*

$$e(\hat{s}) = \left(\|\hat{\boldsymbol{\sigma}} - \mathcal{C}\boldsymbol{\varepsilon}(\hat{\boldsymbol{u}})\|^2_{\sigma,\Omega} + 2\int_{\Gamma_C}\Big(I_K(\hat{\boldsymbol{w}}) + I_{C_\mu}(\hat{\boldsymbol{r}}) + \mu|\hat{r}_n||\hat{w}_t| + \hat{r}_t\hat{w}_t + \hat{r}_n\hat{w}_n\Big)\,d\Gamma\right)^{\frac{1}{2}}, \tag{2.11}$$

where the norm $\|.\|_{\sigma,\Omega}$ *on the stress tensor fields is defined by*

$$\|\boldsymbol{\sigma}\|_{\sigma,\Omega} = \left(\int_\Omega (\mathcal{C}^{-1}\boldsymbol{\sigma}) : \boldsymbol{\sigma}\, d\Omega\right)^{\frac{1}{2}}.$$

Let us notice that the function in the integral term of (2.11) is always nonnegative at $\boldsymbol{x} \in \Gamma_C$: it is equal to $+\infty$ if $\hat{\boldsymbol{w}}(\boldsymbol{x}) \notin K$ or $\hat{\boldsymbol{r}}(\boldsymbol{x}) \notin C_\mu$; otherwise it is nonnegative owing to $(\mu|\hat{r}_n||\hat{w}_t|+\hat{r}_t\hat{w}_t)(\boldsymbol{x}) \geq 0$ and $(\hat{r}_n\hat{w}_n)(\boldsymbol{x}) \geq 0$. To avoid more notation, we will skip over the regularity aspects of the functions defined on Γ_C which are beyond the scope of this paper and we write afterwards integral terms instead of duality pairings. The first natural property arising directly from the definition of $e(\hat{s})$ becomes the following property.

Property 2.3. *Let* $\hat{s}$ *be admissible. Then* $e(\hat{s}) = 0$ *if and only if* $\hat{s} = ((\hat{\boldsymbol{u}}, \hat{\boldsymbol{w}}), (\hat{\boldsymbol{\sigma}}, \hat{\boldsymbol{r}}))$ *is a solution to the reference problem* (2.1)–(2.9).

Let us define some quantities useful for the forthcoming study.

Definition 2.4. *Let $\hat{s}$ be admissible. The relative error $\epsilon(\hat{s})$ is as follows:*

$$\epsilon(\hat{s}) = \frac{e(\hat{s})}{\|\hat{\boldsymbol{\sigma}} + \mathcal{C}\boldsymbol{\varepsilon}(\hat{\boldsymbol{u}})\|_{\sigma,\Omega}}. \tag{2.12}$$

Given a part E of Ω, the local error contribution $\epsilon_E(\hat{s})$ is defined as

$$\epsilon_E(\hat{s}) = \frac{\left(\|\hat{\boldsymbol{\sigma}} - \mathcal{C}\boldsymbol{\varepsilon}(\hat{\boldsymbol{u}})\|^2_{\sigma,E} + 2\displaystyle\int_{\Gamma_C\cap E}\Big(I_K(\hat{\boldsymbol{w}}) + I_{C_\mu}(\hat{\boldsymbol{r}}) + \mu|\hat{r}_n||\hat{w}_t| + \hat{r}_t\hat{w}_t + \hat{r}_n\hat{w}_n\Big)\, d\Gamma\right)^{\frac{1}{2}}}{\|\hat{\boldsymbol{\sigma}} + \mathcal{C}\boldsymbol{\varepsilon}(\hat{\boldsymbol{u}})\|_{\sigma,\Omega}}, \tag{2.13}$$

where $\|\boldsymbol{\sigma}\|_{\sigma,E} = (\int_E(\mathcal{C}^{-1}\boldsymbol{\sigma}) : \boldsymbol{\sigma}\ d\Omega)^{\frac{1}{2}}$.

For the sake of simplicity of notations, we will write ϵ and ϵ_E instead of $\epsilon(\hat{s})$ and $\epsilon_E(\hat{s})$ in the following studies. It is straightforward that

$$\bigcup_{E_i\cap E_j=\emptyset,\ i\neq j} E_i = \Omega \quad\Longrightarrow\quad \epsilon^2 = \sum_i \epsilon^2_{E_i}.$$

2.2. Link between the estimator and the other errors. This part is concerned with the relation between the error in the constitutive law and the other errors. We suppose that a solution to the exact problem (2.1)–(2.9) exists which is satisfied when μ is small enough (see [9]). The next proposition generalizes former results (see [5]) obtained in the frictionless case (corresponding to $\mu = 0$).

Proposition 2.5. *Let $(\boldsymbol{u}, \boldsymbol{w}, \boldsymbol{\sigma}, \boldsymbol{r})$ be a solution to Coulomb's frictional contact problem* (2.1)–(2.9). *Let $\hat{s} = ((\hat{\boldsymbol{u}}, \hat{\boldsymbol{w}}), (\hat{\boldsymbol{\sigma}}, \hat{\boldsymbol{r}}))$ be admissible. Then*

$$\|\boldsymbol{\sigma} - \hat{\boldsymbol{\sigma}}\|^2_{\sigma,\Omega} + \|\boldsymbol{u} - \hat{\boldsymbol{u}}\|^2_{u,\Omega} + 2\mu\int_{\Gamma_C}(r_n - \hat{r}_n)(|\hat{w}_t| - |w_t|)\, d\Gamma \leq e^2(\hat{s}), \tag{2.14}$$

where the norm $\|.\|_{u,\Omega}$ on the displacement fields is defined by

$$\|\boldsymbol{u}\|_{u,\Omega} = \left(\int_\Omega(\mathcal{C}\boldsymbol{\varepsilon}(\boldsymbol{u})) : \boldsymbol{\varepsilon}(\boldsymbol{u})\ d\Omega\right)^{\frac{1}{2}} = \big(a(\boldsymbol{u}, \boldsymbol{u})\big)^{\frac{1}{2}}.$$

Consequently,

$$\|\boldsymbol{\sigma} - \hat{\boldsymbol{\sigma}}\|^2_{\sigma,\Omega} + 2\mu\int_{\Gamma_C}(r_n - \hat{r}_n)(|\hat{w}_t| - |w_t|)\, d\Gamma \leq e^2(\hat{s}), \tag{2.15}$$

$$\|\boldsymbol{u} - \hat{\boldsymbol{u}}\|^2_{u,\Omega} + 2\mu\int_{\Gamma_C}(r_n - \hat{r}_n)(|\hat{w}_t| - |w_t|)\, d\Gamma \leq e^2(\hat{s}). \tag{2.16}$$

Proof. We begin with noticing that the property obviously holds when $\hat{\boldsymbol{w}} \notin K$ or $\hat{\boldsymbol{r}} \notin C_\mu$ on a set of positive measure. In such a case, the error estimator is equal to infinity. Next, we then suppose that $\hat{\boldsymbol{w}} \in K$ and $\hat{\boldsymbol{r}} \in C_\mu$ almost everywhere. One immediately gets

$$\begin{aligned}\|\hat{\boldsymbol{\sigma}} - \mathcal{C}\boldsymbol{\varepsilon}(\hat{\boldsymbol{u}})\|^2_{\sigma,\Omega} &= \|\hat{\boldsymbol{\sigma}} - \boldsymbol{\sigma} + \mathcal{C}\boldsymbol{\varepsilon}(\boldsymbol{u} - \hat{\boldsymbol{u}})\|^2_{\sigma,\Omega}\\ &= \|\hat{\boldsymbol{\sigma}} - \boldsymbol{\sigma}\|^2_{\sigma,\Omega} + \|\boldsymbol{u} - \hat{\boldsymbol{u}}\|^2_{u,\Omega} + 2\int_\Omega(\hat{\boldsymbol{\sigma}} - \boldsymbol{\sigma}) : \boldsymbol{\varepsilon}(\boldsymbol{u} - \hat{\boldsymbol{u}})\ d\Omega.\end{aligned}$$

The stress fields $\boldsymbol{\sigma}$ and $\hat{\boldsymbol{\sigma}}$ satisfy the equilibrium equation (2.2) and the displacement fields $\boldsymbol{u}$ and $\hat{\boldsymbol{u}}$ verify the kinematic conditions (2.1). Hence

$$\|\hat{\boldsymbol{\sigma}} - \mathcal{C}\boldsymbol{\varepsilon}(\hat{\boldsymbol{u}})\|^2_{\sigma,\Omega} = \|\hat{\boldsymbol{\sigma}} - \boldsymbol{\sigma}\|^2_{\sigma,\Omega} + \|\boldsymbol{u} - \hat{\boldsymbol{u}}\|^2_{u,\Omega} + 2\int_{\Gamma_C} (\hat{\boldsymbol{r}} - \boldsymbol{r}).(\boldsymbol{w} - \hat{\boldsymbol{w}})\, d\Gamma. \tag{2.17}$$

Developing the integral term yields

$$\begin{aligned} &\int_{\Gamma_C} (\hat{\boldsymbol{r}} - \boldsymbol{r}).(\boldsymbol{w} - \hat{\boldsymbol{w}})\, d\Gamma \\ &\quad = \int_{\Gamma_C} \hat{r}_n w_n \, d\Gamma + \int_{\Gamma_C} \hat{r}_t w_t \, d\Gamma + \int_{\Gamma_C} r_n \hat{w}_n \, d\Gamma + \int_{\Gamma_C} r_t \hat{w}_t \, d\Gamma \\ &\quad\quad - \int_{\Gamma_C} r_n w_n \, d\Gamma - \int_{\Gamma_C} r_t w_t \, d\Gamma - \int_{\Gamma_C} \hat{r}_n \hat{w}_n \, d\Gamma - \int_{\Gamma_C} \hat{r}_t \hat{w}_t \, d\Gamma. \end{aligned} \tag{2.18}$$

Putting together (2.17) and (2.18) in the definition (2.11) of the estimator leads to

$$\begin{aligned} e^2(\hat{s}) = {} & \|\boldsymbol{\sigma} - \hat{\boldsymbol{\sigma}}\|^2_{\sigma,\Omega} + \|\boldsymbol{u} - \hat{\boldsymbol{u}}\|^2_{u,\Omega} \\ & + 2\int_{\Gamma_C} \hat{r}_n w_n \, d\Gamma + 2\int_{\Gamma_C} \hat{r}_t w_t \, d\Gamma + 2\int_{\Gamma_C} r_n \hat{w}_n \, d\Gamma + 2\int_{\Gamma_C} r_t \hat{w}_t \, d\Gamma \\ & - 2\int_{\Gamma_C} r_n w_n \, d\Gamma - 2\int_{\Gamma_C} r_t w_t \, d\Gamma + 2\int_{\Gamma_C} \mu |\hat{r}_n||\hat{w}_t| \, d\Gamma. \end{aligned}$$

Noting that $r_n \hat{w}_n \geq 0$, $\hat{r}_n w_n \geq 0$, and $r_n w_n = 0$ on Γ_C, we get

$$\begin{aligned} e^2(\hat{s}) \geq {} & \|\boldsymbol{\sigma} - \hat{\boldsymbol{\sigma}}\|^2_{\sigma,\Omega} + \|\boldsymbol{u} - \hat{\boldsymbol{u}}\|^2_{u,\Omega} \\ & + 2\int_{\Gamma_C} \hat{r}_t w_t \, d\Gamma + 2\int_{\Gamma_C} r_t \hat{w}_t \, d\Gamma - 2\int_{\Gamma_C} r_t w_t \, d\Gamma + 2\int_{\Gamma_C} \mu |\hat{r}_n||\hat{w}_t| \, d\Gamma. \end{aligned}$$

According to (2.7)–(2.9), the equality $-r_t w_t = \mu |r_n||w_t|$ holds on Γ_C. Moreover, $\boldsymbol{r} \in C_\mu$ and $\hat{\boldsymbol{r}} \in C_\mu$ lead to the bounds $r_t \hat{w}_t \geq -\mu |r_n||\hat{w}_t|$ and $\hat{r}_t w_t \geq -\mu |\hat{r}_n||w_t|$. Consequently,

$$e^2(\hat{s}) \geq \|\boldsymbol{\sigma} - \hat{\boldsymbol{\sigma}}\|^2_{\sigma,\Omega} + \|\boldsymbol{u} - \hat{\boldsymbol{u}}\|^2_{u,\Omega} + 2\mu \int_{\Gamma_C} (|r_n| - |\hat{r}_n|)(|w_t| - |\hat{w}_t|)\, d\Gamma.$$

The bound (2.14) is obtained thanks to $r_n \leq 0$ and $\hat{r}_n \leq 0$. Both bounds (2.15) and (2.16) are an obvious consequence. □

It is easy to check that no information on the sign of the integral term in (2.14) is available. This is not at all surprising because the evaluation of such a term corresponds also to the study of the uniqueness for the (quasi-)variational inequality (1.11) with classical arguments (i.e., by choosing and subtracting two solutions) which does not lead to a successful conclusion. Nevertheless, the following remark shows that the integral term can be bounded at least in a particular case.

Remark 2.6. If the exact solution and the admissible solution satisfy $w_t \geq 0$ and $\hat{w}_t \geq 0$ on Γ_C (or $w_t \leq 0$ and $\hat{w}_t \leq 0$ on Γ_C), and if the measure of Γ_D is positive, then inequality (2.14) becomes more relevant since the integral term in (2.14) can be estimated as follows:

$$\begin{aligned}\left|\int_{\Gamma_C}(r_n-\hat{r}_n)(|\hat{w}_t|-|w_t|)\,d\Gamma\right| &= \left|\int_{\Gamma_C}(r_n-\hat{r}_n)(\hat{w}_t-w_t)\,d\Gamma\right| \\ &\le \|r_n-\hat{r}_n\|_{H^{-\frac{1}{2}}(\Gamma_C)}\|w_t-\hat{w}_t\|_{H^{\frac{1}{2}}(\Gamma_C)} \\ &\le C\|\boldsymbol{\sigma}-\hat{\boldsymbol{\sigma}}\|_{(L^2(\Omega))^4}\|\boldsymbol{u}-\hat{\boldsymbol{u}}\|_{(H^1(\Omega))^2} \\ &\le C'\|\boldsymbol{\sigma}-\hat{\boldsymbol{\sigma}}\|_{\sigma,\Omega}\|\boldsymbol{u}-\hat{\boldsymbol{u}}\|_{u,\Omega} \\ &\le C''\Big(\|\boldsymbol{\sigma}-\hat{\boldsymbol{\sigma}}\|^2_{\sigma,\Omega}+\|\boldsymbol{u}-\hat{\boldsymbol{u}}\|^2_{u,\Omega}\Big),\end{aligned}$$

where $H^{\frac{1}{2}}(\Gamma_C)$ stands for a fractionally Sobolev space (see [1]) and $H^{-\frac{1}{2}}(\Gamma_C)$ is its dual space. The bounds of $\|r_n-\hat{r}_n\|_{H^{-\frac{1}{2}}(\Gamma_C)}$ and $\|w_t-\hat{w}_t\|_{H^{\frac{1}{2}}(\Gamma_C)}$ are obtained using Green's formula and the trace theorem, respectively. Moreover, the norms $\|.\|_{(H^1(\Omega))^2}$ and $\|.\|_{u,\Omega}$ are equivalent since meas$(\Gamma_D)>0$. In such a case, the integral term can be removed from (2.14), (2.15), and (2.16), and we come to the conclusion that there exists a positive constant C such that for small friction coefficients, $\|\boldsymbol{\sigma}-\hat{\boldsymbol{\sigma}}\|_{\sigma,\Omega}$ and $\|\boldsymbol{u}-\hat{\boldsymbol{u}}\|_{u,\Omega}$ can be bounded by $(1/\sqrt{1-\mu C})e(\hat{s})$. Concerning the general case, we think that one could reasonably expect that if the exact and admissible solutions are smooth enough and if the friction coefficient is small, then the integral term multiplied by 2μ is small in comparison with $\|\boldsymbol{u}-\hat{\boldsymbol{u}}\|^2_{u,\Omega}$ and $\|\boldsymbol{\sigma}-\hat{\boldsymbol{\sigma}}\|^2_{\sigma,\Omega}$.

Remark 2.7. If instead of Coulomb's law (2.10), one considers a Tresca's type friction law,

$$I_K(\boldsymbol{w})+I_C(\boldsymbol{r})+k|w_t|+r_tw_t+r_nw_n=0 \quad \text{on } \Gamma_C, \tag{2.19}$$

where $k\ge 0$ and where $C=\{\boldsymbol{s};\boldsymbol{s}=s_n\boldsymbol{n}+s_t\boldsymbol{t}$ such that $s_n\le 0$ and $|s_t|\le k\}$, then the problem (2.1)–(2.3), (2.19) admits a unique solution $(\boldsymbol{u}_k,\boldsymbol{w}_k,\boldsymbol{\sigma}_k,\boldsymbol{r}_k)$ and the bound

$$\|\boldsymbol{\sigma}_k-\hat{\boldsymbol{\sigma}}\|^2_{\sigma,\Omega}+\|\boldsymbol{u}_k-\hat{\boldsymbol{u}}\|^2_{u,\Omega}\le e^2(\hat{s}) \tag{2.20}$$

holds for any admissible $\hat{s}=((\hat{\boldsymbol{u}},\hat{\boldsymbol{w}}),(\hat{\boldsymbol{\sigma}},\hat{\boldsymbol{r}}))$.

Estimate (2.20) is obtained by following the same points as in the proof of estimate (2.14). In particular, if $k=0$ in (2.19) or equivalently $\mu=0$ in (2.10), we recover the frictionless unilateral contact model.

3. The discrete Coulomb's frictional contact problem.

In this section, we propose and study the properties of two mixed discrete finite element formulations for Coulomb's frictional contact in order to implement the error estimator. Let us mention that a detailed study of several (different) mixed finite element methods for frictionless and frictional contact problems can be found in [10, 11].

3.1. The mixed finite element formulations.

The body Ω is discretized by using a family of triangulations $(\mathcal{T}_h)_h$ made of finite elements of degree one. For technical purposes, we assume (in section 3.1 only) that $\overline{\Gamma_D}\cap\overline{\Gamma_C}=\emptyset$ which is generally not restrictive in engineering applications and that the bilinear form $a(.,.)$ is $\boldsymbol{V}$-elliptic. Let us denote by $h>0$ the discretization parameter representing the greatest diameter of a triangle in $\mathcal{T}_h$. The space approximating $\boldsymbol{V}$ becomes

$$\boldsymbol{V}_h=\Big\{\boldsymbol{v}_h;\ \boldsymbol{v}_h\in(\mathcal{C}(\overline{\Omega}))^2,\ \boldsymbol{v}_h|_T\in(P_1(T))^2\ \ \forall T\in\mathcal{T}_h,\ \boldsymbol{v}_h=0 \text{ on } \Gamma_D\Big\},$$

where $\mathcal{C}(\overline{\Omega})$ stands for the space of continuous functions on $\overline{\Omega}$, and $P_1(T)$ represents the space of polynomial functions of degree one on T. On the boundary of Ω, we still keep the notation $\boldsymbol{v}_h = v_{hn}\boldsymbol{n} + v_{ht}\boldsymbol{t}$ for every $\boldsymbol{v}_h \in \boldsymbol{V}_h$ and we denote by $(T_h)_h$ the family of monodimensional meshes on Γ_C inherited by $(\mathcal{T}_h)_h$.

We next introduce two convex sets of Lagrange multipliers denoted $\boldsymbol{M}'_h(g)$ and $\boldsymbol{M}''_h(g)$. The convex $\boldsymbol{M}'_h(g)$ is defined by $\boldsymbol{M}'_h(g) = M'_{hn} \times M'_{ht}(g)$, where

$$M'_{hn} = \Big\{\nu; \nu \in \mathcal{C}(\overline{\Gamma_C}),\ \nu|_S \in P_1(S)\ \ \forall S \in T_h,\ \nu \leq 0 \text{ on } \Gamma_C\Big\},$$

and for $g \in -M'_{hn}$, we define $M'_{ht}(g)$ as follows:

$$M'_{ht}(g) = \Big\{\nu; \nu \in \mathcal{C}(\overline{\Gamma_C}),\ \nu|_S \in P_1(S)\ \ \forall S \in T_h,\ |\nu| \leq g \text{ on } \Gamma_C\Big\}.$$

We denote by p the number of nodes of the triangulation on Γ_C and by $\psi_i, 1 \leq i \leq p$ the monodimensional basis functions on Γ_C. (The function ψ_i is continuous on Γ_C, linear on each segment of T_h, and equal to 1 at node i and to 0 at the other nodes.) The second convex $\boldsymbol{M}''_h(g)$ is given by $\boldsymbol{M}''_h(g) = M''_{hn} \times M''_{ht}(g)$ with

$$M''_{hn} = \left\{\nu; \nu \in \mathcal{C}(\overline{\Gamma_C}),\ \nu|_S \in P_1(S)\ \ \forall S \in T_h,\ \int_{\Gamma_C} \nu\psi_i \, d\Gamma \leq 0\ \ \forall 1 \leq i \leq p\right\}.$$

If $g \in -M''_{hn}$, $M''_{ht}(g)$ is given by

$$M''_{ht}(g) \\ = \left\{\nu; \nu \in \mathcal{C}(\overline{\Gamma_C}),\ \nu|_S \in P_1(S)\ \ \forall S \in T_h, \left|\int_{\Gamma_C} \nu\psi_i \, d\Gamma\right| \leq \int_{\Gamma_C} g\psi_i \, d\Gamma\ \ \forall 1 \leq i \leq p\right\}.$$

Next, the notation $\boldsymbol{M}_h(g) = M_{hn} \times M_{ht}(g)$ denotes either $\boldsymbol{M}'_h(g) = M'_{hn} \times M'_{ht}(g)$ or $\boldsymbol{M}''_h(g) = M''_{hn} \times M''_{ht}(g)$.

We then introduce an intermediary problem with a given slip limit $-\mu g_{hn}$, where $g_{hn} \in M_{hn}$. This problem is denoted by $P(g_{hn})$ and consists of finding $\boldsymbol{u}_h \in \boldsymbol{V}_h$ and $(\lambda_{hn}, \lambda_{ht}) \in M_{hn} \times M_{ht}(-\mu g_{hn}) = \boldsymbol{M}_h(-\mu g_{hn})$ such that

$$(P(g_{hn})) \quad \left\{\begin{array}{l} a(\boldsymbol{u}_h, \boldsymbol{v}_h) - \displaystyle\int_{\Gamma_C} \lambda_{hn} v_{hn} \, d\Gamma - \int_{\Gamma_C} \lambda_{ht} v_{ht} \, d\Gamma = L(\boldsymbol{v}_h)\ \ \forall \boldsymbol{v}_h \in \boldsymbol{V}_h, \\ \displaystyle\int_{\Gamma_C} (\nu_{hn} - \lambda_{hn}) u_{hn} \, d\Gamma + \int_{\Gamma_C} (\nu_{ht} - \lambda_{ht}) u_{ht} \, d\Gamma \geq 0 \\ \hfill \forall (\nu_{hn}, \nu_{ht}) \in \boldsymbol{M}_h(-\mu g_{hn}). \end{array}\right. \tag{3.1}$$

Problem $P(g_{hn})$ is the equivalent of finding a saddle-point $(\boldsymbol{u}_h, \lambda_{hn}, \lambda_{ht}) = (\boldsymbol{u}_h, \boldsymbol{\lambda}_h)$ in $\boldsymbol{V}_h \times \boldsymbol{M}_h(-\mu g_{hn})$ verifying

$$\mathcal{L}(\boldsymbol{u}_h, \boldsymbol{\nu}_h) \leq \mathcal{L}(\boldsymbol{u}_h, \boldsymbol{\lambda}_h) \leq \mathcal{L}(\boldsymbol{v}_h, \boldsymbol{\lambda}_h) \qquad \forall \boldsymbol{v}_h \in \boldsymbol{V}_h,\ \forall \boldsymbol{\nu}_h \in \boldsymbol{M}_h(-\mu g_{hn}),$$

where

$$\mathcal{L}(\boldsymbol{v}_h, \boldsymbol{\nu}_h) = \frac{1}{2} a(\boldsymbol{v}_h, \boldsymbol{v}_h) - \int_{\Gamma_C} \nu_{hn} v_{hn} \, d\Gamma - \int_{\Gamma_C} \nu_{ht} v_{ht} \, d\Gamma - L(\boldsymbol{v}_h).$$

By using classical arguments on saddle-point problems as Haslinger, Hlaváček, and Nečas [11, p. 338], we deduce that there exists such a saddle-point. The strict convexity of $a(.,.)$ implies that the first argument $\boldsymbol{u}_h$ is unique. Besides, the assumption $\overline{\Gamma_D} \cap \overline{\Gamma_C} = \emptyset$ allows us to write

$$\int_{\Gamma_C} \nu_{hn} v_{hn} \, d\Gamma - \int_{\Gamma_C} \nu_{ht} v_{ht} \, d\Gamma = 0\ \ \forall \boldsymbol{v}_h \in \boldsymbol{V}_h, \qquad \Longrightarrow \qquad \nu_{hn} = 0, \nu_{ht} = 0.$$

Consequently, the second argument $\boldsymbol{\lambda}_h$ is unique and $P(g_{hn})$ admits a unique solution.

It then becomes possible to define a map Φ_h as follows:

$$\begin{array}{rll} \Phi_h : & M_{hn} & \longrightarrow M_{hn}, \\ & g_{hn} & \longmapsto \lambda_{hn}, \end{array}$$

where $(\boldsymbol{u}_h, \lambda_{hn}, \lambda_{ht})$ is the solution of $P(g_{hn})$. The introduction of this map allows the definition of a discrete solution of Coulomb's frictional contact problem.

Definition 3.1. *Let $\boldsymbol{M}_h(g) = \boldsymbol{M}'_h(g)$ or $\boldsymbol{M}_h(g) = \boldsymbol{M}''_h(g)$. A solution of Coulomb's discrete frictional contact problem is the solution of $P(\lambda_{hn})$, where $\lambda_{hn} \in M_{hn}$ is a fixed point of Φ_h.*

Proposition 3.2. *Let $\boldsymbol{M}_h(g) = \boldsymbol{M}'_h(g)$ or $\boldsymbol{M}_h(g) = \boldsymbol{M}''_h(g)$. Then for any μ, there exists a solution to Coulomb's discrete frictional contact problem.*

Proof. To establish existence, we use Brouwer's fixed point theorem.

Step 1. We prove that the mapping Φ_h is continuous. Set

$$\tilde{\boldsymbol{V}}_h = \Big\{ \boldsymbol{v}_h \in \boldsymbol{V}_h;\ v_{ht} = 0 \text{ on } \Gamma_C \Big\}, \quad W_h = \Big\{ \nu; \nu \in \mathcal{C}(\overline{\Gamma_C}),\ \nu|_S \in P_1(S)\ \ \forall S \in T_h \Big\}.$$

Since $\overline{\Gamma_D} \cap \overline{\Gamma_C} = \emptyset$, it is easy to check that the definition of $\|.\|_{-\frac{1}{2},h}$ given by

$$\|\nu\|_{-\frac{1}{2},h} = \sup_{\boldsymbol{v}_h \in \tilde{\boldsymbol{V}}_h} \frac{\displaystyle\int_{\Gamma_C} \nu v_{hn}\, d\Gamma}{\|\boldsymbol{v}_h\|_1},$$

is a norm on W_h. The notation $\|.\|_1$ represents the $(H^1(\Omega))^2$-norm.

Let $(\boldsymbol{u}_h, \lambda_{hn}, \lambda_{ht})$ and $(\overline{\boldsymbol{u}_h}, \overline{\lambda_{hn}}, \overline{\lambda_{ht}})$ be the solutions of $(P(g_{hn}))$ and $(P(\overline{g_{hn}}))$, respectively. On the one hand, we get

$$a(\boldsymbol{u}_h, \boldsymbol{v}_h) - \int_{\Gamma_C} \lambda_{hn} v_{hn}\, d\Gamma = L(\boldsymbol{v}_h) \qquad \forall \boldsymbol{v}_h \in \tilde{\boldsymbol{V}}_h,$$
$$a(\overline{\boldsymbol{u}_h}, \boldsymbol{v}_h) - \int_{\Gamma_C} \overline{\lambda_{hn}} v_{hn}\, d\Gamma = L(\boldsymbol{v}_h) \qquad \forall \boldsymbol{v}_h \in \tilde{\boldsymbol{V}}_h.$$

Subtracting the previous equalities and using the continuity of the bilinear form $a(.,.)$ gives

$$\int_{\Gamma_C} (\lambda_{hn} - \overline{\lambda_{hn}}) v_{hn}\, d\Gamma = a(\boldsymbol{u}_h - \overline{\boldsymbol{u}_h}, \boldsymbol{v}_h) \le M \|\boldsymbol{u}_h - \overline{\boldsymbol{u}_h}\|_1 \|\boldsymbol{v}_h\|_1 \qquad \forall \boldsymbol{v}_h \in \tilde{\boldsymbol{V}}_h.$$

Hence, we get a first estimate

$$\|\lambda_{hn} - \overline{\lambda_{hn}}\|_{-\frac{1}{2},h} \le M \|\boldsymbol{u}_h - \overline{\boldsymbol{u}_h}\|_1. \tag{3.2}$$

On the other hand, we have from (3.1)

$$a(\boldsymbol{u}_h, \boldsymbol{v}_h) - \int_{\Gamma_C} \lambda_{hn} v_{hn}\, d\Gamma - \int_{\Gamma_C} \lambda_{ht} v_{ht}\, d\Gamma = L(\boldsymbol{v}_h) \qquad \forall \boldsymbol{v}_h \in \boldsymbol{V}_h, \tag{3.3}$$

$$a(\overline{\boldsymbol{u}_h}, \boldsymbol{v}_h) - \int_{\Gamma_C} \overline{\lambda_{hn}} v_{hn}\, d\Gamma - \int_{\Gamma_C} \overline{\lambda_{ht}} v_{ht}\, d\Gamma = L(\boldsymbol{v}_h) \qquad \forall \boldsymbol{v}_h \in \boldsymbol{V}_h. \tag{3.4}$$

Choosing $\boldsymbol{v}_h = \overline{\boldsymbol{u}_h} - \boldsymbol{u}_h$ in (3.3) and $\boldsymbol{v}_h = \boldsymbol{u}_h - \overline{\boldsymbol{u}_h}$ in (3.4) implies by addition that

$$\begin{aligned} & a(\boldsymbol{u}_h - \overline{\boldsymbol{u}_h}, \boldsymbol{u}_h - \overline{\boldsymbol{u}_h}) \\ & = \int_{\Gamma_C} (\lambda_{hn} - \overline{\lambda_{hn}})(u_{hn} - \overline{u_{hn}})\, d\Gamma + \int_{\Gamma_C} (\lambda_{ht} - \overline{\lambda_{ht}})(u_{ht} - \overline{u_{ht}})\, d\Gamma. \end{aligned} \tag{3.5}$$

Let us notice that the inequality in (3.1) is obviously equivalent to the two following conditions:

$$\int_{\Gamma_C} (\nu_{hn} - \lambda_{hn}) u_{hn}\, d\Gamma \geq 0 \qquad \forall \nu_{hn} \in M_{hn}, \tag{3.6}$$

$$\int_{\Gamma_C} (\nu_{ht} - \lambda_{ht}) u_{ht}\, d\Gamma \geq 0 \qquad \forall \nu_{ht} \in M_{ht}(-\mu g_{hn}). \tag{3.7}$$

According to the definitions of M'_{hn} and M''_{hn}, we can choose $\nu_{hn} = 0$ and $\nu_{hn} = 2\lambda_{hn}$ in (3.6), which gives

$$\int_{\Gamma_C} \lambda_{hn} u_{hn}\, d\Gamma = 0 \quad \text{ and } \quad \int_{\Gamma_C} \nu_{hn} u_{hn}\, d\Gamma \geq 0 \qquad \forall \nu_{hn} \in M_{hn},$$

from which we deduce that

$$\int_{\Gamma_C} (\lambda_{hn} - \overline{\lambda_{hn}})(u_{hn} - \overline{u_{hn}})\, d\Gamma \leq 0.$$

Denoting by α the ellipticity constant of the bilinear form $a(.,.)$, (3.5) becomes

$$\alpha \|\boldsymbol{u}_h - \overline{\boldsymbol{u}_h}\|_1^2 \leq \int_{\Gamma_C} (\lambda_{ht} - \overline{\lambda_{ht}})(u_{ht} - \overline{u_{ht}})\, d\Gamma. \tag{3.8}$$

To evaluate the latter integral term, let us first introduce the p-by-p mass matrix $\mathcal{M} = (m_{ij})_{1 \leq i,j \leq p}$ on Γ_C as

$$m_{ij} = \int_{\Gamma_C} \psi_i \psi_j\, d\Gamma, \qquad 1 \leq i,j \leq p, \tag{3.9}$$

and let U_T, $\overline{U_T}$, G_N, $\overline{G_N}$, denote the vectors of components the nodal values of u_{ht}, $\overline{u_{ht}}$, g_{hn}, and $\overline{g_{hn}}$, respectively.

• We begin with considering the mixed method, where $\boldsymbol{M}_h(g) = \boldsymbol{M}'_h(g)$. From (3.7), we get

$$\int_{\Gamma_C} \lambda_{ht} u_{ht}\, d\Gamma \leq \int_{\Gamma_C} \nu_{ht} u_{ht}\, d\Gamma \quad \forall \nu_{ht} \in M_{ht}(-\mu g_{hn}),$$

or, equivalently,

$$\int_{\Gamma_C} \lambda_{ht} u_{ht}\, d\Gamma \leq \sum_{i=1}^{p} M_i (\mathcal{M} U_T)_i \quad \forall M \in \mathbb{R}^p \text{ such that } |M_i| \leq -\mu (G_N)_i,\ 1 \leq i \leq p.$$

It is easy to construct a vector M minimizing the sum and yielding the following bound:

$$\int_{\Gamma_C} \lambda_{ht} u_{ht}\, d\Gamma \leq \mu \sum_{i=1}^{p} (G_N)_i |(\mathcal{M} U_T)_i|.$$

A similar expression can be obtained when integrating the term $\overline{\lambda_{ht}u_{ht}}$. The two remaining terms of the integral in (3.8) are roughly bounded as follows:

$$-\int_{\Gamma_C} \lambda_{ht}\overline{u_{ht}}\, d\Gamma \le -\mu\sum_{i=1}^{p}(G_N)_i|(\mathcal{M}\overline{U_T})_i|;\ -\int_{\Gamma_C} \overline{\lambda_{ht}}u_{ht}\, d\Gamma \le -\mu\sum_{i=1}^{p}(\overline{G_N})_i|(\mathcal{M}U_T)_i|.$$

Finally, (3.8) becomes

$$\begin{aligned}\alpha\|\boldsymbol{u}_h - \overline{\boldsymbol{u}_h}\|_1^2 &\le \mu\sum_{i=1}^{p}(G_N - \overline{G_N})_i\left(|(\mathcal{M}U_T)_i| - |(\mathcal{M}\overline{U_T})_i|\right)\\ &\le \mu\left(\sum_{i=1}^{p}(G_N - \overline{G_N})_i^2\right)^{\frac{1}{2}}\left(\sum_{i=1}^{p}(\mathcal{M}(U_T - \overline{U_T}))_i^2\right)^{\frac{1}{2}}\\ &= \mu\|G_N - \overline{G_N}\|_{\mathbb{R}^p}\ \|U_T - \overline{U_T}\|_{\mathbb{R}^p,\mathcal{M}}, \end{aligned} \tag{3.10}$$

where $\big||x| - |y|\big| \le |x - y|$ and the Hölder inequality have been used. The notations $\|.\|_{\mathbb{R}^p}$ and $\|.\|_{\mathbb{R}^p,\mathcal{M}}$ whose definitions are straightforward represent norms on $\mathbb{R}^p$. (The mass matrix $\mathcal{M}$ is nonsingular.) As a consequence, there exist constants $C_1(h)$ and $C_2(h)$ depending on h (or equivalently on p) such that

$$\|G_N - \overline{G_N}\|_{\mathbb{R}^p} \le C_1(h)\|g_{hn} - \overline{g_{hn}}\|_{-\frac{1}{2},h} \tag{3.11}$$

and

$$\|U_T - \overline{U_T}\|_{\mathbb{R}^p,\mathcal{M}} \le C_2(h)\|u_{ht} - \overline{u_{ht}}\|_{L^2(\Gamma_C)} \le C_3(h)\|\boldsymbol{u}_h - \overline{\boldsymbol{u}_h}\|_1, \tag{3.12}$$

where the trace theorem has been used. Combining (3.10), (3.11), (3.12), and (3.2) implies that there exists a constant $C(h)$ such that

$$\|\lambda_{hn} - \overline{\lambda_{hn}}\|_{-\frac{1}{2},h} \le \mu C(h)\|g_{hn} - \overline{g_{hn}}\|_{-\frac{1}{2},h}. \tag{3.13}$$

Hence Φ_h is continuous.

- In the case where $\boldsymbol{M}_h(g) = \boldsymbol{M}''_h(g)$, the proof is analogous: first (3.7) implies

$$\int_{\Gamma_C} \lambda_{ht}u_{ht}\, d\Gamma \le \sum_{i=1}^{p}(\mathcal{M}M)_i(U_T)_i \ \ \forall M \in \mathbb{R}^p \text{ such that}$$
$$|(\mathcal{M}M)_i| \le -\mu(\mathcal{M}G_N)_i, \quad 1 \le i \le p.$$

Therefore,

$$\int_{\Gamma_C} \lambda_{ht}u_{ht}\, d\Gamma \le \mu\sum_{i=1}^{p}(\mathcal{M}G_N)_i|(U_T)_i|.$$

Expression (3.8) then leads to

$$\begin{aligned}\alpha\|\boldsymbol{u}_h - \overline{\boldsymbol{u}_h}\|_1^2 &\le \mu\sum_{i=1}^{p}(\mathcal{M}(G_N - \overline{G_N}))_i\left(|(U_T)_i| - |(\overline{U_T})_i|\right)\\ &\le \mu\left(\sum_{i=1}^{p}(\mathcal{M}(G_N - \overline{G_N}))_i^2\right)^{\frac{1}{2}}\left(\sum_{i=1}^{p}(U_T - \overline{U_T})_i^2\right)^{\frac{1}{2}}\\ &= \mu\|G_N - \overline{G_N}\|_{\mathbb{R}^p,\mathcal{M}}\ \|U_T - \overline{U_T}\|_{\mathbb{R}^p},\end{aligned}$$

and the continuity of Φ_h is proved following the same arguments as in the first case.

Step 2. Let $(\boldsymbol{u}_h, \lambda_{hn}, \lambda_{ht})$ be the solution of $(P(g_{hn}))$. Taking $\boldsymbol{v}_h = \boldsymbol{u}_h$ in (3.1) gives

$$a(\boldsymbol{u}_h, \boldsymbol{u}_h) - \int_{\Gamma_C} \lambda_{hn} u_{hn} \, d\Gamma - \int_{\Gamma_C} \lambda_{ht} u_{ht} \, d\Gamma = L(\boldsymbol{u}_h). \tag{3.14}$$

According to

$$\int_{\Gamma_C} \lambda_{hn} u_{hn} \, d\Gamma = 0 \quad \text{and} \quad \int_{\Gamma_C} \lambda_{ht} u_{ht} \, d\Gamma \leq 0,$$

we deduce from (3.14), the $\boldsymbol{V}$-ellipticity of $a(.,.)$, and the continuity of $L(.)$ that

$$\alpha \|\boldsymbol{u}_h\|_1^2 \leq a(\boldsymbol{u}_h, \boldsymbol{u}_h) \leq L(\boldsymbol{u}_h) \leq C \|\boldsymbol{u}_h\|_1,$$

where the constant C depends on the loads $\boldsymbol{f}$ and $\boldsymbol{F}$. Therefore, we get

$$\|\boldsymbol{u}_h\|_1 \leq \frac{C}{\alpha}.$$

In other respects

$$a(\boldsymbol{u}_h, \boldsymbol{v}_h) - \int_{\Gamma_C} \lambda_{hn} v_{hn} \, d\Gamma = L(\boldsymbol{v}_h) \qquad \forall \boldsymbol{v}_h \in \tilde{\boldsymbol{V}}_h,$$

leads to

$$\int_{\Gamma_C} \lambda_{hn} v_{hn} \, d\Gamma \leq M \|\boldsymbol{u}_h\|_1 \|\boldsymbol{v}_h\|_1 + C \|\boldsymbol{v}_h\|_1 \qquad \forall \boldsymbol{v}_h \in \tilde{\boldsymbol{V}}_h.$$

That implies

$$\|\lambda_{hn}\|_{-\frac{1}{2},h} \leq M \|\boldsymbol{u}_h\|_1 + C \leq \left(\frac{M}{\alpha} + 1\right) C.$$

Finally,

$$\|\Phi_h(g_{hn})\|_{-\frac{1}{2},h} \leq C' \qquad \forall g_{hn} \in M_{hn},$$

where C' depends only on the applied loads $\boldsymbol{f}, \boldsymbol{F}$ and on the continuity and ellipticity constant of $a(.,.)$. This together with the continuity of Φ_h proves that there exists at least a solution of Coulomb's discrete frictional contact problem according to Brouwer's fixed point theorem. □

Remark 3.3. From (3.13) when $\boldsymbol{M}_h(g) = \boldsymbol{M}'_h(g)$ or from the equivalent bound which is obtained when $\boldsymbol{M}_h(g) = \boldsymbol{M}''_h(g)$, we get a (quite weak) uniqueness result when $\mu \, C(h) \leq 1$. That means that uniqueness holds when μ is small enough, where the denomination "small" depends on the discretization parameter. A more detailed study would show that we are not able to prove that $C(h)$ remains bounded as h tends towards 0. Using another mixed finite element formulation (with a single multiplier instead of two which is not adapted to our a posteriori error estimator) leads to similar existence and uniqueness results (see [10, 11]).

3.2. The matrix formulation of the frictional contact conditions. Let us consider a solution $(\boldsymbol{u}_h, \lambda_{hn}, \lambda_{ht}) \in \boldsymbol{V}_h \times M_{hn} \times M_{ht}(-\mu\lambda_{hn})$ of Coulomb's discrete frictional contact problem. We are interested in the matrix translation of the frictional contact conditions

$$\int_{\Gamma_C} (\nu_{hn} - \lambda_{hn}) u_{hn} \, d\Gamma \geq 0 \qquad \forall \nu_{hn} \in M_{hn}, \tag{3.15}$$

$$\int_{\Gamma_C} (\nu_{ht} - \lambda_{ht}) u_{ht} \, d\Gamma \geq 0 \qquad \forall \nu_{ht} \in M_{ht}(-\mu\lambda_{hn}). \tag{3.16}$$

As before, Γ_C contains p nodes of the triangulation and ψ_i, $1 \leq i \leq p$, denote the scalar monodimensional basis functions on Γ_C. The p-by-p mass matrix $\mathcal{M} = (m_{ij})_{1\leq i,j\leq p}$ on Γ_C is given by (3.9).

Let U_N and U_T denote the vectors of components the nodal values of u_{hn} and u_{ht}, respectively, and let L_N and L_T denote the vectors of components the nodal values of λ_{hn} and λ_{ht}, respectively. We begin with considering the mixed method, where $\boldsymbol{M}_h(g) = \boldsymbol{M}'_h(g)$.

PROPOSITION 3.4. *Let $\boldsymbol{M}_h(g) = \boldsymbol{M}'_h(g)$. The vectors U_N, U_T, L_N, L_T associated with a solution of Coulomb's discrete frictional contact problem satisfy, for any $1 \leq i \leq p$,*

$$(L_N)_i \leq 0, \tag{3.17}$$

$$(\mathcal{M}U_N)_i \leq 0, \tag{3.18}$$

$$(L_N)_i(\mathcal{M}U_N)_i = 0, \tag{3.19}$$

$$|(L_T)_i| \leq -\mu(L_N)_i, \tag{3.20}$$

$$|(L_T)_i| < -\mu(L_N)_i \implies (\mathcal{M}U_T)_i = 0, \tag{3.21}$$

$$(L_T)_i(\mathcal{M}U_T)_i \leq 0. \tag{3.22}$$

Proof. From $\lambda_{hn} \in M'_{hn}$, we immediately get (3.17). Condition (3.15) is equivalent to

$$\int_{\Gamma_C} \nu_{hn} u_{hn} \, d\Gamma \geq 0 \quad \forall \nu_{hn} \in M'_{hn} \qquad \text{and} \qquad \int_{\Gamma_C} \lambda_{hn} u_{hn} \, d\Gamma = 0. \tag{3.23}$$

Choosing in the inequality of (3.23), $\nu_{hn} = -\psi_i$, $1 \leq i \leq p$, and writing $u_{hn} = \sum_{j=1}^{p} (U_N)_j \, \psi_j$ gives (3.18). Putting $\lambda_{hn} = \sum_{i=1}^{p} (L_N)_i \, \psi_i$ and $u_{hn} = \sum_{j=1}^{p} (U_N)_j \, \psi_j$ in the equality of (3.23) yields

$$\sum_{i=1}^{p} (L_N)_i (\mathcal{M}U_N)_i = 0.$$

The latter estimate together with (3.17) and (3.18) implies (3.19).

Inequality (3.20) follows directly from $\lambda_{ht} \in M'_{ht}(-\mu\lambda_{hn})$. For any $1 \leq i \leq p$, choose ν_{ht} in (3.16) as follows: $\nu_{ht} = \mu\lambda_{hn}$ at node i and $\nu_{ht} = \lambda_{ht}$ at the $p-1$ other nodes. We obtain

$$\begin{aligned} \int_{\Gamma_C} (\nu_{ht} - \lambda_{ht}) u_{ht} \, d\Gamma &= (\mu L_N - L_T)_i \int_{\Gamma_C} \psi_i u_{ht} \, d\Gamma \\ &= (\mu L_N - L_T)_i (\mathcal{M}U_T)_i \geq 0. \end{aligned} \tag{3.24}$$

Similarly, take $\nu_{ht} = -\mu\lambda_{hn}$ at node i and $\nu_{ht} = \lambda_{ht}$ at the $p-1$ other nodes. We get

$$\int_{\Gamma_C} (\nu_{ht} - \lambda_{ht})u_{ht}\, d\Gamma = (-\mu L_N - L_T)_i(\mathcal{M}U_T)_i \geq 0. \tag{3.25}$$

Putting together estimates (3.24) and (3.25) implies (3.21).

It remains to prove (3.22). Define ν_{ht} in (3.16) as follows: $\nu_{ht} = \frac{1}{2}\lambda_{ht}$ at node i and $\nu_{ht} = \lambda_{ht}$ at the $p-1$ other nodes. Therefore

$$\int_{\Gamma_C} (\nu_{ht} - \lambda_{ht})u_{ht}\, d\Gamma = -\frac{1}{2}(L_T)_i \int_{\Gamma_C} \psi_i u_{ht}\, d\Gamma = -\frac{1}{2}(L_T)_i(\mathcal{M}U_T)_i \geq 0.$$

Hence inequality (3.22) is proved. □

Proceeding in a similar way when $\boldsymbol{M}_h(g) = \boldsymbol{M}''_h(g)$, we obtain the following proposition.

PROPOSITION 3.5. *Let $\boldsymbol{M}_h(g) = \boldsymbol{M}''_h(g)$. The vectors U_N, U_T, L_N, L_T associated with a solution of Coulomb's discrete frictional contact problem satisfy, for any $1 \leq i \leq p$,*

$$\begin{aligned}
(\mathcal{M}L_N)_i &\leq 0,\\
(U_N)_i &\leq 0,\\
(\mathcal{M}L_N)_i(U_N)_i &= 0,\\
|(\mathcal{M}L_T)_i| &\leq -\mu(\mathcal{M}L_N)_i,\\
|(\mathcal{M}L_T)_i| &< -\mu(\mathcal{M}L_N)_i \implies (U_T)_i = 0,\\
(\mathcal{M}L_T)_i(U_T)_i &\leq 0.
\end{aligned}$$

Remark 3.6. We show in the next section that the choice of the method using $\boldsymbol{M}'_h(g)$ is quite appropriate and easier than $\boldsymbol{M}''_h(g)$ to compute the estimator. However, most of the finite element codes solving contact problems (with or without friction) make use of nodal displacements U_N, U_T and of nodal forces F_N, F_T as dual unknowns (and not pressures like L_N, L_T) on the contact part Γ_C. This means that the frictional contact conditions are generally $(F_N)_i \leq 0$, $(U_N)_i \leq 0$, $(F_N)_i(U_N)_i = 0$, $|(F_T)_i| \leq -\mu(F_N)_i$, and so on. This is precisely the choice of $\boldsymbol{M}''_h(g)$ when supposing that pressures and forces are linked by $F_N = \mathcal{M}L_N$ and $F_T = \mathcal{M}L_T$. As a consequence, we must also be able to propose a practical computation of the estimator for this widespread case.

4. Construction of admissible fields.

4. Construction of admissible fields. The purpose of this section is to describe the building of admissible fields $\hat{\boldsymbol{u}}, \hat{\boldsymbol{w}}, \hat{\boldsymbol{\sigma}}, \hat{\boldsymbol{r}}$ satisfying the kinematic conditions (2.1) and the equilibrium equations (2.2) to compute the error estimator (2.11). Moreover, in order to obtain a finite value of the error estimator, the displacement fields $\hat{\boldsymbol{w}}$ on the contact part Γ_C must satisfy the nonpenetration conditions, and the densities of forces $\hat{\boldsymbol{r}}$ should belong to Coulomb's friction cone C_μ on Γ_C.

To perform such a construction, we will obviously make use of the finite element solution of Coulomb's frictional contact problem $(\boldsymbol{u}_h, \lambda_{hn}, \lambda_{ht}) \in \boldsymbol{V}_h \times M_{hn} \times M_{ht}(-\mu\lambda_{hn}) = \boldsymbol{V}_h \times \boldsymbol{M}_h(-\mu\lambda_{hn})$ which satisfies

$$\begin{cases}
a(\boldsymbol{u}_h, \boldsymbol{v}_h) - \displaystyle\int_{\Gamma_C} \lambda_{hn} v_{hn}\, d\Gamma - \int_{\Gamma_C} \lambda_{ht} v_{ht}\, d\Gamma = L(\boldsymbol{v}_h) \quad \forall \boldsymbol{v}_h \in \boldsymbol{V}_h,\\
\displaystyle\int_{\Gamma_C} (\nu_{hn} - \lambda_{hn})u_{hn}\, d\Gamma + \int_{\Gamma_C} (\nu_{ht} - \lambda_{ht})u_{ht}\, d\Gamma \geq 0 \quad \forall (\nu_{hn}, \nu_{ht}) \in \boldsymbol{M}_h(-\mu\lambda_{hn}),
\end{cases}$$

where $\boldsymbol{M}_h(-\mu\lambda_{hn}) = M_{hn} \times M_{ht}(-\mu\lambda_{hn})$ denotes either $\boldsymbol{M}'_h(-\mu\lambda_{hn}) = M'_{hn} \times M'_{ht}(-\mu\lambda_{hn})$ or $\boldsymbol{M}''_h(-\mu\lambda_{hn}) = M''_{hn} \times M''_{ht}(-\mu\lambda_{hn})$.

We begin with building the displacement fields $\hat{\boldsymbol{u}}$ and $\hat{\boldsymbol{w}}$ satisfying the kinematic conditions (2.1) and the nonpenetration conditions.

4.1. Construction of the displacement fields. For both finite element approaches (i.e., $\boldsymbol{M}_h(g) = \boldsymbol{M}'_h(g)$ or $\boldsymbol{M}_h(g) = \boldsymbol{M}''_h(g)$) the finite element displacement field $\boldsymbol{u}_h$ verifies the embedding conditions in (2.1).

When $\boldsymbol{M}_h(g) = \boldsymbol{M}'_h(g)$, the finite element displacement field may not fulfill the nonpenetration conditions according to Proposition 3.4. At the nodes $\boldsymbol{x}_j$ which are not located on Γ_C, we set $\hat{\boldsymbol{u}}(\boldsymbol{x}_j) = \boldsymbol{u}_h(\boldsymbol{x}_j)$. At the nodes $\boldsymbol{x}_i$ lying on Γ_C, we set $\hat{u}_t(\boldsymbol{x}_j) = u_{ht}(\boldsymbol{x}_j)$ and $\hat{u}_n(\boldsymbol{x}_j) = \min(u_{hn}(\boldsymbol{x}_j), 0)$. Using these nodal values, the displacement field $\hat{\boldsymbol{u}}$ is then built in $\boldsymbol{V}_h$ (and $\hat{\boldsymbol{w}} = \hat{\boldsymbol{u}}$ on Γ_C).

When $\boldsymbol{M}_h(g) = \boldsymbol{M}''_h(g)$, the finite element displacement field also satisfies the nonpenetration conditions according to Proposition 3.5. In that case, we simply take $\hat{\boldsymbol{u}} = \boldsymbol{u}_h$ in Ω (and $\hat{\boldsymbol{w}} = \hat{\boldsymbol{u}}$ on Γ_C).

4.2. Construction of the stress fields. Let us describe the building of the stress fields $\hat{\boldsymbol{\sigma}}$ and $\hat{\boldsymbol{r}}$ verifying the equilibrium equations and $\hat{\boldsymbol{r}} \in C_\mu$ on Γ_C.

4.2.1. Building of $\hat{\boldsymbol{r}} \in \boldsymbol{C_\mu}$ on $\boldsymbol{\Gamma_C}$. When $\boldsymbol{M}_h(g) = \boldsymbol{M}'_h(g)$, the building is straightforward. According to Proposition 3.4, we can directly choose $\hat{\boldsymbol{r}} = \boldsymbol{\lambda}_h$ (i.e., $\hat{r}_n = \lambda_{hn}$ and $\hat{r}_t = \lambda_{ht}$).

When $\boldsymbol{M}_h(g) = \boldsymbol{M}''_h(g)$, the situation is more complicated. However, from Proposition 3.5, we are not ensured that the multipliers $\boldsymbol{\lambda}_h = (\lambda_{hn}, \lambda_{ht})$ always belong to C_μ on Γ_C (i.e., satisfy $\lambda_{hn} \leq 0$ and $|\lambda_{ht}| \leq -\mu\lambda_{hn}$) as in the previous case. Nevertheless, there is in the construction a freedom on the choice of the tangential components $\hat{r}_t$. Indeed, they can be modified edge by edge by adding a density with null resultant and moment. More precisely, when the computed multipliers satisfy $|\lambda_{ht}| > -\mu\lambda_{hn}$ at the node i, it is possible to compute a new density $\tilde{\lambda}_{ht}$ on the edge $[i, j]$ (j is one of the neighboring nodes) as follows:

$$\begin{aligned} \tilde{\lambda}_{ht} &= \lambda_{ht} - d \text{ at node } i, \\ \tilde{\lambda}_{ht} &= \lambda_{ht} + d \text{ at node } j. \end{aligned} \tag{4.1}$$

If $d \in \mathbb{R}$ is chosen such that $|\tilde{\lambda}_{ht}| \leq -\mu\lambda_{hn}$ at nodes i and j, then the modification is satisfying. In such a case, the modified tangential pressure $\tilde{\lambda}_{ht}$ is piecewise linear on each mesh and possibly discontinuous on Γ_C. We finally choose $\hat{r}_n = \lambda_{hn}$ and $\hat{r}_t = \tilde{\lambda}_{ht}$.

4.2.2. Building of $\hat{\boldsymbol{\sigma}}$ verifying the equilibrium equations. Next, having at our disposal $\hat{\boldsymbol{r}}$, the stress fields $\hat{\boldsymbol{\sigma}}$ satisfying (2.2) are to be constructed. It is straightforward that the stress field obtained from the finite element displacement field $\boldsymbol{u}_h$ with the constitutive relation $\boldsymbol{\sigma}_h = \mathcal{C}\,\boldsymbol{\varepsilon}(\boldsymbol{u}_h)$ does not satisfy the equilibrium equation (2.2). If we want to compute the error estimator, a stress field $\hat{\boldsymbol{\sigma}}$ that strictly satisfies the equilibrium equations must be obtained. The construction of $\hat{\boldsymbol{\sigma}}$ is performed in two steps which can be summarized as follows:

- The first step consists of building densities of forces $\hat{\boldsymbol{F}}$ on each edge of the mesh satisfying equilibrium with the body forces $\boldsymbol{f}$:

$$\int_E \boldsymbol{f}.\boldsymbol{v}\, d\Omega + \int_{\partial E} \eta_E \hat{\boldsymbol{F}}.\boldsymbol{v}\, d\Gamma = 0 \qquad \forall \boldsymbol{v} \text{ such that } \boldsymbol{\varepsilon}(\boldsymbol{v}) = 0,$$

where η_E is a function defined on the boundary of the triangular element E, constant on each edge of E, equal to 1 or equal to -1 and satisfying $\eta_E+\eta_{E'}=0$ on the common edge to adjacent elements E and E'. Note that the construction of $\hat{\boldsymbol{F}}$ is always possible and it is generally not unique. In the numerical experiments, the nonuniqueness of $\hat{\boldsymbol{F}}$ is handled in minimizing (locally, on each patch of elements connected to a node) the difference (least squares) with the finite element solution. The choice of such a technique very often leads to satisfactory effectivity indexes (between 1 and 1.5; see 5.3.1 hereafter). The details of these techniques can be found in [18].

• The second step is devoted to the construction of $\hat{\boldsymbol{\sigma}}$ locally on each element E by solving

$$\left\{\begin{array}{lll} \mathbf{div}\,\hat{\boldsymbol{\sigma}}+\boldsymbol{f} & =0 & \text{in } E,\\ \hat{\boldsymbol{\sigma}}\boldsymbol{n} & =\eta_E\hat{\boldsymbol{F}} & \text{on } \partial E,\end{array}\right.$$

where $\boldsymbol{n}$ stands for the unit outward normal on ∂E.

There are two techniques to compute locally the stress admissible field from the densities:

1. analytical construction; it is easy to check that there does not exist an $\hat{\boldsymbol{\sigma}}$ linear on E due to the stress symmetry requirement. The chosen technique for determining $\hat{\boldsymbol{\sigma}}$ on each triangle E is then to divide E into three subtriangles and to search $\hat{\boldsymbol{\sigma}}$ which is linear on each subtriangle. The details of this construction can be found in [18];
2. numerical construction by using higher-degree polynomials (see [4]).

5. Numerical studies.

5.1. Mesh adaption. The aim of adaptive procedures is to offer the user a level of accuracy denoted ϵ_0 with a minimal computational cost. We use the h-version which is the most widespread procedure of adaptivity currently in use: the size and the topology of the elements are modified but the same kind of basis functions for the different meshes are retained. A mesh T^* is said to be optimal with respect to a measure of the error ϵ^* if (see [16])

$$\left\{\begin{array}{l}\epsilon^*=\epsilon_0,\\ N^*\text{minimal } (N^*\text{: number of elements of } T^*).\end{array}\right. \tag{5.1}$$

To solve problem (5.1), the following procedure is applied:

1. An initial analysis is performed on a relatively uniform and coarse mesh T.
2. The corresponding global error ϵ in (2.12) and the local contributions ϵ_E in (2.13) are computed.
3. The characteristics of the optimal mesh T^* are determined in order to minimize the computational costs in respect of the global error.
4. A second finite element analysis is performed on the mesh T^*.

The optimal mesh T^* is determined by the computation of a size modification coefficient r_E on each element E of the mesh T:

$$r_E=\frac{h_E^*}{h_E},$$

where h_E denotes the size of E and h_E^* represents the size that must be imposed to the elements of T^* in the region of E in order to ensure optimality. The computation of the coefficients r_E uses the rate of convergence of the error which depends on the used element but also on the regularity of the solution [6]. Therefore, to compute the coefficients r_E, we use a technique detailed in [7] that automatically takes into account the steep gradient regions. The mesh T^* is generated by an automatic mesher able to accurately respect a map of sizes. Practically, the previous procedure allows us to divide in two or three the error ϵ. If the user wishes more accuracy, then the procedure is repeated as far as a precision close to ϵ_0 is reached (see [6]).

5.2. Examples. We consider two-dimensional plane strain problems where no body forces are applied. As a constitutive relation in Ω, we choose Hooke's law of homogeneous isotropic elastic materials:

$$\boldsymbol{\sigma}_{ij} = \frac{E\nu}{(1-2\nu)(1+\nu)}\delta_{ij}\boldsymbol{\varepsilon}_{kk}(\boldsymbol{u}) + \frac{E}{1+\nu}\boldsymbol{\varepsilon}_{ij}(\boldsymbol{u}),$$

where E and ν denote Young's modulus and Poisson's ratio, respectively, and the notation δ_{ij} stands for Kronecker's symbol. The implementation is achieved using CASTEM 2000 developed at the CEA, and an HP-C3000 computer has been used.

Three examples are studied with various friction coefficients. The computations have been carried out by using the mixed finite element method with $\boldsymbol{M}_h(g) = \boldsymbol{M}''_h(g)$. (See Remark 3.6 for comments.)

5.3. First example. We consider the problem depicted in Figure 2. The dimensions of the rectangular body are 40mm × 160mm, and computations are performed on the left half of the structure due to symmetry. The material characteristics are $E = 13$Gpa, $\nu = 0.2$, and $\mu = 0.5$ is the friction coefficient. The load on the left side is 10N.mm^{-2} and the upper side is clamped.

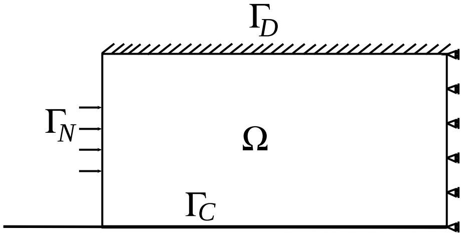


FIG. 2. *Setting of the problem.*

The initial mesh comprises 1088 three-node elements and 627 nodes for an accuracy ϵ of 9.74% (Figure 3). We show the deformed body in which separation occurs in Figure 4 and the contributions to the error ϵ_E in Figure 5. The contact pressures are reported in Figure 6. We show the normal pressure $-\lambda_{hn}$, the friction cone (between $-|\lambda_{hn}|$ and $|\lambda_{hn}|$), the tangential pressure $-\lambda_{ht}$, and the modified tangential pressure $-\tilde{\lambda}_{ht}$.

Due to the separation on the left part of Γ_C and the choice of the finite element method $\boldsymbol{M}_h(g) = \boldsymbol{M}''_h(g)$, we see that the normal pressure does not always satisfy the convenient sign property. Moreover, at the last mesh on the right part, the tangential pressure is outside the friction cone. By using the modification of the densities (4.1), we are able to compute a modified tangential pressure inside the cone which allows the computing of the statically admissible field. Concerning the normal pressure, the difficulty cannot be solved by the densities modification. The proposed mixed finite element method $\boldsymbol{M}_h(g) = \boldsymbol{M}'_h(g)$ will allow us to solve this difficulty.

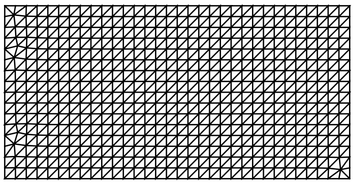


FIG. 3. *Initial mesh:* 1088 *three-node elements,* 627 *nodes,* $\epsilon = 9.74\%$.

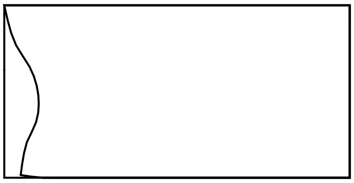


FIG. 4. *Deformed configuration.*

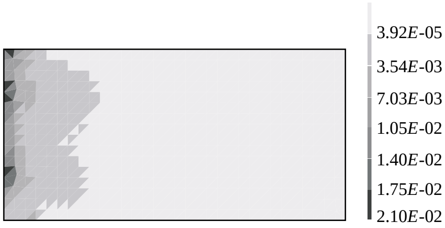


FIG. 5. *Local contributions* ϵ_E.

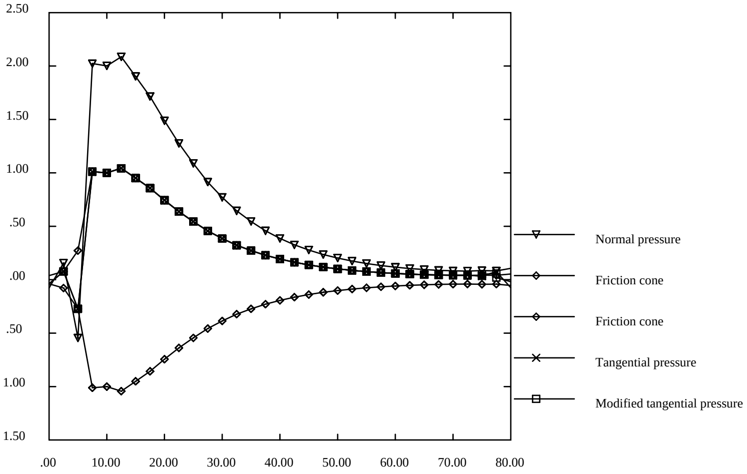


FIG. 6. *Contact pressures.*

The prescribed accuracy ϵ_0 is 5%. The optimized mesh is obtained in one step and comprises 1148 three-node elements and 647 nodes for an accuracy ϵ of 4.28% (Figure 7).

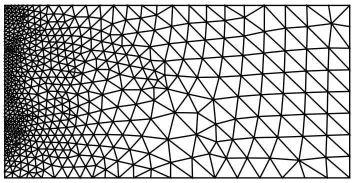


FIG. 7. *Optimized mesh:* 1148 *three-node elements,* 647 *nodes,* $\epsilon = 4.28\%$.

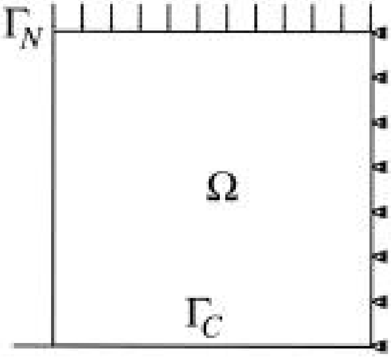


FIG. 8. *Setting of the problem.*

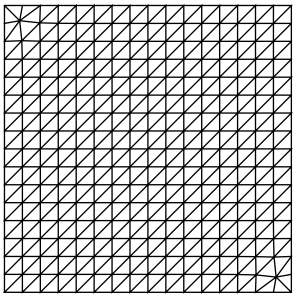


FIG. 9. *Initial mesh:* 544 *three-node elements,* 323 *nodes,* $\epsilon = 1.46\%$.

5.3.1. Second example. Next, we consider the structure depicted in Figure 8. The dimensions of the rectangle are 40mm × 80mm, and symmetry conditions are adopted. We choose $E = 13$Gpa, $\nu = 0.2$, and a friction coefficient of 0.3. The load on the upper side is 5N.mm^{-2}, and no embedding conditions are applied. In such a case, the bilinear form $a(.,.)$ is no longer $\boldsymbol{V}$-elliptic but satisfies some semicoercivity property (see [11, Theorem 6.3]).

The initial mesh is made of 544 three-node elements and 323 nodes corresponding to an accuracy ϵ of 1.46% (Figure 9). The deformed configuration and the map of local contributions ϵ_E are shown in Figures 10 and 11, respectively. In Figure 12, the normal contact pressure, the friction cone, and the tangential pressure are reported. We can notice that on the left node, the tangential pressure is outside the cone. By using the modification of the densities (4.1), we compute a modified tangential pressure which gives a new pressure inside the cone. Notice that in this example, we have stick on the contact zone, whereas the body was slipping in the previous

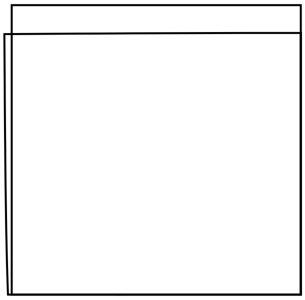


Fig. 10. *Deformed configuration.*

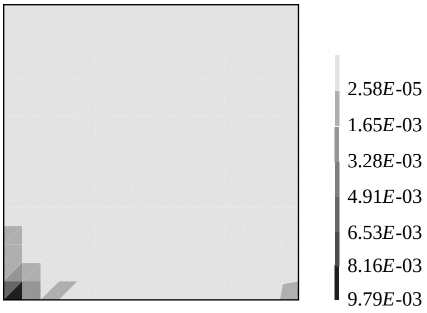


Fig. 11. *Local contributions ϵ_E.*

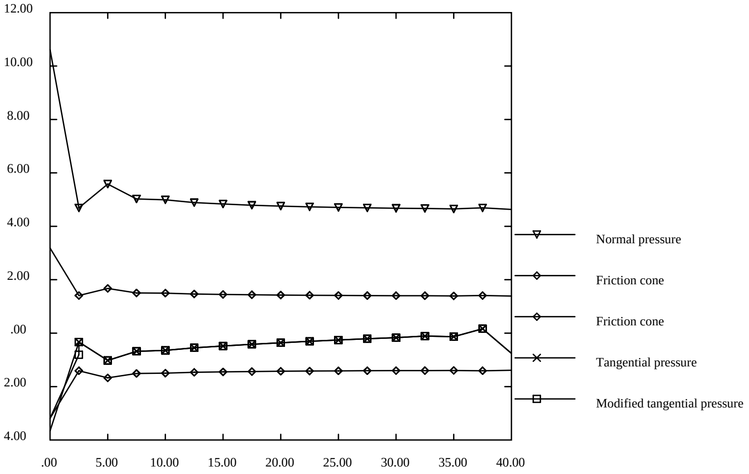


Fig. 12. *Contact pressures.*

example. (See also Proposition 3.5 for some corroboration.)

The prescribed accuracy ϵ_0 is 0.5%. The optimized mesh (obtained in one step) comprising 1178 three-node elements and 666 nodes for an accuracy ϵ of 0.57% is represented in Figure 13.

Next, we again consider the initial mesh in Figure 9, and we compute the effec-

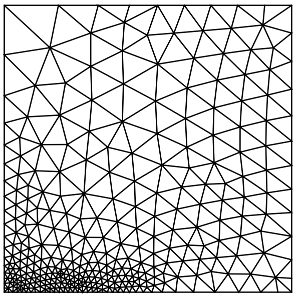


FIG. 13. *Optimized mesh:* 1178 *three-node elements,* 666 *nodes,* $\epsilon = 0.57\%$.

TABLE 1
Effectivity indexes.

μ	ϵ (in %)	ϵ_{ex} (in %)	effectivity γ
0.2	1.26	0.93	1.36
0.4	1.50	1.02	1.48
0.6	1.50	1.02	1.48
0.8	1.50	1.02	1.48

tivity indexes as a function of the friction coefficient μ. Since no analytical solution is available, we use a reference solution denoted $\boldsymbol{u}_{ref}$ corresponding to a very refined mesh. The exact error denoted ϵ_{ex} and the effectivity index γ can be defined as follows:

$$\epsilon_{ex} = \frac{\|\boldsymbol{u}_{ref} - \boldsymbol{u}_h\|_{u,\Omega}}{\|\boldsymbol{u}_{ref} + \boldsymbol{u}_h\|_{u,\Omega}}, \qquad \gamma = \frac{e}{\|\boldsymbol{u}_{ref} - \boldsymbol{u}_h\|_{u,\Omega}},$$

where e is the error estimator defined in (2.11). The results are reported in Table 1. Note that the frictionless case is not interesting because it corresponds to a pure compression case, and the error is negligible. The effectivity indexes close to 1 show the accuracy of the error estimator.

5.3.2. Third example: Numerical extension to two bodies in frictional contact. We consider the problem of two elastic bodies initially in contact (Figure 14). The upper body is submitted to a uniform load of 10N.mm^{-2}. We have adopted symmetry conditions on the lower side of Ω^2 in order to avoid a greater

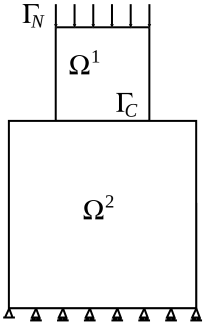


FIG. 14. *Setting of the problem.*

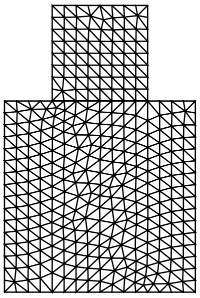


Fig. 15. *Initial mesh:* 640 *three-node elements,* 370 *nodes,* $\epsilon = 8.51\%$.

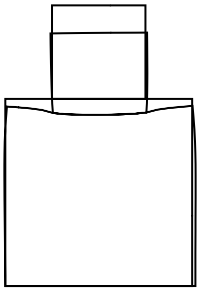


Fig. 16. *Deformed configuration.*

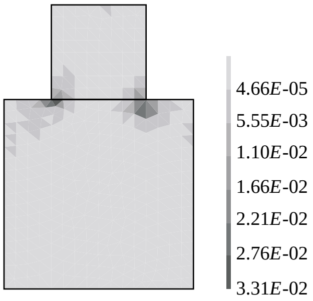


Fig. 17. *Local contributions* ϵ_E.

number of singularities, and the lower body is fixed on the left node of its lower side. The two materials are identical (E = 200GPa, $\nu = 0.25$), the dimensions are 100mm × 100mm and 200mm × 200mm, and the friction coefficient μ is 0.3.

The initial mesh with 640 three-node elements, 370 nodes, and an accuracy ϵ of 8.51% is shown in Figure 15. We show the deformed bodies (Figure 16) and the contributions to the error ϵ_E (Figure 17). The contact pressures are drawn in Figure 18. As in the previous example, the computed tangential pressure is outside the friction cone (on both extreme meshes), and as before, it can be successfully modified to compute the estimator.

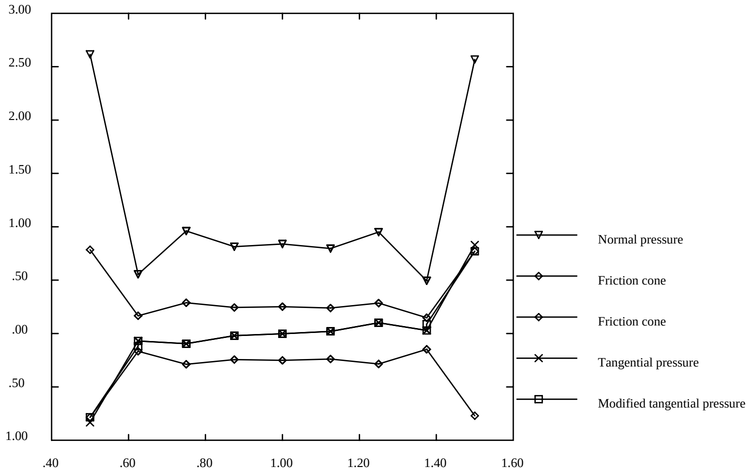


FIG. 18. *Contact pressures.*

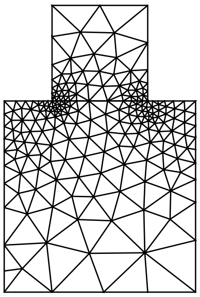


FIG. 19. *Optimized mesh:* 381 *three-node elements,* 230 *nodes,* $\epsilon = 5.46\%$.

An accuracy ϵ_0 of 5% is prescribed. The optimized mesh, obtained in one step, made of 381 three-node elements and 230 nodes for an accuracy ϵ of 5.46% is shown in Figure 19.

REFERENCES

[1] R. A. ADAMS, *Sobolev Spaces*, Academic Press, New York, 1975.

[2] I. BABUŠKA AND W. RHEINBOLDT, *Error estimates for adaptive finite element computations*, SIAM J. Numer. Anal., 15 (1978), pp. 736–754.

[3] C. CARSTENSEN, O. SCHERF, AND P. WRIGGERS, *Adaptive finite elements for elastic bodies in contact*, SIAM J. Sci. Comput., 20 (1999), pp. 1605–1626.

[4] P. COOREVITS, J.-P. DUMEAU, AND J.-P. PELLE, *Control of analyses with isoparametric elements in both 2D and 3D*, Internat. J. Numer. Methods Engrg., 46 (1999) pp. 157–176.

[5] P. COOREVITS, P. HILD, AND J.-P. PELLE, *A posteriori error estimation for unilateral contact with matching and nonmatching meshes*, Comput. Methods Appl. Mech. Engrg., 186 (2000), pp. 65–83.

[6] P. COOREVITS, P. LADEVÈZE, AND J.-P. PELLE, *An automatic procedure for finite element analysis in 2D elasticity*, Comput. Methods Appl. Mech. Engrg., 121 (1995), pp. 91–120.

[7] P. COOREVITS, P. LADEVÈZE, AND J.-P. PELLE, *Mesh optimization for problems with steep gradients*, Engrg. Comput., 11 (1994), pp. 129–144.

[8] G. DUVAUT AND J.-L. LIONS, *Les inéquations en mécanique et en physique*, Dunod, Paris, 1972.

[9] C. ECK AND J. JARUŠEK, *Existence results for the static contact problem with Coulomb friction*,

Math. Models Methods Appl. Sci., 8 (1998), pp. 445–468.

[10] J. Haslinger, *Approximation of the Signorini problem with friction, obeying the Coulomb law*, Math. Methods Appl. Sci., 5 (1983), pp. 422–437.

[11] J. Haslinger, I. Hlaváček, and J. Nečas, *Numerical methods for unilateral problems in solid mechanics*, in Handbook of Numerical Analysis, Vol. IV, P. G. Ciarlet and J. L. Lions, eds., North Holland, Amsterdam, 1996, pp. 313–485.

[12] J. Jarušek, *Contact problems with bounded friction. Coercive case*, Czechoslovak. Math. J., 33 (1983) pp. 237–261.

[13] N. Kikuchi and J. T. Oden, *Contact Problems in Elasticity: A Study of Variational Inequalities and Finite Element Methods*, SIAM, Philadelphia, 1988.

[14] P. Ladevèze, *Comparaison de modèles de milieux continus*, Ph.D. thesis, Université Pierre et Marie Curie, Paris 6, 1975.

[15] P. Ladevèze, *Mécanique non linéaire des structures*, Hermès, Paris, 1996.

[16] P. Ladevèze, G. Coffignal, and J.-P. Pelle, *Accuracy of elastoplastic and dynamic analysis*, in Accuracy Estimates and Adaptative Refinements in Finite Element Computations, Babuška, Gago, Oliveira, Zienkiewicz, eds., Wiley, Chichester, UK, 1986, pp. 181–203.

[17] P. Ladeveze and D. Leguillon, *Error estimate procedure in the finite element method and applications*, SIAM J. Numer. Anal., 20 (1983) pp. 485–509.

[18] P. Ladevèze, J.-P. Pelle, and P. Rougeot, *Error estimates and mesh optimization for FE computation*, Engrg. Comput., 8 (1991) pp. 69–80.

[19] C. Y. Lee and J. T. Oden, *A posteriori error estimation of $h-p$ finite element approximations of frictional contact problems*, Comput. Methods Appl. Mech. Engrg., 113 (1994), pp. 11–45.

[20] J. Nečas, J. Jarušek, and J. Haslinger, *On the solution of the variational inequality to the Signorini problem with small friction*, Boll. Un. Mat. Ital. B (5), 17 (1980), pp. 796–811.

[21] R. Verfürth, *A review of a posteriori error estimation techniques for elasticity problems*, Comput. Methods Appl. Mech. Engrg., 176 (1999), pp. 419–440.

[22] O. C. Zienkiewicz and J. Z. Zhu, *A simple error estimator and adaptive procedure for practical engineering analysis*, Internat. J. Numer. Methods Engrg., 24 (1987), pp. 337–357.

SIAM J. SCI. COMPUT.
Vol. 23, No. 3, pp. 1000–1026

THE RANDOM PROJECTION METHOD FOR STIFF DETONATION CAPTURING*

WEIZHU BAO† AND SHI JIN‡

Abstract. In this paper we present a simple and robust random projection method for under-resolved numerical simulation of stiff detonation waves in chemically reacting flows. This method is based on the random projection method proposed by the authors for general hyperbolic systems with stiff reaction terms [W. Bao and S. Jin, *J. Comput. Phys.*, 163 (2000), pp. 216–248], where the ignition temperature is randomized in a suitable domain. It is simplified using the equations of instantaneous reaction and then extended to handle the interactions of detonations. Extensive numerical experiments, including interaction of detonation waves, and in two dimensions, demonstrate the reliability and robustness of this novel method.

Key words. detonation, random projection method, reacting flow, shock capturing

AMS subject classifications. 80A32, 35L67, 65M60

PII. S1064827599364969

1. Introduction.

An inviscid, compressible, reacting flow is described by the reactive Euler equations

$$U_t + F(U)_x + G(U)_y = \frac{1}{\varepsilon}\Psi(U), \tag{1.1}$$

$$U = \begin{pmatrix} \rho \\ m \\ n \\ e \\ \rho z \end{pmatrix}, \quad F(U) = \begin{pmatrix} m \\ m^2/\rho + p \\ mn/\rho \\ m(e+p)/\rho \\ mz \end{pmatrix}, \quad G(U) = \begin{pmatrix} n \\ mn/\rho \\ n^2/\rho + p \\ n(e+p)/\rho \\ nz \end{pmatrix}, \tag{1.2}$$

$$\Psi(U) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ -\rho z e^{-T_c/T} \end{pmatrix} \equiv \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \psi(U) \end{pmatrix}. \tag{1.3}$$

The dependent variables $\rho(x,y,t)$, $m(x,y,t)$, $n(x,y,t)$, $e(x,y,t)$, and $z(x,y,t)$ are the density, x- and y-momentum, total energy, and the fraction of unburnt fluid, respectively. The pressure for ideal gas is given by

$$p = (\gamma - 1)\left(e - \frac{1}{2}\left(m^2 + n^2\right)/\rho - q_0\rho z\right),$$

and the temperature is defined as $T = p/\rho$. Let $(u,v) = (m/\rho, n/\rho)$ be the velocity. The parameters q_0, T_c, γ, and ε correspond to chemical heat release, ignition temperature, c_p to c_v ratio, and reaction time, respectively. The equations have been

*Received by the editors December 16, 1999; accepted for publication (in revised form) March 26, 2001; published electronically September 26, 2001.

http://www.siam.org/journals/sisc/23-3/36496.html

†Department of Computational Science, National University of Singapore, Singapore 117543 (bao@cz3.nus.edu.sg.)

‡Department of Mathematics, University of Wisconsin-Madison, Madison, WI 53706 (jin@math.wisc.edu). This author's research was supported in part by NSF grants DMS-9704957 and DMS-0196106.

nondimensionalized, leaving the choice of these four parameters to completely determine the problem.

The focus of this paper is on the computations of stiff detonation waves. For these waves the viscosity is not as important as for the slower deflagration wave solutions.

Equations (1.1)–(1.3) are usually referred to as the reactive Euler equations with Arrhenius kinetics. When the source term is replaced by

$$-\frac{1}{\varepsilon}\rho z H(T - T_c),$$

where $H(x) = 1$ for $x > 0$ and $H(x) = 0$ for $x < 0$, the kinetics is referred to as the *Heaviside kinetics* [21]. The difference in the kinetics affect the details of the detonation layers, which are of width $O(\varepsilon)$ with pressure and temperature spikes that decay exponentially into the postdetonation equilibria.

One of the main numerical challenges for reacting flows is that the kinetics equations (1.1) often include reactions with widely varying time scales. The chemical time scales, as characterized by ε, may be orders of magnitude faster than the fluid dynamical time scale. This leads to problems of severe numerical stiffness. Actually, the stiffness issue with the Heaviside kinetics is the more severe one [11]. Even a stable numerical scheme may lead to spurious unphysical solutions unless the small chemical time scale is fully resolved numerically.

Numerical methods for such problems have attracted a great deal of attention in the last decade. In particular, many works have contributed to the analysis and development of underresolved numerical methods which are capable of capturing the physically relevant macroscopic solutions without resolving the details of the denotation layers . Of course, when one does not resolve the chemical scale numerically (using grid size larger than the reaction zone $O(\varepsilon)$), it is impossible to capture the pressure and temperature spikes in the reaction zone. Thus the best one can hope for is to capture the speed of detonation, as well as other wave features associated with the fluid dynamics. It was first observed by Colella, Majda and Roytburd [9] that an underresolved numerical method, where ε is not resolved by suitably small time steps and grid sizes, leads to a spurious weak detonation wave that travels one grid per time step. Since then, lots of attention has been paid to study this peculiar numerical phenomenon (see [4], [6], [13], [18], [19]). It is known that numerical shock profile, an essential ingredient in all shock capturing methods, leads to premature chemical reactions once the smeared value of the temperature in the numerical detonation layer is above the ignition temperature. Various approaches have been suggested to fix this numerical problem. For example, in [11], a temperature extrapolation technique was proposed. In [5] the ignition temperature was artificially raised. In [20] the reaction time ε was replaced by a larger one, and thus the reaction zone was made much wider than the physical one. Recently, a modified fractional step method was introduced [15], where the structure of the Riemann solution of the homogeneous part was used to determine where burning should occur in each time step. This recipe works within the framework of the Godunov-type methods.

Recently, we proposed the random projection method as a general and systematic method to solve hyperbolic systems with stiff reaction term, applicable to reacting flow problems [1]. Unlike the random choice method of Chorin for reacting flow [7], which was originated from Glimm's scheme [12], and requires solving a generalized Riemann problem for hyperbolic systems with source terms [4], our method is a fractional step method, which combines a standard—no Riemann solver is needed—shock capturing method for the homogeneous convection with a strikingly simple random projection step for the reaction terms. In the random projection step, the ignition temperature is chosen to be a uniformly distributed random variable between the two stable equilibria. At each time step, this random projection will move the shock by at most one grid point. The statistical average, however, yields the correct speed, even though the small time scale ε is not numerically resolved. In particular, when the random number is chosen to be the equidistributed van der Corput sampling sequence [14], [8] we have proven, for a model scalar problem, a first order accuracy on the shock speed if a monotonicity-preserving method, which includes all TVD schemes, is used in the convection step [1], [2]. A large amount of numerical experiments for one- and two-dimensional detonation waves demonstrate the robustness of this novel approach.

The generality of the random projection method lies in the fact that it applies to *any* shock capturing method for the homogeneous part, while other approaches, such as the ideas of [7], [15], are restricted to Godunov-type methods that require Riemann or generalized Riemann solvers.

In this paper, we conduct extensive numerical experiments to examine the applicability of the random projection method for reacting flows. We focus on stiff detonation waves and their interactions with other waves, including the interaction of detonation waves. Since the aim is to test the validity of an underresolved numerical method, which completely ignores the details of the reaction zone but captures all the main features of the solution outside the reaction zone, it is adequate to formulate this method using the reacting flow model of instantaneous reaction (with zone-width reaction zone, the so-called Chapman–Jouguet (C-J) model [10]), as given by [7], in which the chemical heat is released instantaneously:

$$U_t + F(U)_x + G(U)_y = 0, \tag{1.4}$$

$$U = \begin{pmatrix} \rho \\ m \\ n \\ e \end{pmatrix}, \quad F(U) = \begin{pmatrix} m \\ m^2/\rho + p \\ mn/\rho \\ m(e+p)/\rho \end{pmatrix}, \quad G(U) = \begin{pmatrix} n \\ mn/\rho \\ n^2/\rho + p \\ n(e+p)/\rho \end{pmatrix}, \tag{1.5}$$

with equation of state

$$p = (\gamma - 1)\left(e - \frac{1}{2}\left(m^2 + n^2\right)/\rho - q_0\rho z\right), \tag{1.6}$$

and the fraction of unburnt gas

$$z = \begin{cases} 0 & \text{if } T > T_c, \\ 1 & \text{if } T < T_c. \end{cases} \tag{1.7}$$

Formally, when the reaction time goes to zero, (1.1)–(1.3) with Arrhenius or Heaviside kinetics reduce effectively to this model. Although this "zero reaction limit" is not rigorously justified mathematically, the numerical comparisons between the random projection method based on this model with the resolved calculations based on the original kinetics of Arrhenius or Heaviside, as carried out later in this paper, do support the validity of this reduction unless one wants the full details of the reaction layer.

There certainly are restrictions with the instantaneous reaction model, or more generally, with any underresolved numerical method. First, if one needs the details of the reaction layer, then one does need to numerically resolve the layer by solving the full equations (1.1)–(1.3) using fine meshes. Second, these methods cannot predict the instability in overdrive detonation waves, since, by ignoring the reaction layer, the peak value of pressure, which oscillates due to the instability, cannot be accurately computed. In order to obtain such fine structures, one has no choice but to use fine meshes, at least around the reaction zone by using techniques such as the adaptive mesh refinements.

The random projection method consists of two steps, the first being any standard shock capturing method for (1.4), followed by a random projection for the fraction variable z in (1.7), in which the ignition temperature T_c is replaced by a uniformly distributed random sequence. Here, the convection step is slightly simpler than the original one proposed in [1], where the full convection equation, including the homogeneous part of the species equation in (1.1)–(1.3), is solved. Algorithms for the collision of detonation waves are also introduced. Many numerical examples, including the C-J detonation, strong detonation, collisions of detonation with shocks, rarefaction wave, or another detonation, as well as two dimensional examples, will be used to justify the robustness of this novel approach.

The paper is organized as follows. In section 2 we provide a random projection method for the problem (1.4)–(1.7) in one space dimension with general initial data. Algorithms for multidetonations are also introduced. In section 3 this method is extended to two space dimension. In section 4 many numerical examples will be presented. In section 5 some conclusions are drawn.

2. One-dimensional detonations.

In this section, we shall describe the random projection method for (1.4)–(1.7) in one space dimension. Moreover, we will describe its implementation for the case of interaction of detonation waves. The problem to be solved is given by

$$U_t + F(U)_x = 0, \tag{2.1}$$

$$U = \begin{pmatrix} \rho \\ m \\ e \end{pmatrix}, \qquad F(U) = \begin{pmatrix} m \\ m^2/\rho + p \\ m(e+p)/\rho \end{pmatrix}, \tag{2.2}$$

with equation of state

$$p = (\gamma - 1)\left(e - \frac{1}{2}m^2/\rho - q_0\rho z\right), \tag{2.3}$$

and the fraction of unburnt gas

$$z = \begin{cases} 0 & \text{if } T > T_c, \\ 1 & \text{if } T < T_c. \end{cases} \tag{2.4}$$

Let the grid points be x_i, $i = \cdots, -1, 0, 1, \ldots$, with equal mesh spacing $h = x_{i+1} - x_i$. The time level $t_0 = 0$, t_1, t_2, $\ldots$ are also uniformly spaced with time step $k = t_{n+1} - t_n$. We use $U_i^n = (\rho_i^n, m_i^n, e_i^n, (\rho z)_i^n)$ to denote the approximate solution of $U = (\rho, m, e, \rho z)$ at the point $(x_i, t_n) = (ih, nk)$. Our main interest is an underresolved numerical method which allows $k = O(h) \gg \varepsilon$ and still obtains physically relevant numerical solutions.

The random projection method is a fractional step method that consists of a standard shock capturing method for (2.1), denoted by $S_F(k)$ for one time step, followed by a random projection step for the fraction variable z defined by (2.4) where T_c, the ignition temperature, is randomized in a suitable domain. Let $U^{n+1} = S_F(k)U^n$. To obtain z^{n+1}, we replace (2.4) by

$$z_j^{n+1} = \begin{cases} 0 & \text{if } T_j^{n+1} > \theta_n, \\ 1 & \text{if } T_j^{n+1} < \theta_n, \end{cases} \tag{2.5}$$

where $T_j^{n+1} = p_j^{n+1}/\rho_j^{n+1}$ and θ_n is a random number, chosen one per time step, between two equilibrium temperatures on both sides of the detonation. To be more precise, consider the initial data

$$(\rho(x,0), u(x,0), p(x,0), z(x,0)) = \begin{cases} (\rho_l(x), u_l(x), p_l(x), 0) & \text{if } x \le x_0, \\ (\rho_r(x), u_r(x), p_r(x), 1) & \text{if } x > x_0, \end{cases} \tag{2.6}$$

where x_0 is a given point. Without loss of generality these data are chosen such that the detonation, initially at $x = x_0$, moves to the right. The case when the detonation moved to the left can be treated similarly. Since our projection always makes z either 1 or 0, therefore, at any time step t_n, there is an $l(n) = j_0$, j_0 an integer, such that

$$z_j^n = \begin{cases} 0 & \text{if } j \le l(n), \\ 1 & \text{if } j > l(n). \end{cases} \tag{2.7}$$

Here $l(n)$ is the location of the jump for z in the approximate solution at time t_n and we assume $x_0 = l(0)h$ to be a grid point. Let

$$\theta_n = (T_l - T_r)\vartheta_n + T_r, \qquad T_l = \min_{x<x_0} \frac{p_l(x,0)}{\rho_l(x,0)}, \qquad T_r = \max_{x>x_0} \frac{p_r(x,0)}{\rho_r(x,0)} \tag{2.8}$$

with ϑ_n being the van der Corput sampling sequence on the interval $[0,1]$.

The van der Corput sequence is an equidistributed sequence with the minimal deviation among all random sequences [14]. It is obtained as follows: let $1 \le n = \sum_{k=0}^m i_k 2^k$, $i_k = 0, 1$, be the binary expansion of the integer n. One gets ϑ_n on $[0,1]$ as

$$\vartheta_n = \sum_{k=0}^m i_k 2^{-(k+1)}, \qquad n = 1, 2, \ldots . \tag{2.9}$$

Since there are other waves in the domain, one cannot project z according to (2.5) in the whole domain. Instead, we do it around the denotation, a procedure called the *local random projection* in [1]. Specifically, we move the jump of z according to the following algorithm:

```
(2.10)  $S_{sp}(k)$ :  set $l(n+1) := l(n) - 1$;
                For $l = l(n)-1, l(n), \ldots, l(n)+d$, do
                  $l(n+1) := l$   if $T_l^{n+1} > \theta_n$;
                $z_j^{n+1} = \begin{cases} 0 & \text{if } j \le l(n+1) \\ 1 & \text{if } j > l(n+1) \end{cases}$   for all $j$,
```

where d is the number of smeared points in the shock layer. In the above algorithm, only $d + 2$ points will be scanned.

The stability condition for this algorithm, as well as the algorithms for multidetonations (2.15) and (2.20), is the usual CFL condition determined by the operator $S_F(k)$ for the convection terms.

In our numerical comparison, we will compare the random projection method with the *deterministic* projection method which projects the fraction of unburnt gas, z, after the convection step according to the *fixed* ignition temperature T_c in (2.4).

We now extend the random projection method to handle the problems involving more than one detonation wave. For clarity of presentation we present only the case of two detonations. It is straightforward to extend to the case where there are more than two detonations.

Consider (2.1)–(2.4) with initial data

$$(\rho(x,0),u(x,0),p(x,0),z(x,0)) = \begin{cases} (\rho_l(x),u_l(x),p_l(x),0) & \text{if } x \le x_1, \\ (\rho_m(x),u_m(x),p_m(x),1) & \text{if } x_1 < x < x_2, \\ (\rho_r(x),u_r(x),p_r(x),0) & \text{if } x_2 \le x. \end{cases} \tag{2.11}$$

These data are chosen such that the two detonations move toward each other; i.e., the detonation initially at $x = x_1$ moves to the right and the one initially at $x = x_2$ moves to the left. Thus after some time, the two detonations will collide.

Let

$$T_l = \min_{x<x_1} \frac{p_l(x)}{\rho_l(x)}, \qquad T_m = \max_{x_1<x<x_2} \frac{p_m(x)}{\rho_m(x)}, \qquad T_r = \min_{x>x_2} \frac{p_r(x)}{\rho_r(x)} \tag{2.12}$$

and

$$\theta_n^{(1)} = (T_l - T_m)\vartheta_n + T_m, \qquad \theta_n^{(2)} = (T_r - T_m)\vartheta_n + T_m. \tag{2.13}$$

Since the projection always makes z either 1 or 0, the profile of z at any time step is a piecewise constant function. Therefore, at any time step t_n, there are $l_1(n) = j_1$ and $l_2(n) = j_2$ with $j_1 \le j_2$ integers such that

$$z_j^n = \begin{cases} 0 & \text{if } j \le l_1(n), \\ 1 & \text{if } l_1(n) < j < l_2(n), \\ 0 & \text{if } l_2(n) \le j. \end{cases} \tag{2.14}$$

Here we assume that $x_1 = l_1(0)h$ and $x_2 = l_2(0)h$ are grid points. Since $x_1 \le x_2$, then $l_1(0) \le l_2(0)$. One can use the following algorithm to obtain z^{n+1} if the positions of the two detonations at time t_n, i.e., $l_1(n)$ and $l_2(n)$, are known. The detailed algorithm to find z^{n+1} is as follows:

```
$S_{cp}(k)$ :  $l_{\text{mid}} = (l_1(n) + l_2(n))/2$;
               set $l_1(n+1) := l_1(n) - 1$;
               For $l = l_1(n) - 1, l_1(n), \ldots, \min\{l_1(n) + d,\ l_{\text{mid}} + 1\}$ do
                   $l_1(n+1) := l$  if $T_l^{n+1} > \theta_n^{(1)}$;
               set $l_2(n+1) := l_2(n) + 1$;
               For $l = l_2(n) + 1, l_2(n), \ldots, \max\{l_2(n) - d,\ l_{\text{mid}} - 1\}$ do
                   $l_2(n+1) := l$  if $T_l^{n+1} > \theta_n^{(2)}$;
```

$$z_j^{n+1} = \begin{cases} 0 & \text{if } j \le l_1(n+1) \\ 1 & \text{if } l_1(n+1) < j < l_2(n+1) \\ 0 & \text{if } l_2(n+1) \le j \end{cases} \qquad \text{for all } j. \tag{2.15}$$

This algorithm still works even after the detonations have collided and then become extinct. After the detonations have collided, the fraction of unburnt gas $z \equiv 0$. From this algorithm, once $l_1(n) \geq l_2(n)$ at some time step $t = t_n$, then $l_1(n+1) \geq l_2(n+1)$ for the next step. Thus the profile of z^{n+1} determined from (2.15) is the zero function.

Another case is when two detonations move away from each other. Consider the initial data

$$(\rho(x,0), u(x,0), p(x,0), z(x,0)) = \begin{cases} (\rho_l(x), u_l(x), p_l(x), 1) & \text{if } x < x_1, \\ (\rho_m(x), u_m(x), p_m(x), 0) & \text{if } x_1 \leq x \leq x_2, \\ (\rho_r(x), u_r(x), p_r(x), 1) & \text{if } x_2 < x. \end{cases} \tag{2.16}$$

These data are chosen such that the two detonations move away from each other; i.e., the detonation initially at $x = x_1$ moves to the left and the one initially at $x = x_2$ moves to the right. In this case, there is no collision of detonations at all.

Let

$$T_l = \max_{x<x_1} \frac{p_l(x)}{\rho_l(x)}, \qquad T_m = \min_{x_1<x<x_2} \frac{p_m(x)}{\rho_m(x)}, \qquad T_r = \max_{x>x_2} \frac{p_r(x)}{\rho_r(x)} \tag{2.17}$$

and

$$\theta_n^{(1)} = (T_m - T_l)\vartheta_n + T_l, \qquad \theta_n^{(2)} = (T_m - T_r)\vartheta_n + T_r. \tag{2.18}$$

At any time step t_n, there are $l_1(n) = j_1$ and $l_2(n) = j_2$ with $j_1 \leq j_2$ integers such that

$$z_j^n = \begin{cases} 1 & \text{if } j < l_1(n), \\ 0 & \text{if } l_1(n) \leq j \leq l_2(n), \\ 1 & \text{if } l_2(n) < j. \end{cases} \tag{2.19}$$

The detailed algorithm to find z^{n+1} is as follows:

```
$S_{bp}(k)$ :  set $l_1(n+1) := l_1(n) + 1$;
               For $l = l_1(n)+1, l_1(n), \ldots, l_1(n) - d$, do
                 $l_1(n+1) := l$  if $T_l^{n+1} > \theta_n^{(1)}$;
               set $l_2(n+1) := l_2(n) - 1$;
               For $l = l_2(n)-1, l_2(n), \ldots, l_2(n) + d$, do
                 $l_2(n+1) := l$  if $T_l^{n+1} > \theta_n^{(2)}$;
```

$$z_j^{n+1} = \begin{cases} 1 & \text{if } j < l_1(n+1) \\ 0 & \text{if } l_1(n+1) \leq j \leq l_2(n+1) \\ 1 & \text{if } l_2(n+1) < j \end{cases} \qquad \text{for all } j. \tag{2.20}$$

Remark 2.1. The algorithms presented in this section reply on the assumption that initially the detonation front is already formed. Thus it cannot be used to predict the creation of a detonation from a completely unburned gas (when the initial value of z is identically zero in the entire domain). Since the detonation will be formed beyond the initial layer, one can use a refined calculation within the initial layer and then use the random projection method beyond the initial layer. While one can afford to resolve the initial layer with a refined computation, it is certainly more advantageous to use an underresolved numerical method for all later time beyond the initial layer.

3. The two-dimensional method. In this section, the random projection method is extended to the two space dimensional problem (1.4)–(1.7). For simplicity, we consider the detonation waves in a two-dimensional channel. Let the initial data be

$$(\rho(x,y,0),u(x,y,0),v(x,y,0),p(x,y,0),z(x,y,0)) = \begin{cases} (\rho_l,u_l,0,p_l,0) & \text{if } x \le \xi(y), \\ (\rho_r,u_r,0,p_r,1) & \text{if } x > \xi(y), \end{cases} \tag{3.1}$$

where $\xi(y)$ is a given function of y and these data are chosen such that the detonation moves to the right. Let

$$\theta_n = (T_l - T_r)\vartheta_n + T_r, \qquad T_l = \frac{p_l}{\rho_l}, \qquad T_r = \frac{p_r}{\rho_r} \tag{3.2}$$

with ϑ_n (see (2.9) for detail) being the van der Corput sampling sequence on the interval $[0,1]$.

Let the grid points $(x_i,y_j) = (ih,jh)$, $i,j = \cdots,-1,0,1,\ldots$, with equal mesh spacing h. The time level $t_n = nk$, $k = 0,1,2,\ldots$, are also uniformly spaced with time step k. Let $U^n_{i,j} = \left(\rho^n_{i,j}, m^n_{i,j}, n^n_{i,j}, e^n_{i,j}, (\rho z)^n_{i,j}\right)$ be the approximate solution of $U = (\rho,m,n,e,(\rho z))$ at $(x_i,y_j,t_n) = (ih,jh,nk)$. Let $S_{FG}(k)$ be a standard shock capturing method for (1.4). Notice that, at any time step, for each j, there is an $l_j(n) = j_n$, j_n an integer such that

$$z^n_{i,j} = \begin{cases} 0 & \text{if } j \le l_j(n), \\ 1 & \text{if } i > l_j(n). \end{cases} \tag{3.3}$$

Here, $l_j(n)$ is the location of the jump for z at the grid line $y = y_j$ in the approximate solution at time $t_n = nk$. Then the random project algorithm to find z^{n+1} is as follows:

```
$S_{2p}(k)$ :  For $j$ do
                   Set $l_j(n+1) := l_j(n) - 1$,
                   For $l = l_j(n)-1, l_j(n), \ldots, l_j(n)+d$, do
                       $l_j(n+1) := l$   if $T^{n+1}_{l,j} > \theta_n$;
```

$$z^{n+1}_{i,j} = \begin{cases} 0 & \text{if } i \le l_j(n+1) \\ 1 & \text{if } i > l_j(n+1) \end{cases} \qquad \text{for all } i. \tag{3.4}$$

The stability condition for this algorithm is still the usual CFL condition determined from the convection step $S_{FG}(k)$.

Although the above algorithm is written for a detonation traveling in the direction of the x-axis, little additional effort is needed to extend it to more general cases where the detonation front moves toward all possible directions. One such example is given in the next section (Example 4.8) where the detonation is advancing in circular direction.

4. Numerical examples. In order to verify the performance of the random projection method proposed in this paper, we conduct extensive numerical experiments, including the C-J detonation, strong detonation, collision of a detonation with a shock, a rarefaction wave, and another detonation. We also give two-dimensional examples. In our computation, the operators $S_F(k)$ and $S_{FG}(k)$ are chosen as the second order relaxed scheme [17], which is a TVD scheme without the usage of Riemann solvers or local characteristic decompositions. We choose $d = 5$ in (2.10), (2.15), (2.20), and (3.4) in our computations in this section.

In this section, we compare our numerical results with the resolved solutions. The resolved ones are obtained by solving (1.1)–(1.3) with Heaviside kinetics with a fractional step approach that consists of a second order relaxed scheme for the homogeneous part in (1.1) for one time step, followed by an implicit backward Euler scheme for the chemical reaction. One can see the details in [1] and [11]. Resolved computations based on the Arrhenius kinetics vary only the detailed structure of the reaction zones, so it will not be reported here since the goal of the paper is not to get the accurate reaction zone but all the macroscopic structures outside the reaction zone.

Example 4.1 (C-J detonation). This is Example 4.1 in [1] revisited. We choose here the case of ozone decomposition C-J detonation discussed and computed in [9] and [4]. We use CGS units and the following parameter values:

$$\gamma = 1.4, \qquad q_0 = 0.5196 \times 10^{10}, \qquad \frac{1}{\varepsilon} = K = 0.5825 \times 10^{10}, \qquad T_c = 0.1155 \times 10^{10}.$$

The initial data are taken as the piecewise constant data defining a C-J detonation as a single wave. (Recall that in the C-J model a C-J detonation corresponds to a sonic detonation, or, in other words, a sharp reaction wave that moves at minimal speed relative to the unburnt gas.) The reaction rate K in all the examples is irrelevant for the projection method but is needed for the resolved calculations. The initial state was given by

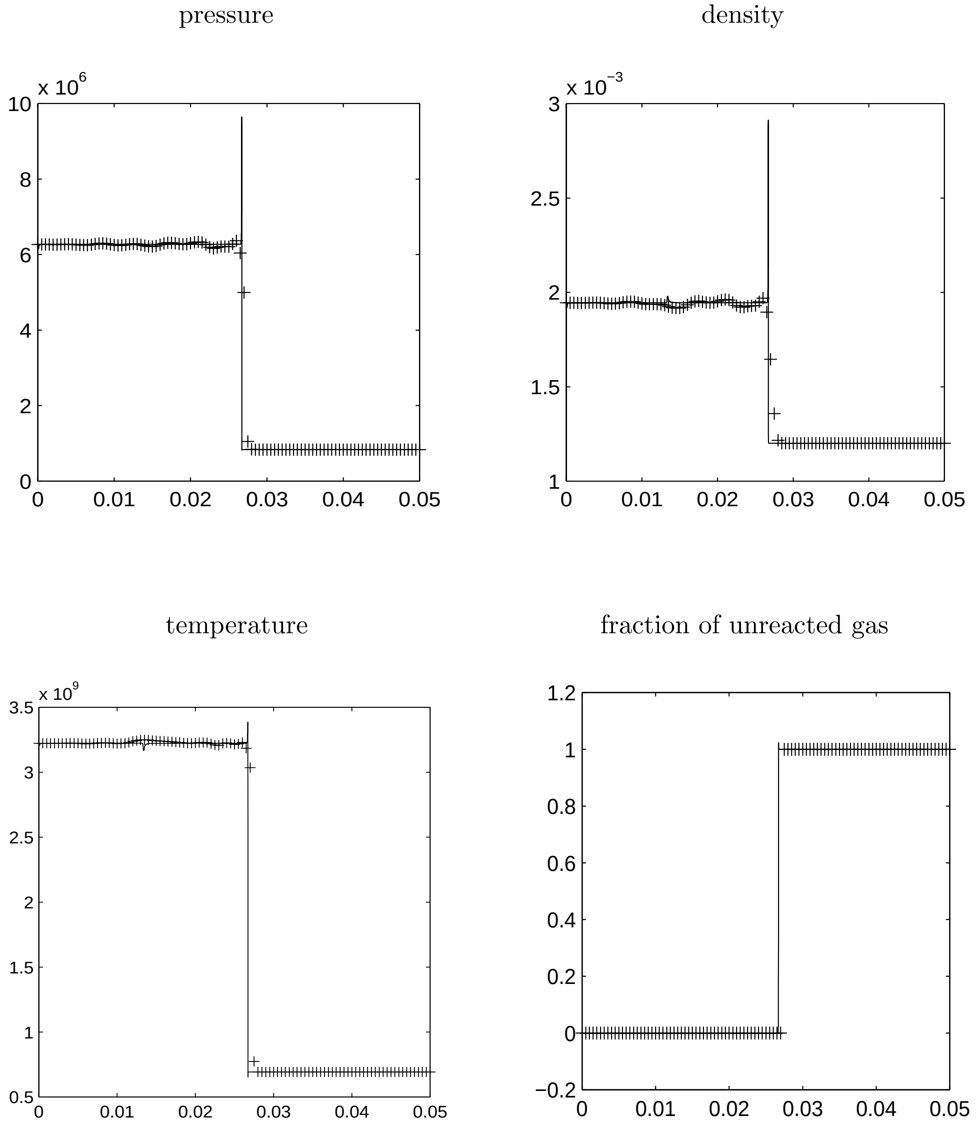
$$(\rho, u, p, z)(x, 0) = \begin{cases} (\rho_l, u_l, p_l, 0) & \text{if } x \le 0.005, \\ (\rho_r, u_r, p_r, 1) & \text{if } x > 0.005, \end{cases}$$

where $p_l = p_{CJ} = 6.270 \times 10^6$, $\rho_l = \rho_{CJ} = 1.945 \times 10^{-3}$, $u_l = u_{CJ} = 4.162 \times 10^4$; and $p_r = 8.321 \times 10^5$, $\rho_r = 1.201 \times 10^{-3}$, $u_r = 0$. The speed of the sharp front in this example is $D = D_{CJ} = 1.088 \times 10^5$. In this example, the width of the reaction zone is approximately 5×10^{-5} [4], [9]. This width is irrelevant for the underresolved calculation but is used for the resolved calculation and for a comparison with the mesh size to check whether the calculation is resolved or underresolved.

This problem is solved on the interval $[0, 0.05]$. The "exact" solution is obtained by using a resolved calculation with $h = 5 \times 10^{-6}$ (i.e., 10001 grid points on the interval $[0, 0.05]$) and $k = 5 \times 10^{-12}$. The mesh size and time step resolve the chemical scale. Now we compare the results obtained by the random projection method and the deterministic method when the reaction scale is underresolved. We use $h = 5 \times 10^{-4}$ (i.e., 101 grid points for the interval $[0, 0.05]$) and $k = 5 \times 10^{-10}$ and output the numerical solution at $t = 2 \times 10^{-7}$.

Figure 4.1(a) shows the numerical solution by using the random projection method (2.10), while Figure 4.1(b) shows the numerical solution obtained by the deterministic method. It can be seen that the random projection method can capture the correct speed of the discontinuity of the C-J detonation wave even when the chemical reaction scale is not numerically resolved. As mentioned earlier, with an underresolved method it is impossible to capture the pressure spike which has a width in the order of reaction scale ε. There are small postshock statistical fluctuations due to the random nature of the method, but they are at an acceptable level. The deterministic method produces spurious waves, as was observed in earlier literatures.

In all of the following examples, the deterministic method always produces spurious waves when the chemical scale is not resolved. We will not report those results and will present only the solutions obtained by the random projection method.



pressure

density

temperature

fraction of unreacted gas

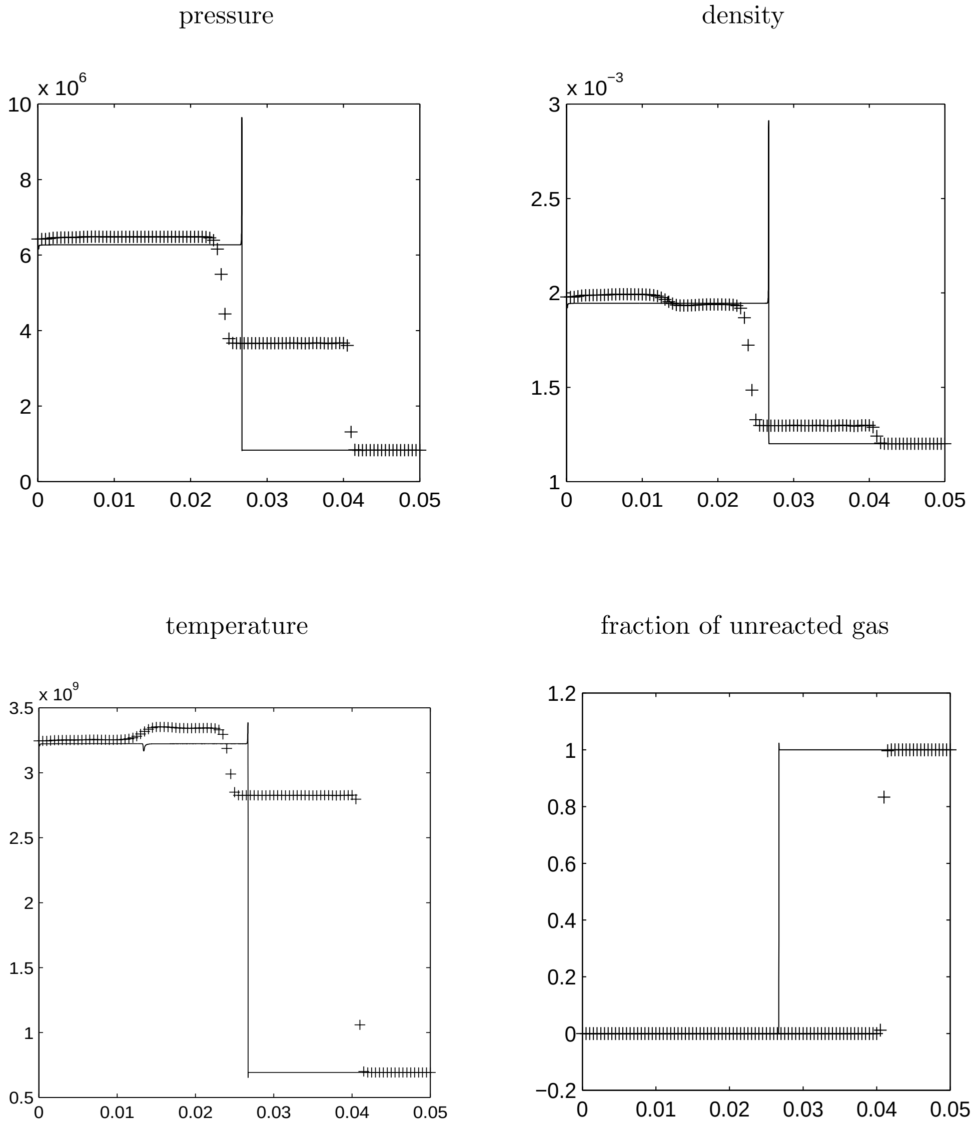
FIG. 4.1. *Numerical solutions of Example* 4.1 *at* $t = 2 \times 10^{-7}$ *calculated with* $h = 5 \times 10^{-4}$, $k = 5 \times 10^{-10}$. −: *"exact" solutions*; ++: *computed solutions.* (a) *The random projection method.*

Example 4.2 (a strong detonation). This is Example 4.3 in [1] revisited. The setup of this example is similar to those in Example 4.1 (i.e., γ, q_0, $K = \frac{1}{\varepsilon}$, and T_c are the same), except that the initial data are changed to

$$(\rho, u, p, z)(x, 0) = \begin{cases} (\rho_l, u_l, p_l, 0) & \text{if } x \le 0.005, \\ (\rho_r, u_r, p_r, 1) & \text{if } x > 0.005, \end{cases}$$

where $u_l = 9.162 \times 10^4 > u_{CJ}$, $\rho_l = \rho_{CJ}$, $p_l = 8.27 \times 10^6 > p_{CJ}$, and p_r, u_r, ρ_r, p_{CJ}, u_{CJ}, and ρ_{CJ} are the same as those in Example 4.1. In this case there is a strong detonation, a contact discontinuity, and a shock, all moving to the right.

The "exact" solution is obtained similarly as that in Example 4.1. Figure 4.2



pressure

density

temperature

fraction of unreacted gas

FIG. 4.1 (*cont.*). (b) *The deterministic method.*

shows the numerical solutions by the random projection method (2.10) with $h = 5 \times 10^{-4}$ (i.e., 101 grid points for the interval $[0, 0.05]$) and $k = 5 \times 10^{-10}$ at time $t = 2 \times 10^{-7}$.

Example 4.3 (collision of a detonation with a rarefaction wave). We choose $\gamma = 1.2$, $q_0 = 50$, $T_c = 3.0$, and $\frac{1}{\varepsilon} = K = 230.75$. The data are taken from [16]. The initial state was given by

$$(\rho, u, p, z)(x, 0) = \begin{cases} (\rho_l, u_l, p_l, 0) & \text{if } x \leq 10, \\ (\rho_m, u_m, p_m, 0) & \text{if } 10 < x \leq 20, \\ (\rho_r, u_r, p_r, 1) & \text{if } 20 < x, \end{cases}$$

where $p_l = 40.0$, $\rho_l = 2.0$, $u_l = 4.0$; $p_m = 54.8244$, $\rho_m = 3.64282$; $u_m = 6.2489$; and

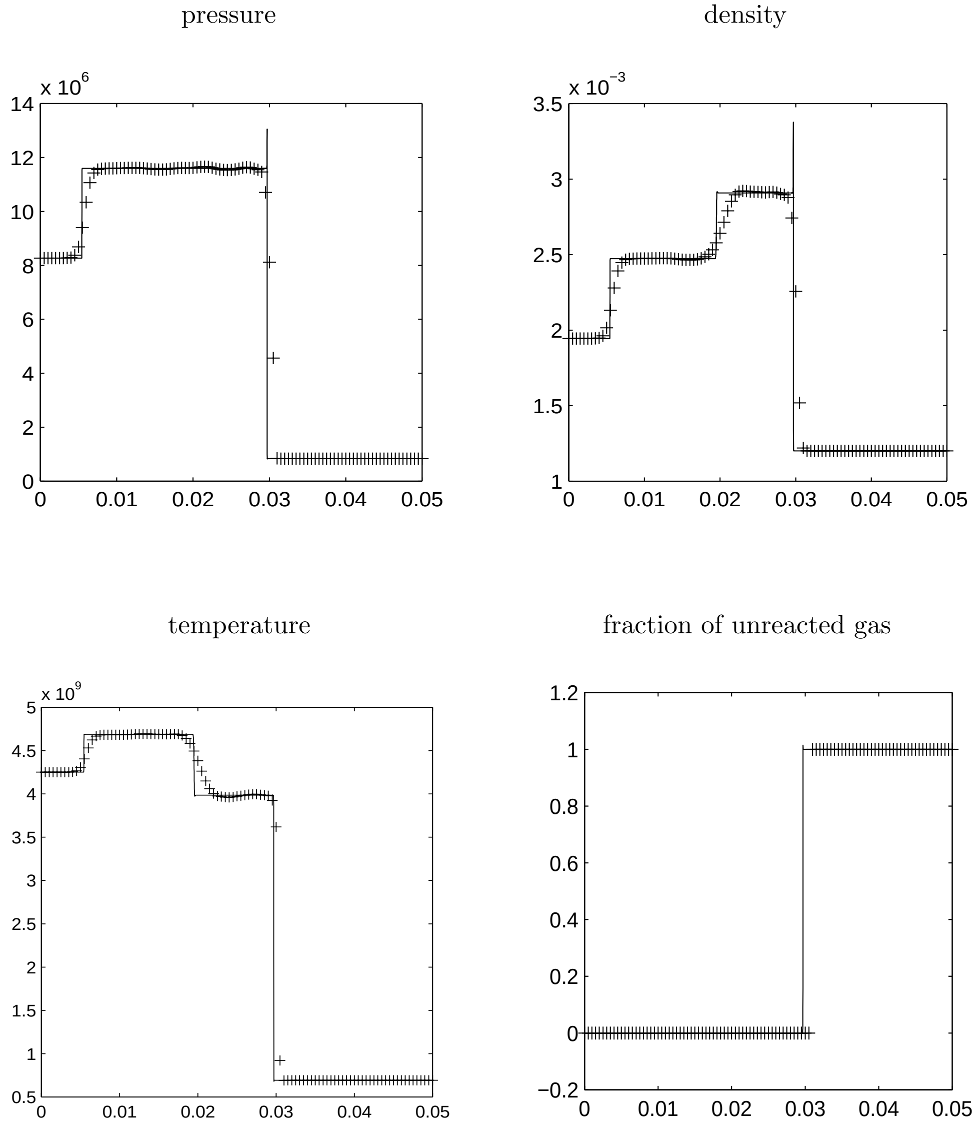


FIG. 4.2. *Numerical results at $t = 2 \times 10^{-7}$ for a strong detonation in Example* 4.2 *calculated by the random projection method* (2.10). $h = 5 \times 10^{-4}$*,* $k = 5 \times 10^{-10}$*. −: "exact" solutions; ++: computed solutions.*

$p_r = 1.0$, $\rho_r = 1.0$, $u_r = 0$. By selecting these data, the "half reaction length" $L_{\frac{1}{2}}$ is the spatial unit 1 [16]. This number is used to compare the mesh sizes in resolved and underresolved numerical experiments.

In this example, there is a right moving detonation, a right moving rarefaction wave, a right moving contact discontinuity, and a left moving rarefaction wave before the right moving rarefaction catches the detonation wave.

This problem is solved on the interval $[0, 100]$. The "exact" solution is obtained by using a resolved calculation $h = 0.005$ (i.e., 20001 grid points on the interval $[0, 100]$) and $k = 0.00025$.

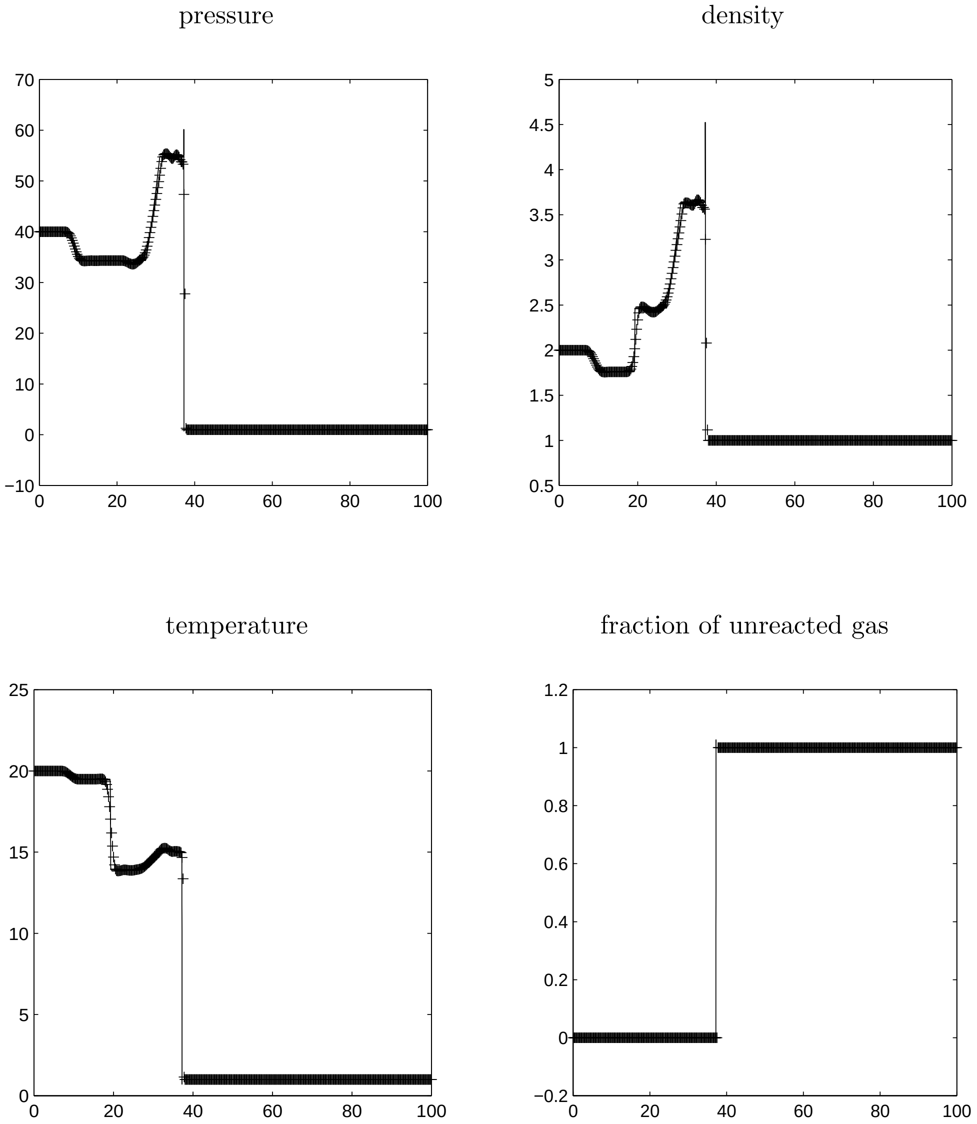
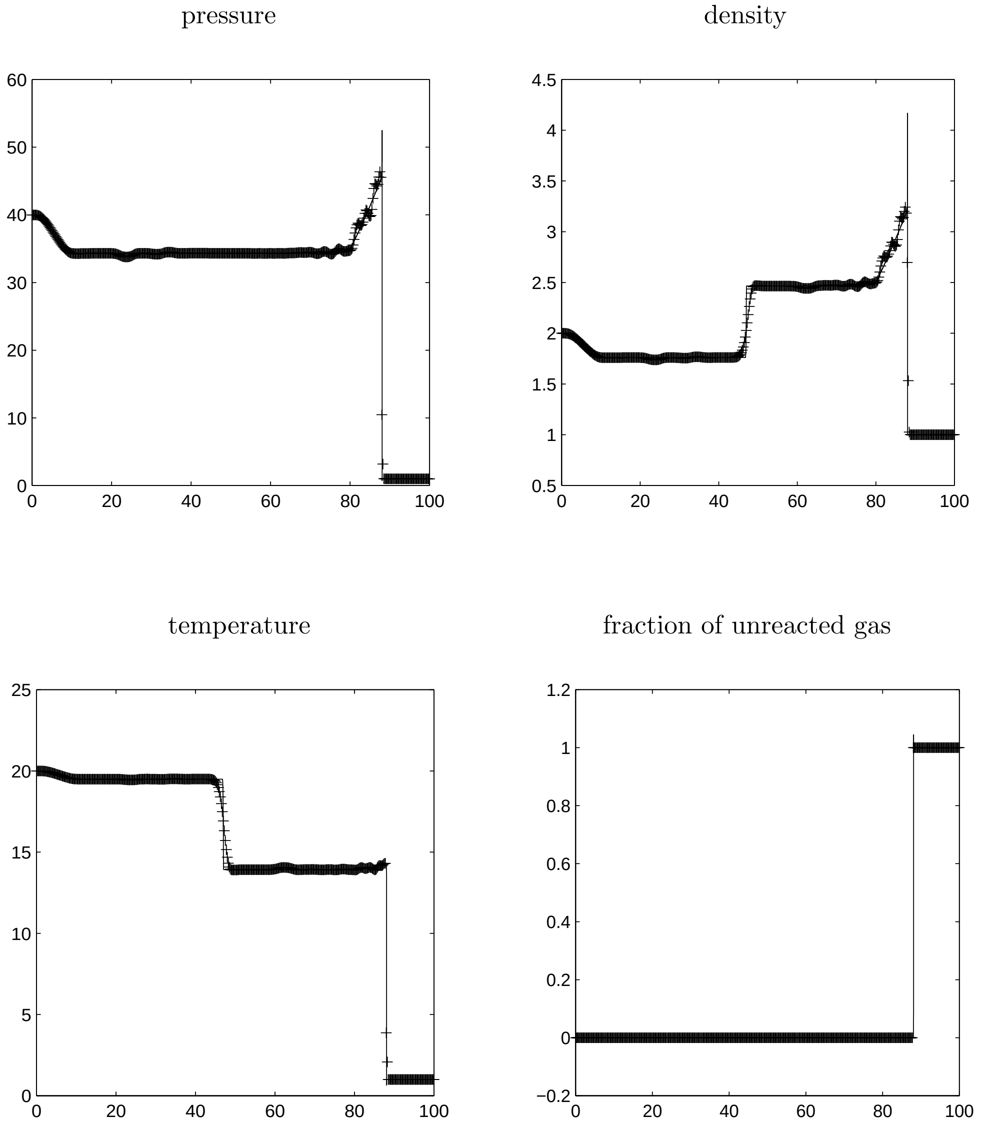


FIG. 4.3. *Numerical results of Example* 4.3 *involving the collision of a detonation with a rarefaction wave using the random projection method* (2.10). $h = 0.25$, $k = 0.01$. $-$: *"exact" solutions;* $++$: *computed solutions.* (a) $t = 2$ *(before collision).*

Figure 4.3 shows the numerical solution by using the random projection method (2.10) with $h = 0.25$ (i.e., 401 grid points for the interval $[0, 100]$) and $k = 0.01$ at time $t = 2$ (before collision) and $t = 8.0$ (after collision), respectively. After the collision, there are some small downstream wiggles which are numerical artifacts and do not appear in the resolved calculation.

Example 4.4 (a detonation interacting with an oscillatory profile). The setup of this problem is similar to those in Example 4.3 (i.e., γ, q_0, and T_c are the same),



pressure

density

temperature

fraction of unreacted gas

FIG. 4.3 (*cont.*). (b) $t = 8$ *(after collision).*

except that we change $K = 1000.0$ and the initial data to

$$(\rho, u, p, z)(x, 0) = \begin{cases} (\rho_l, u_l, p_l, 0) & \text{if } x \le \frac{\pi}{2}, \\ (\rho_r(x), u_r, p_r, 1) & \text{if } \frac{\pi}{2} < x, \end{cases}$$

where $p_l = 21.53134$, $\rho_l = 1.79463$, $u_l = 3.0151$; and $p_r = 1.0$, $\rho_r(x) = 1.0 + 0.5 \sin 2x$, $u_r = 0$.

This problem is solved on the interval $[0, 2\pi]$. The "exact" solutions are obtained by using $h = \frac{\pi}{10000}$ (i.e., 20001 grid points on the interval $[0, 2\pi]$) and $k = \frac{h}{20}$. This is a resolved calculation.

Figure 4.4 shows the numerical solutions by using the random projection method (2.10) with $h = \frac{\pi}{400}$ (i.e., 801 grid points for the interval $[0, 2\pi]$) and $k = \frac{h}{20}$ at time

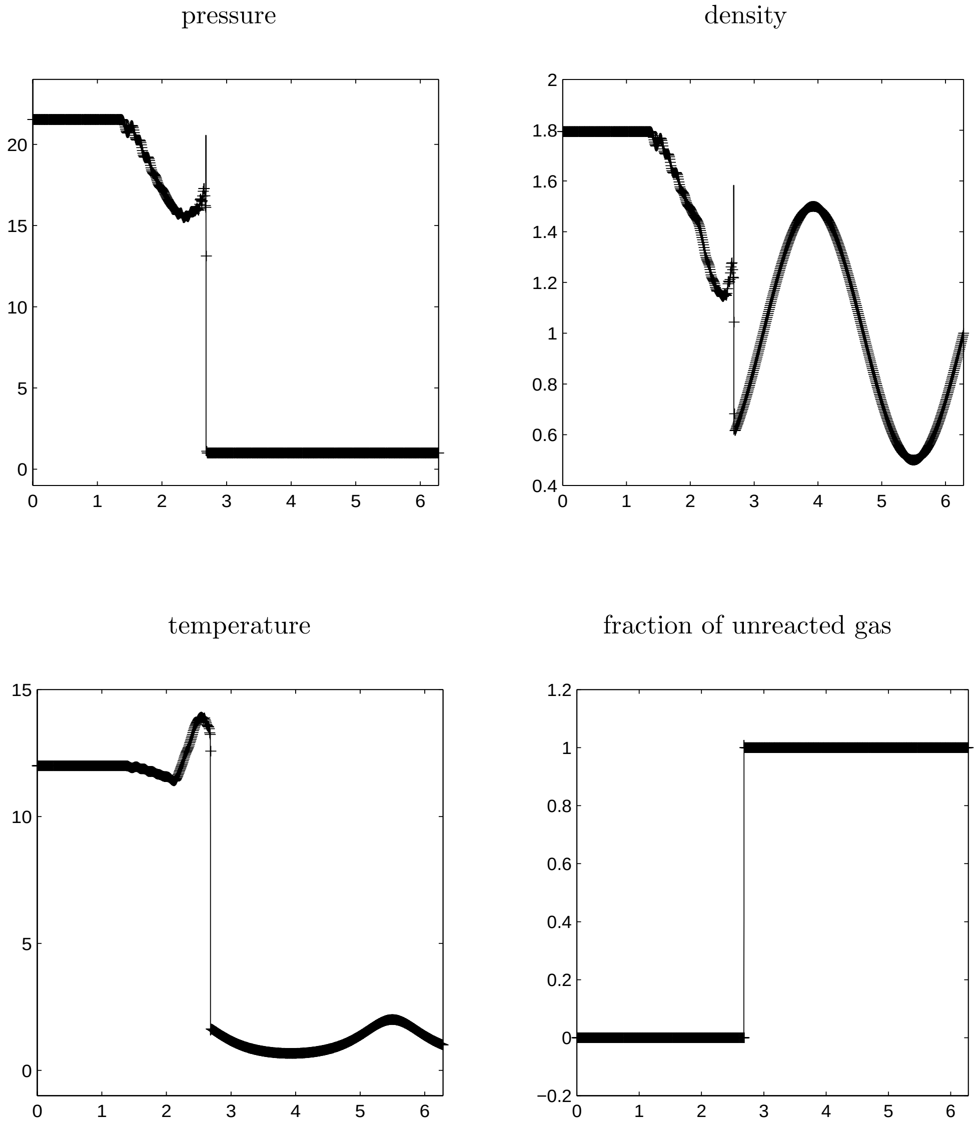


FIG. 4.4. *Numerical results of Example* 4.4 *by the random projection method* (2.10). $h = \frac{\pi}{400}$, $k = \frac{h}{20}$. $-$: *"exact" solutions;* $++$: *computed solutions.* (a) $t = \frac{\pi}{20}$.

$t = \frac{\pi}{20}$ and $t = \frac{\pi}{5}$, respectively.

Example 4.5 (collision of a detonation with a shock, a contact discontinuity, and a rarefaction). The setup of this problem is similar to those in Example 4.3 (i.e., γ, q_0, $K = \frac{1}{\varepsilon}$, and T_c are the same), except that we change the initial data to

$$(\rho, u, p, z)(x, 0) = \begin{cases} (\rho_l, u_l, p_l, 0) & \text{if } x \le 10, \\ (\rho_m, u_m, p_m, 1) & \text{if } 10 < x \le 40, \\ (\rho_r, u_r, p_r, 1) & \text{if } 40 < x, \end{cases}$$

where $p_l = 54.8244$, $\rho_l = 3.64282$, $u_l = 6.2489$; $p_m = 1.0$, $\rho_m = 1.0$, $u_m = 0.0$; and $p_r = 10.0$, $\rho_r = 4.0$, $u_r = 0$.

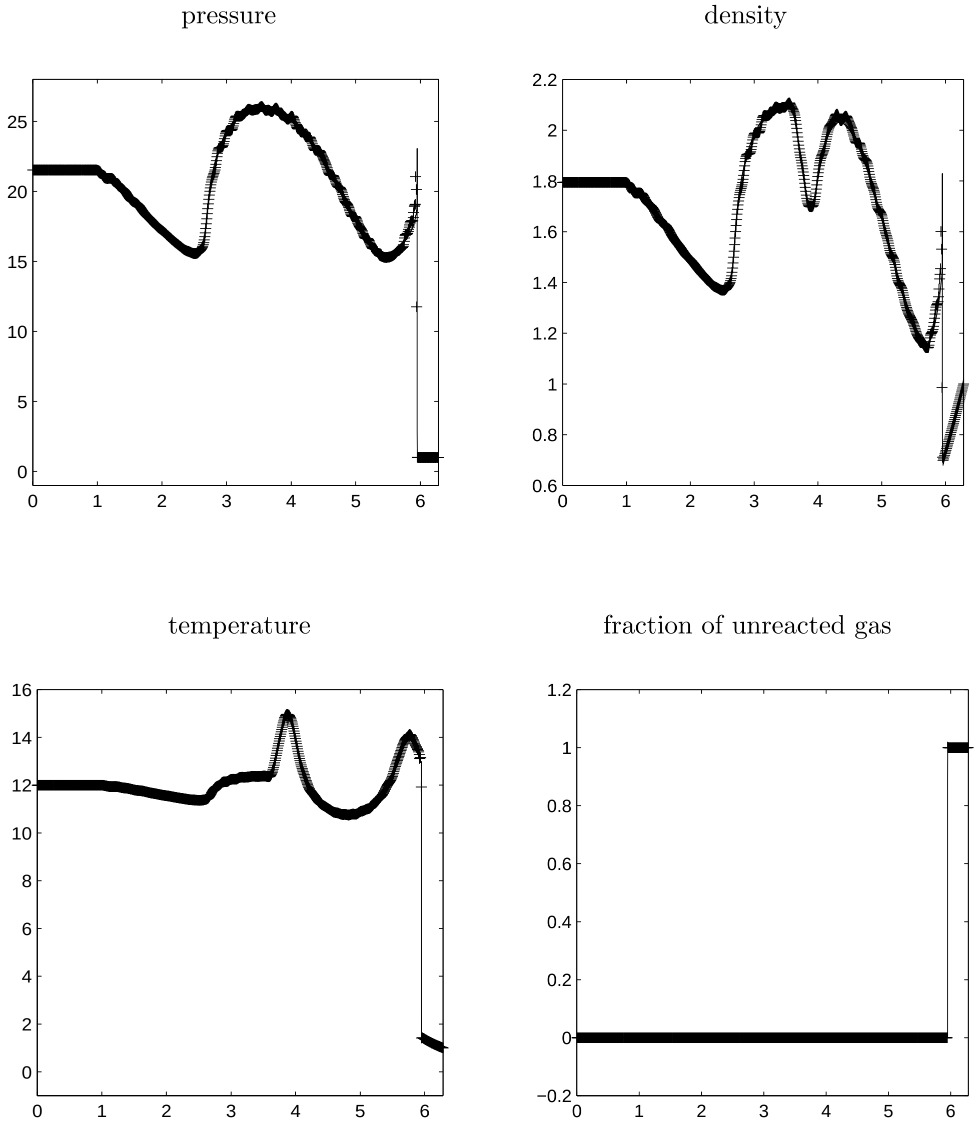


FIG. 4.4 (*cont.*). (b) $t = \frac{\pi}{5}$.

In this example, there is a right moving detonation, a right moving rarefaction, a stationary contact discontinuity, and a left moving shock before a series of collisions occur after the detonation catches up with the other waves.

The "exact" solution is obtained similarly as that in Example 4.3. Figures 4.5(a)–(c) show the numerical solution by using the random projection method (2.10) with $h = 0.125$ (i.e., 801 grid points for the interval $[0, 100]$) and $k = 0.005$ at time $t = 2$ (before collision) $t = 4$ (between the collisions with the shock and with the rarefaction), and $t = 8.0$ (after all collisions), respectively.

This example shows that the random projection method is able to handle the interactions between the detonation and all other waves of a compressible gas.

Example 4.6 (collision of two detonations). The setup in this example is similar

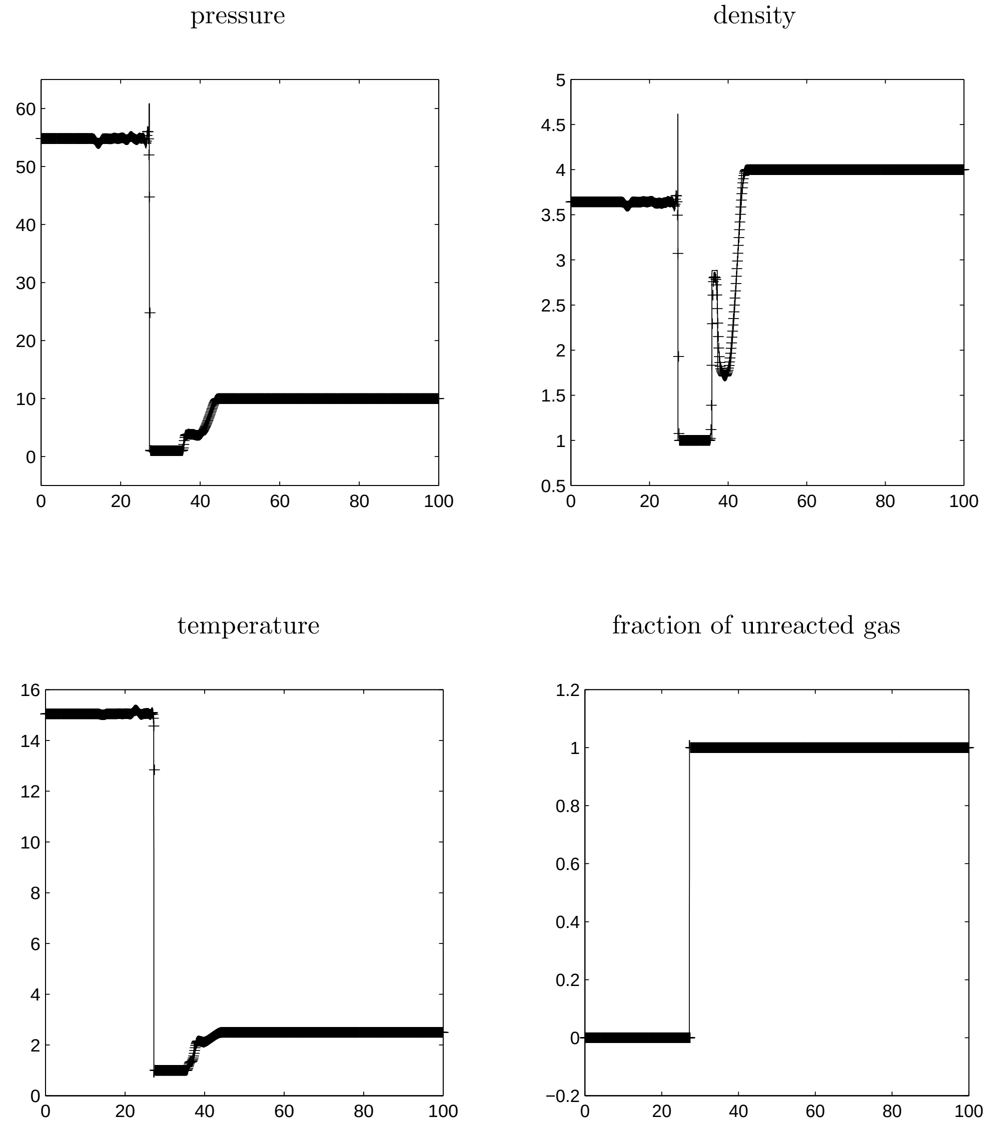


FIG. 4.5. *Numerical results of Example* 4.5 *involving the collisions of a detonation with a shock and then a rarefaction wave by the random projection method* (2.10). $h = 0.125$*,* $k = 0.005$. $-$: *"exact" solutions;* $++$: *computed solutions.* (a) $t = 2$ *(before collision).*

to those in Example 4.3 (i.e., γ, q_0, $K = \frac{1}{\varepsilon}$, and T_c are the same), except that we change the initial data to

$$(\rho, u, p, z)(x, 0) = \begin{cases} (\rho_l, u_l, p_l, 0) & \text{if } x \le 10, \\ (\rho_m, u_m, p_m, 1) & \text{if } 10 < x < 90, \\ (\rho_r, u_r, p_r, 0) & \text{if } 90 \le x, \end{cases}$$

where $p_l = 30.0$, $\rho_l = 1.79463$, $u_l = 3.0151$; $p_m = 1.0$, $\rho_m = 1.0$, $u_m = 0.0$; and $p_r = 21.53134$, $\rho_r = 1.79463$, $u_r = -8.0$.

In this example, there is a right moving detonation, a left moving strong det-

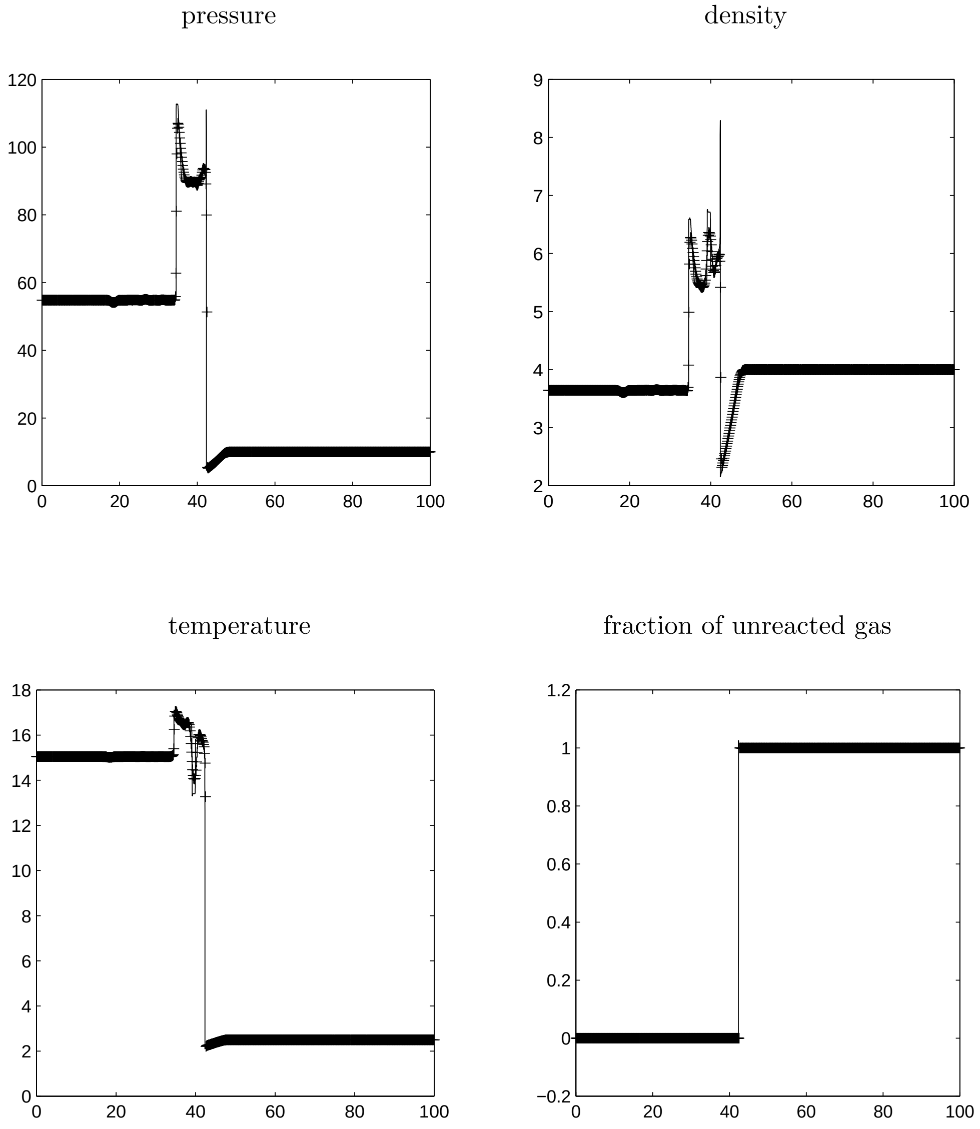


FIG. 4.5 (*cont.*). (b) $t = 4$ *(between the collisions with the shock and the rarefaction).*

onation, and other waves. After some time, there is a collision between the two detonations.

The "exact" solution is obtained similarly as that in Example 4.3. Figure 4.6 shows the numerical solution by using the random projection method (2.15) with $h = 0.25$ (i.e., 401 grid points for the interval $[0, 100]$) and $k = 0.01$ at time $t = 4$ (before collision) and $t = 6.0$ (after collision), respectively. After the collisions, the detonation becomes extinct and two shocks are formed. This example shows that the random projection is valid even after the detonation disappears.

From the above examples, we can see that the random projection method works very well for one-dimensional detonation wave problems even if the reaction scale is not numerically resolved. It not only captures the correct speeds of detonations but

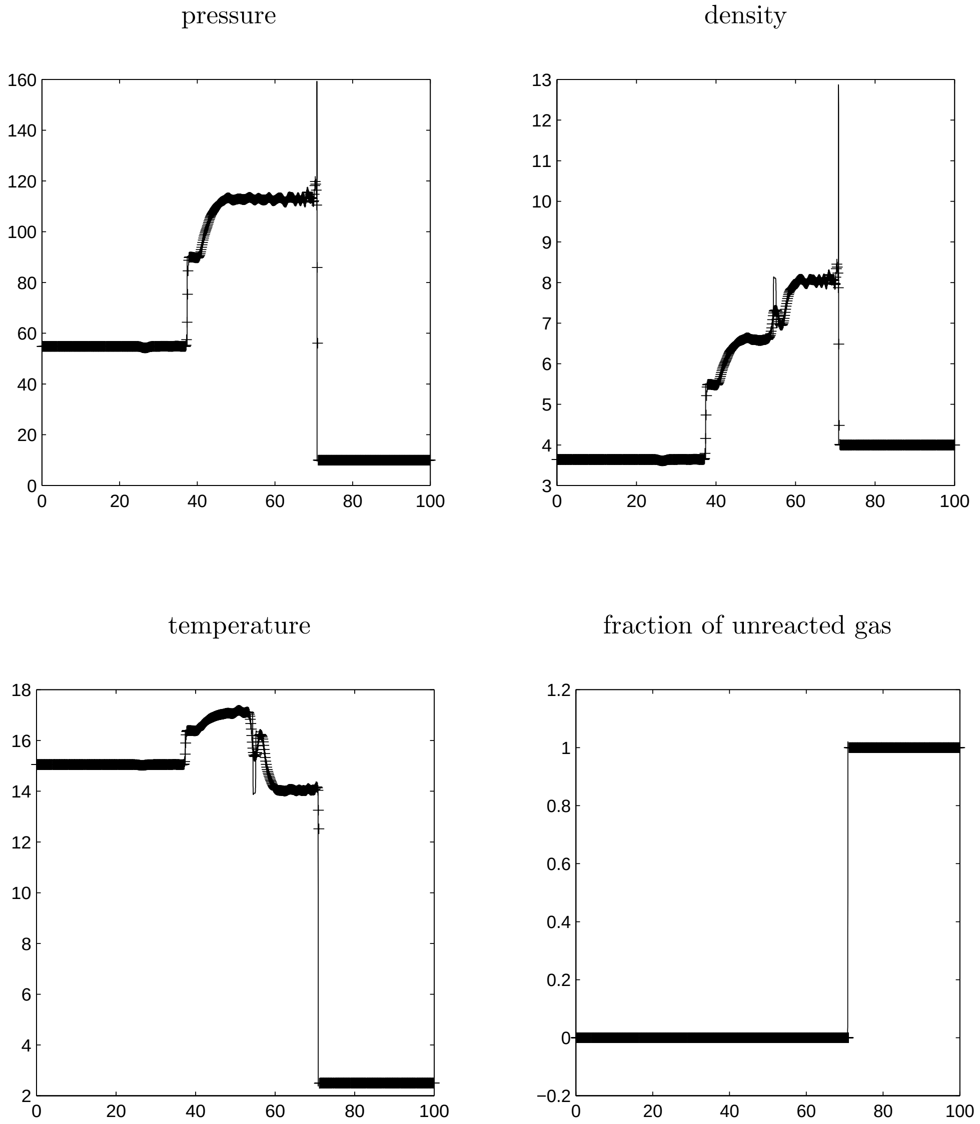


FIG. 4.5 (*cont.*). (c) $t = 8$ *(after all collisions).*

also is able to handle the interactions between detonations and between a detonation with other waves of a compressible gas. Its applicability remains after the extinction of the detonation.

Example 4.7 (a two-dimensional circular detonation front). We consider the problem (1.4)–(1.7) in a two-dimensional channel; the upper and lower boundaries are solid walls. We choose γ, q_0, $K = \frac{1}{\varepsilon}$, and T_c the same as those in Example 4.3. The initial data (3.1) are chosen as $p_l = 54.8244$, $\rho_l = 3.64282$, $u_l = 6.2489$, $v_l = 0.0$; and $p_r = 1.0$, $\rho_r = 1.0$, $u_r = 0.0$, $v_r = 0.0$. The corresponding initial setup in one dimension is a strong detonation. This problem is solved on $[0, 300] \times [0, 50]$ with a

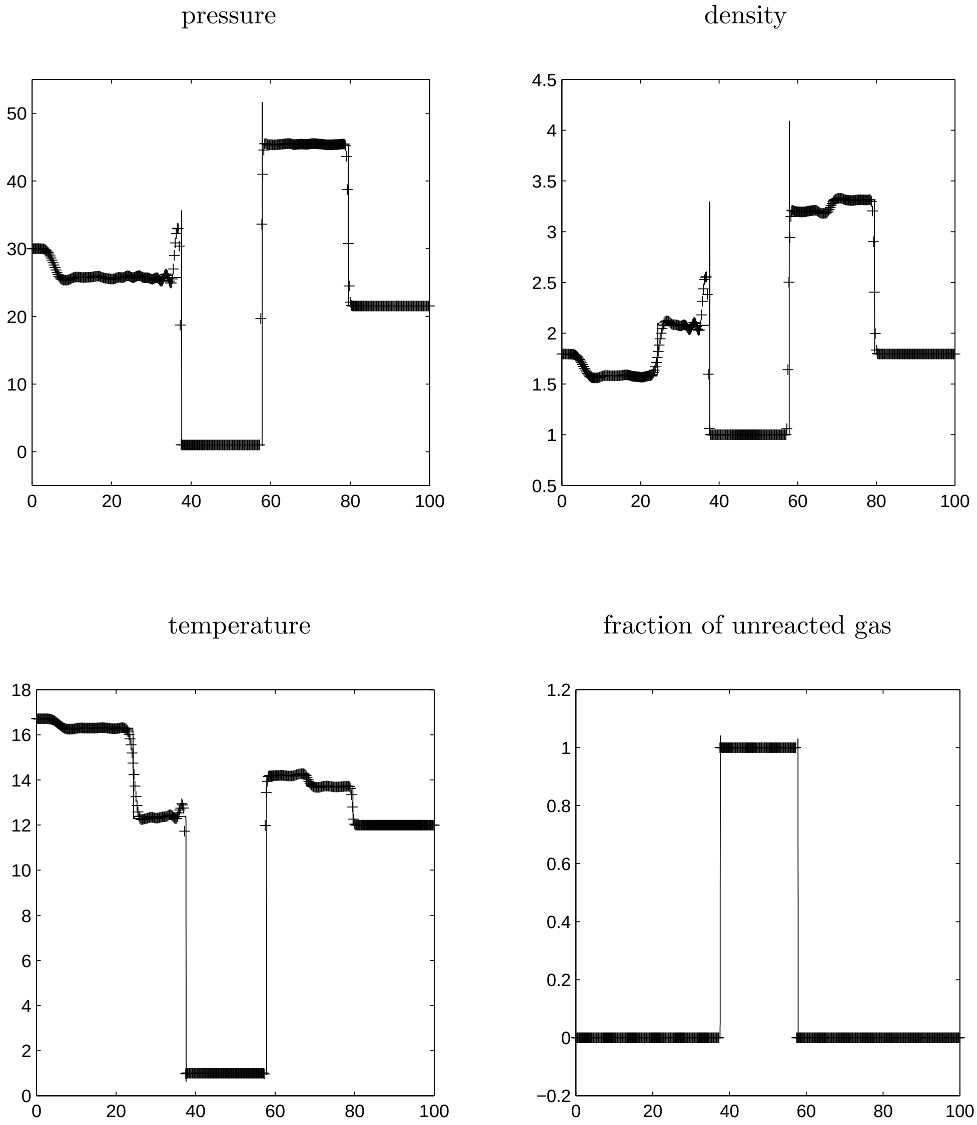


FIG. 4.6. *Numerical results of Example* 4.6 *involving the collision of two detonations by the random projection method* (2.15)*.* $h = 0.25$*,* $k = 0.01$*.* $-$: *"exact" solutions;* $++$: *computed solutions.* (a) *at* $t = 4$ *(before collision).*

301×51 mesh, and

$$\xi(y) = \begin{cases} 10, & |y-25| \geq 15, \\ 25 - |y-25|, & |y-25| < 15. \end{cases}$$

Thus the mesh size $h = 1$. The time step is chosen as $k = 0.01$.

Figure 4.7 shows density contours at several different times. One can see that the triple points, which are the important features of the solution, travel in the transverse direction and bounce back and forth against the upper and lower walls. On the con-

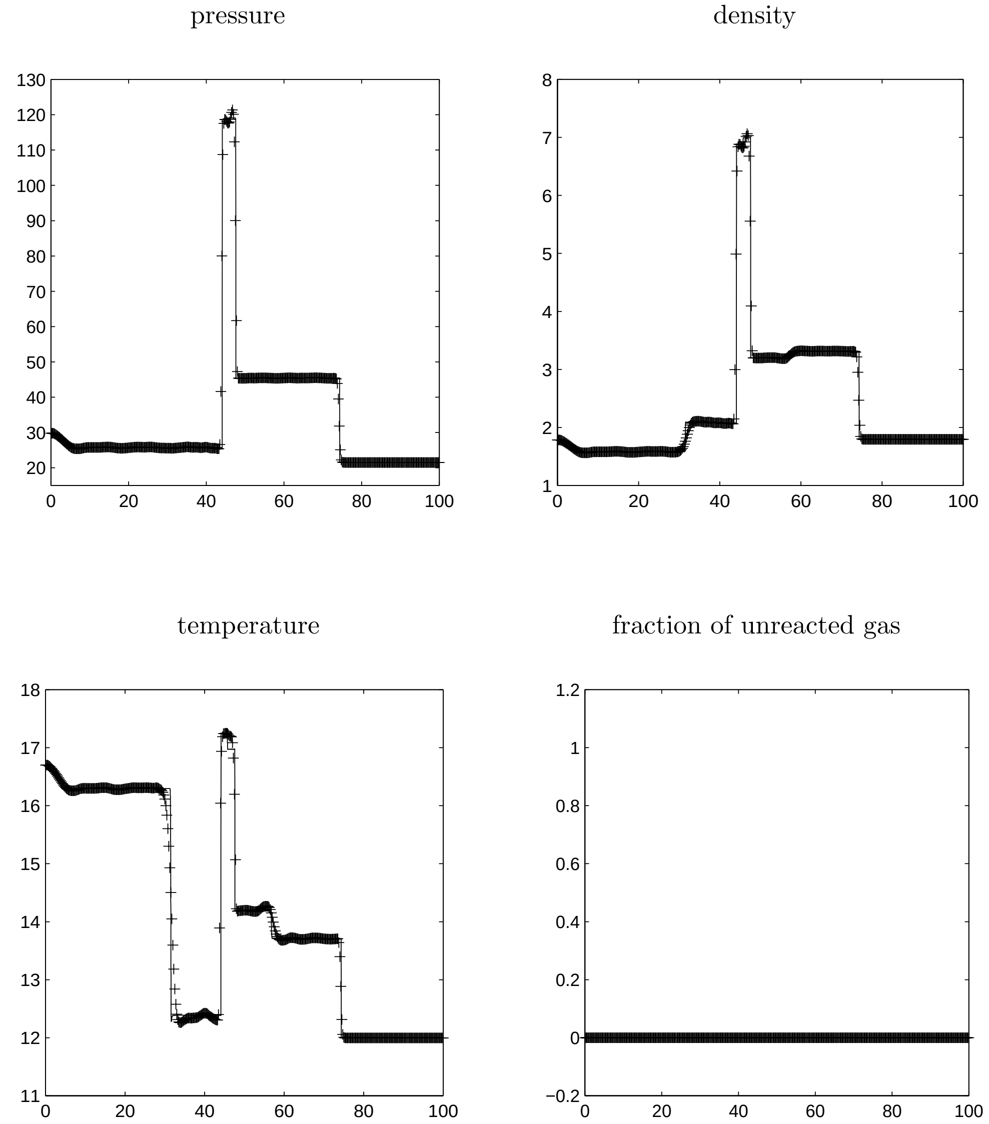


FIG. 4.6 (*cont.*). (b) *at $t = 6$. After collision the detonation becomes extinct and the gas is completely burned.*

trary, the triple points cease to move after some time by using the usual deterministic method [11].

Example 4.8 (another two-dimensional detonation wave). This is a two-dimensional example with radial symmetry. The parameters are chosen as

$$\gamma = 1.2, \qquad q_0 = 50, \qquad K = \frac{1}{\varepsilon} = 1000, \qquad \text{and} \qquad T_c = 2.0.$$

A similar example was used in [15].

The initial values consist of totally burnt gas inside of a circle with radius 10 and totally unburnt gas everywhere outside of the circle. Furthermore, the unburnt and burnt states are chosen in a way analogous to the one-dimensional case, i.e.,

$$(\rho, u, v, p, z)(x, y, 0) = \begin{cases} (\rho_l, u_l(x, y), v_l(x, y), p_l, 0) & \text{if } r \leq 10, \\ (\rho_r, u_r, v_r, p_r, 1) & \text{if } r > 10, \end{cases}$$

where $r = \sqrt{x^2 + y^2}$; $p_l = 21.53134$, $\rho_l = 1.79463$, $u_l(x, y) = 10x/r$, $v_l(x, y) = 10y/r$; and $p_r = 1.0$, $\rho_r = 1.0$, $u_r = 0.0$, $v_r = 0.0$.

This is a radially symmetric problem, and the important feature is that the detonation front is circular. This problem is solved on the domain $[-50, 50] \times [0, 50]$ with mesh size $h = 0.5$ and time step $k = 0.01$. Solid wall boundary conditions are used along $x = 0$. Inflow boundary condition is used along $x = -50$. Outflow boundary conditions are used along $x = 50$ and $y = 50$.

The random projection algorithm for this example is as follows. Notice that at any time step $t = t_n$, for each j, there are two integers, $l_j(n) = l_n$ and $r_j(n) = r_n$, such that $l_n \leq r_n$ and

$$z_{i,j}^n = \begin{cases} 1 & \text{if } j \leq l_j(n), \\ 0 & \text{if } l_j(n) < j < r_j(n), \\ 1 & \text{if } i \geq r_j(n). \end{cases} \tag{4.1}$$

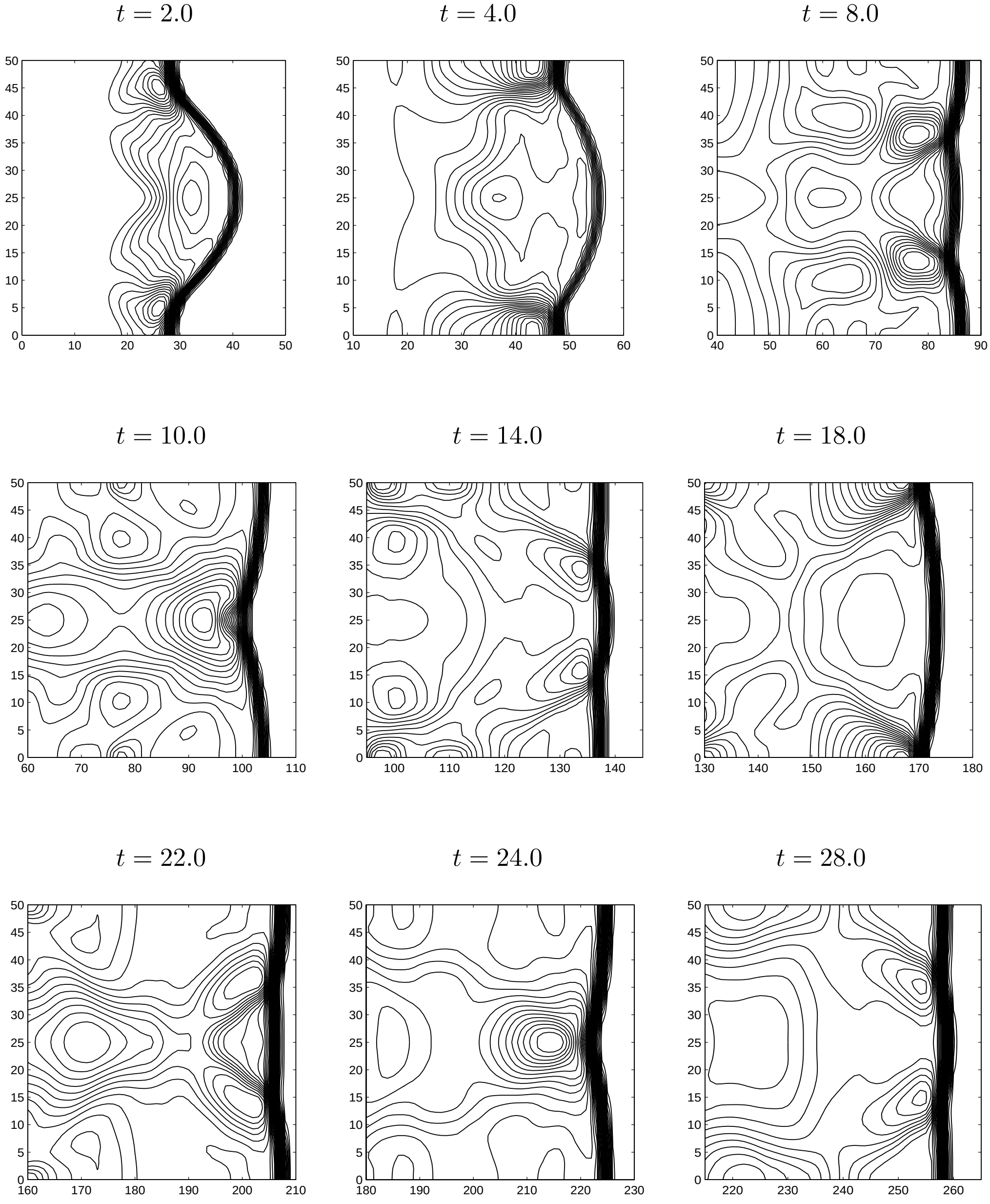
Here $l_j(n)$ and $r_j(n)$ are the left and right locations of the jump for z at the grid line $y = y_j$ in the approximate solution at time $t_n = nk$. Then the random project algorithm to find z^{n+1} is as follows:

```
$\tilde{S}_{2p}(k)$ :  For $j$ do
        Set $l_j(n+1) := l_j(n) + 1$,
        For $l = l_j(n) + 1, l_j(n), \ldots, l_j(n) - d$, do
            $l_j(n+1) := l$   if $T_{l,j}^{n+1} > \theta_n$,
        Set $r_j(n+1) := r_j(n) - 1$,
        For $r = r_j(n) + 1, r_j(n), \ldots, r_j(n) + d$, do
            $r_j(n+1) := r$   if $T_{r,j}^{n+1} > \theta_n$,
        If $l_j(n+1) > r_j(n+1)$ then
            $l_j(n+1) := r_j(n+1) := (l_j(n+1) + r_j(n+1))/2$,
```

$$z_{i,j}^{n+1} = \begin{cases} 1 & \text{if } i \leq l_j(n+1) \\ 0 & \text{if } l_j(n+1) < i < r_j(n+1) \\ 1 & \text{if } i \geq r_j(n+1) \end{cases} \quad \text{for all } i. \tag{4.2}$$

The stability condition for this algorithm is still the usual CFL condition determined from the convection step.

Figure 4.8(a) shows the velocity fields and Figure 4.8(b) shows profiles of the pressure p, temperature T, and 30 times the mass fraction of unburnt gas, $30z$ (here we show $30z$, not z itself, for better visualization), on the line $y = x$ ($0.0 \leq x \leq 50$) by the random projection method (4.2) at time $t = 1$, $t = 2$, $t = 4$, and $t = 5$, respectively.


$t = 2.0$ $t = 4.0$ $t = 8.0$

$t = 10.0$ $t = 14.0$ $t = 18.0$

$t = 22.0$ $t = 24.0$ $t = 28.0$

FIG. 4.7. *Numerical density contours for Example* 4.7 *by the two-dimensional random projection method* (3.4). $h = 1.0$, $k = 0.01$.

It can be seen that the detonation front remains circular, and no spurious nonphysical wave is generated when using the random projection method (4.2). On the other hand, if one uses the deterministic method, the detonation front does not remain circular, and spurious nonphysical wave is generated if the same grid size and time step are used.

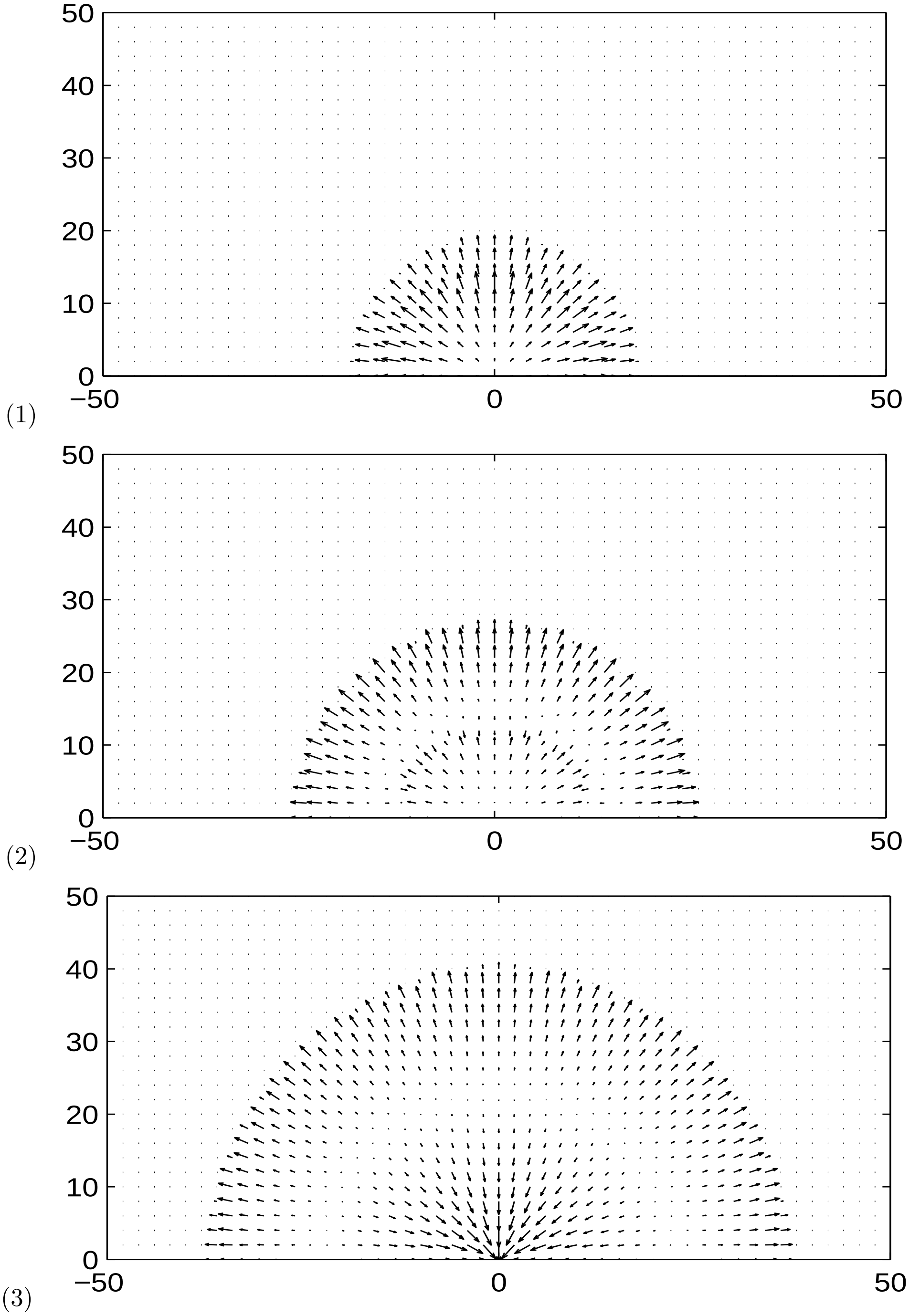


FIG. 4.8. *Numerical solutions of Example* 4.8 *calculated by the two-dimensional random projection method* (4.2) *with* $h = 0.5$, $k = 0.01$. (1): $t = 1.0$, (2): $t = 2.0$, (3): $t = 4.0$. (a) *Velocity fields at different times.*

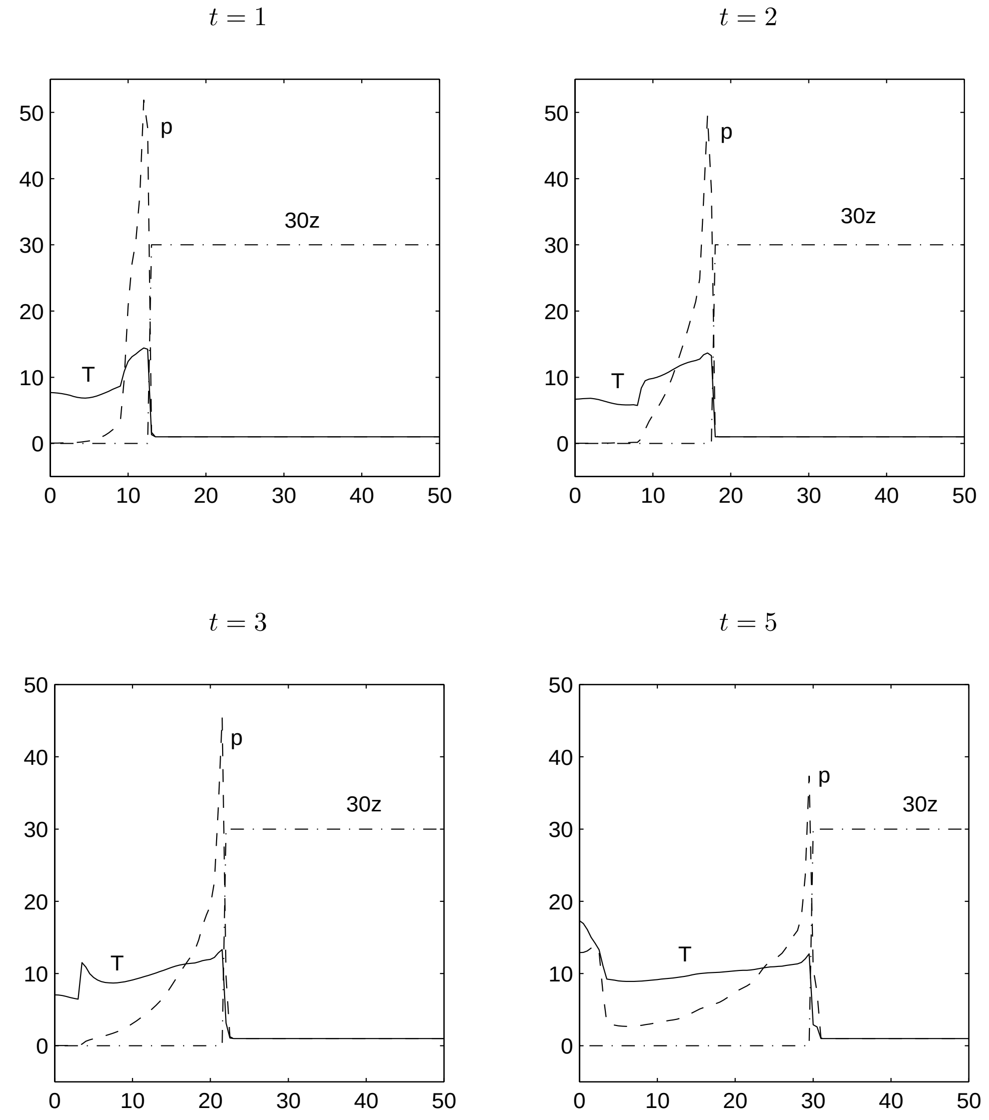


FIG. 4.8 (*cont.*). (b) *Profiles of pressure* p *(- -), temperature* T *(–), and the fraction of unburnt gas multiplied by* 30, $30z$ *(- ·) on the line* $y = x (0 \leq x \leq 50)$ *at different times.*

5. Conclusions.

In this paper we presented a simple and robust random projection method for underresolved numerical simulation of stiff detonation waves in chemically reacting flows. This method is based on the random projection method proposed by the authors for general hyperbolic systems with stiff reaction terms [1], where the ignition temperature is randomized in a suitable domain. The method is simplified using the equations of instantaneous reaction and then extended to handle the interactions of detonations. Extensive numerical experiments, including interaction of detonation waves, and in two dimensions, demonstrate that this method, although very simple and efficient, is very reliable and robust in calculating a wide range of problems in reacting flows.

In [3] this method is generalized to multispecies reactions.

Acknowledgments. W. Bao thanks the School of Mathematics of the Georgia Institute of Technology and the Department of Mathematics of the University of Wisconsin-Madison for their hospitality during his extended visits there.

REFERENCES

[1] W. Bao and S. Jin, *The random projection method for hyperbolic conservation laws with stiff reaction terms,* J. Comput. Phys., 163 (2000), pp. 216–248.

[2] W. Bao and S. Jin, *Error Estimates on the Random Projection Methods for Hyperbolic Conservation Laws with Stiff Reaction Terms,* unpublished notes, 1999.

[3] W. Bao and S. Jin, *The Random Projection Method for Stiff Multi-Species Detonation Capturing,* preprint, 2000.

[4] M. Ben-Artzi, *The generalized Riemann problem for reactive flows*, J. Comput. Phys., 81 (1989), pp. 70–101.

[5] A. C. Berkenbosch, E. F. Kaasschieter, and R. Klein, *Detonation capturing for stiff combustion chemistry*, Combust. Theory Model., 2 (1998), pp. 313–348.

[6] A. Bourlioux, A. J. Majda, and V. Roytburd, *Theoretical and numerical structure for unstable one-dimensional detonations,* SIAM J. Appl. Math., 51 (1991), pp. 303–343.

[7] A. J. Chorin, *Random choice methods with applications to reacting gas flow,* J. Comput. Phys., 25 (1977), pp. 253–272.

[8] P. Colella, *Glimm's method for gas dynamics,* SIAM J. Sci. Statist. Comput., 3 (1982), pp. 76–110.

[9] P. Colella, A. Majda, and V. Roytburd, *Theoretical and numerical structure for reacting shock waves,* SIAM J. Sci. Statist. Comput., 7 (1986), pp. 1059–1080.

[10] R. Courant and K. O. Friedrichs, *Supersonic Flow and Shock Waves,* Interscience, New York, 1967.

[11] B. Engquist and B. Sjogreen, *Robust Difference Approximations of Stiff Inviscid Detonation Waves,* UCLA CAM Report 91-03, University of California-Los Angeles, Los Angeles, CA, 1991.

[12] J. Glimm, *Solutions in the large for nonlinear hyperbolic systems of equations,* Comm. Pure Appl. Math., 18 (1965), pp. 697–715.

[13] D. F. Griffiths, A. M. Stuart, and H. C. Yee, *Numerical wave propagation in an advection equation with a nonlinear source term,* SIAM J. Numer. Anal., 29 (1992), pp. 1244–1260.

[14] J. M. Hammersley and D. C. Handscomb, *Monte Carlo Methods*, Methuen, London, 1965.

[15] C. Helzel, R. J. LeVeque, and G. Warnecke, *A modified fractional step method for the accurate approximation of detonation waves,* SIAM J. Sci. Comput., 22 (2000), pp. 1489–1510.

[16] P. Hwang, R. P. Fedkiw, B. Merriman, A. K. Karagozian, and S. J. Osher, *Numerical resolution of pulsating detonation waves,* Combust. Theory Model., 4 (2000), pp. 217–240.

[17] S. Jin and Z. P. Xin, *The relaxation schemes for systems of conservation laws in arbitrary space dimensions,* Comm. Pure Appl. Math., 48 (1995), pp. 235–276.

[18] R. J. LeVeque and H. C. Yee, *A study of numerical methods for hyperbolic conservation laws with stiff source terms,* J. Comput. Phys., 86 (1990), pp. 187–210.

[19] R. B. Pember, *Numerical methods for hyperbolic conservation laws with stiff relaxation* I. *Spurious solutions,* SIAM J. Appl. Math., 53 (1993), pp. 1293–1330.

[20] M. A. Sussman, *Source Term Evaluation for Combustion Modeling,* AIAA paper 93-0239, 1993.

[21] F. A. Williams , *Combustion Theory*, Addison-Wesley, Reading, MA, 1965.

SIAM J. SCI. COMPUT. © 2001 Society for Industrial and Applied Mathematics
Vol. 23, No. 3, pp. 1027–1041

AN INTERPOLATING POLYNOMIAL METHOD FOR NUMERICAL CONFORMAL MAPPING[*]

R. MICHAEL PORTER[†]

Abstract. A simple algorithm is presented for numerical approximation of conformal mappings for simply connected planar regions. The desired mapping is represented by a normalized polynomial of degree N, determined by the boundary values which it is presumed to take at the Nth roots of unity; then its values at the N intermediate $2N$th roots of unity are used to correct the initial guess. Each iteration, which consists essentially of matrix multiplication and boundary projection, costs $O(N \log N)$ arithmetic operations. Numerical results are provided to confirm linear convergence of the algorithm.

Key words. numerical conformal mapping, interpolating polynomial

AMS subject classification. 30C30

PII. S1064827599355256

1. Introduction.

We present here a method for calculating boundary values of conformal mappings from the unit disk D_0 to simply connected planar domains D. Following the notation of [4], suppose that the boundary ∂D is parametrized by the simple closed curve $\gamma : [0, L] \to \mathbb{C}$ and that the Riemann mapping $f : D_0 \to D$, normalized by $f(0) = 0$ (and perhaps $f'(0) > 0$ or $f(1) = w_0$), sends the Nth roots of unity $(\zeta_N)^j$ to $w_j \in \partial D$. We wish to compute the w_j, or, equivalently, the values $\sigma_0, \dots, \sigma_{N-1}$ such that $\gamma(\sigma_j) = w_j$. As is well known, f is determined in the interior of the domain D by its boundary values (see [1], [10]).

The method derived here resembles in some ways that given by Fornberg in [6]. In Fornberg's method, the starting boundary data, given by an N-tuple of purported image points $\vec{w}^* = (w_j^*)$, determine the set of Fourier coefficients of a boundary function $\zeta_N^j \mapsto w_j^*$, and equations are derived for replacing w_j^* with the true values w_j, characterized by the property that the Fourier coefficients with negative indices all vanish. When this is accomplished, the corresponding Laurent series is a polynomial and gives an approximation to f.

In our method, in contrast, we start with the interpolating polynomial $P = P_{\vec{w}^*}$ of degree N defined by $P(0) = 0$, $P(\zeta_N^j) = w_j^*$. The criterion we use to adjust the values of w_j^* is that P *should send the remaining* $2N$*th roots of unity,* $\zeta_N^{j+(1/2)}$ *onto the curve* γ *as well.*

Interpolating polynomials were studied by Wegmann [13]. There it is shown that for ∂D sufficiently smooth and for N sufficiently large, there is a unique (suitably normalized) polynomial of degree $N+1$ close to f which takes the $2N$th roots of unity to points cyclicly ordered along ∂D. These polynomials approach f as $N \to \infty$. Two algorithms were derived there for approximating these polynomials. The method we give here (together with several variants) is much simpler than the aforementioned one: each iteration consists of two multiplications by fixed matrices (alternatively,

[*] Received by the editors April 27, 1999; accepted for publication (in revised form) May 4, 2001; published electronically September 26, 2001. This work was partially supported by CONACyT grant 35521E.

http://www.siam.org/journals/sisc/23-3/35525.html

[†] Departamento de Matemáticas, Centro de Investigación y de Estudios Avanzados del I.P.N., Apartado Postal 14-740, 07000 México D. F., Mexico (mike@math.cinvestav.mx).

four discrete Fourier transforms) and two projections to ∂D of the N-tuples thus obtained.

Although faster methods than that of [6] have been developed, we use an example given there for the purpose of numerical comparison, since that example is well known and there exist in the literature comparisons of that method with others. (See, for example, [11], [12] for calculations with methods by Fornberg, Theodorsen, and Wegmann, each of which proves superior in certain instances.) Further, this example is an excellent illustration of the problems caused by the well-known "crowding" phenomenon [4], [9], that is, the tendency for $|f'|$ to be much larger at certain parts of ∂D_0 than others, especially, in loose terms, at points which are sent farther from the origin. This causes the images of the grid points to be sparse at some parts of ∂D and heavily concentrated at others, so a large number of grid points is intrinsically required for a good approximation over the whole boundary.

One must bear in mind that the error in approximating f stems from two contributions: the error of our method in approximating P and the degree to which P serves as an approximation to f (as studied in [13]). For an analysis of the inherent error in using a truncated power series in place of f, see [4]. For methods which calculate the inverse mapping $D \to D_0$, see the references in [9], [10]. In many such cases the most appropiate algorithm would be a Schwarz–Christoffel method, such as the excellent algorithm given in [5], which completely eliminates the problem of crowding for polygons.

The method given here is a "boundary method" in the sense that we are not concerned explicitly with the values of $f(z)$ for $0 < |z| < 1$. Some features of our method are the following. For a fixed value of N, linear convergence is observed for most domains. No extra work (e.g., interpolation) is required for doubling the number of grid points. The method produces the image values w_j directly, rather than giving, say, Taylor coefficients (which would then require an additional calculation to find the w_j). Once these image points are found, the values at general boundary points $P(e^{i\beta}\zeta_N^j)$ can be obtained easily via a discrete Fourier transform. The simplicity of the method lends itself to easy programming.

2. Notation and preliminaries. Let the domain D be bounded by the curve $\gamma(\sigma) \in \mathbb{C}$, $0 \le \sigma \le L$, as described in the introduction. For each $\theta \in [0, 2\pi]$ there is $\sigma(\theta) \in [0, L]$ such that $f(e^{i\theta}) = \gamma(\sigma(\theta))$. We can assume that the real-valued function $\sigma(\theta) - \theta$ is continuous and L-periodic. We do not require γ to be parametrized according to arc length. We will need to be able to calculate the projection $\rho_\gamma(w)$ to the boundary ∂D for points w near ∂D. For this it may be helpful to have the complex derivative $\gamma' \neq 0$ available computationally. This would be the case, for instance, if points along the boundary are given numerically and γ is obtained by cubic spline interpolation [3]. As an alternative, one may work with ∂D defined implicitly by an equation $F(x, y) = 0$ and then will need to make only minor adjustments in the following, eliminating explicit mention of γ and σ.

Let the number $N > 1$ of grid points be fixed (normally a power of 2 for efficiency of calulation) and write $\zeta = \zeta_N = \exp(2\pi i/N)$. Given $0 = \sigma_0 < \sigma_1 < \cdots < \sigma_{N-1} < L$, set $w_j = \gamma(\sigma_j)$ and abbreviate $\vec{\zeta} = (\zeta^j) = (1, \zeta, \ldots, \zeta^{N-1})$, $\vec{w} = (w_j) = (w_0, w_1, \ldots, w_{N-1}) \in \mathbb{C}^N$. The interpolating polynomial taking the Nth root of unity $\zeta^j \in \partial D_0$ to w_j and fixing the origin can be expressed as

$$P_{\vec{w}}(z) = \sum_{j=0}^{N-1} Q(\zeta^{-j} z) w_j, \tag{2.1}$$

where

$$Q(z) = \frac{z}{N}\frac{1-z^N}{1-z} \;=\; \frac{1}{N}(z+z^2+\cdots+z^N)$$

is the unique polynomial of degree N satisfying $Q(0)=0$, $Q(1)=1$, $Q(\zeta^j)=0$ for $1 \le j \le N-1$. (Observe that P is not identical to the trigonometric interpolation polynomial as in [8], [9], often given in barycentric form, and which does not fix the origin.)

We will follow the convention in [13] of writing scalar functions applied elementwise to N-tuples when there is no cause for confusion. Thus $P_{\vec{w}}$ is characterized among polynomials of degree $\le N$ fixing the origin by the property $P_{\vec{w}}(\vec{\zeta}) = \vec{w}$.

2.1. Rotation of an *N*-tuple. Fix $\beta \in \mathbb{R}$ and define the "rotated" N-tuple $\vec{w}^{[\beta]} \in \mathbb{C}^N$ by

$$(\vec{w}^{[\beta]})_j = P_{\vec{w}}(\zeta^{j+\beta}) = \sum_k Q(\zeta^{j-k+\beta})w_k = \sum_k R^{[\beta]}_{jk} w_k,$$

where $R^{[\beta]}_{jk} = r^{[\beta]}_{j-k}$ and

$$r^{[\beta]}_j = Q(\zeta^{j+\beta}) = \frac{1}{N}e^{\pi i\beta}(\sin\pi\beta)\left(\cot\left(\frac{\pi}{N}(j+\beta)\right)+i\right) \; . \tag{2.2}$$

(We will always understand subindices of elements of $\mathbb{C}^N$ to be taken modulo N.) Thus $R^{[\beta]} = (R^{[\beta]}_{jk})$ is a *circulant matrix* [2]; that is, its columns are $\vec{r}^{[\beta]}$, $E\vec{r}^{[\beta]}$, $E^2\vec{r}^{[\beta]}, \ldots$, where the shift operator $E : \mathbb{C}^N \to \mathbb{C}^N$ moves the elements cyclically one space forward. The point of these definitions is that we can express

$$\vec{w}^{[\beta]} = R^{[\beta]}\vec{w} \; . \tag{2.3}$$

Since $\vec{w}^{[\beta]}$ can also be interpreted as the convolution $\vec{r}^{[\beta]} * \vec{w}$, we know that it can be evaluated by a fast fourier transform (FFT) (see [8], [9]) implying a computational cost of $O(N\log N)$ multiplications.

The matrix $R^{[\beta]}$ is a smooth function of β of period N. Since the function $z \mapsto P_{\vec{w}}(\zeta^\beta z)$ sends $\vec{\zeta}$ to $\vec{w}^{[\beta]}$, by the characterization of our interpolating polynomials we have $P_{R^{[\beta]}\vec{w}}(z) = P_{\vec{w}}(\zeta^\beta z)$ for all β, $\vec{w}$, and z. Applying this in particular with $\beta+\beta'$ in place of β yields the matrix relation

$$R^{[\beta]}R^{[\beta']} = R^{[\beta+\beta']} \; .$$

2.2. Half-click matrix *C*. We shall mainly be interested in the special value $\beta = 1/2$ and make use of the constants

$$c_j = r^{[1/2]}_j = \frac{1}{N}\left(-1 + i\cot\frac{\pi}{N}\left(j+\frac{1}{2}\right)\right) \tag{2.4}$$

and $\vec{c} = (c_j)$, $C = R^{[1/2]}$. From the symmetry $\bar{c}_j = c_{-(j+1)}$, in other words, $\bar{C} = EC^T$, and the fact that $C^2 = R^{[1]} = E$, it follows that C is unitary:

$$C^{-1} = \bar{C}^T \; . \tag{2.5}$$

In summary, we see that given the values $\vec{w}$ of a normalized interpolating polynomial at the points of $\vec{\zeta}$ via a single matrix multiplication (or FFT) we obtain the values $C\vec{w}$ of the same polynomial at the intermediate points $\zeta^{1/2}\vec{\zeta} = (\zeta^{1/2}, \zeta^{3/2}, \ldots, \zeta^{N-1/2})$.

2.3. Boundary projection. For the present method, the specific algorithm for approximating the projection $\rho_\gamma(w)$ of the point w onto the curve $\gamma = \partial D$ is of little importance. For most examples in which ∂D is parametrized as $\gamma(\sigma)$, we have proceeded as follows. Note that $\rho_\gamma(w)$ is defined if γ is sufficiently smooth and if w is sufficiently close to ∂D, i.e., if $\vec{w}$ lies in a neighborhood of the N-torus $(\partial D)^N \subseteq \mathbb{C}^N$. The boundary curve is covered by a fixed number of small disks, and any point lying outside of all of these disks is simply mapped to the closest of the centers. For a point w inside one of these disks, centered at, say, $\gamma(s_0)$, first w is projected orthogonally to a point $\hat{w}$ lying on the tangent line to γ at $\gamma(s_0)$. This determines the approximate amount $\Delta s = |\gamma(s_0) - \hat{w}|/\gamma'(s_0)$ by which s_0 should be incremented. One uses $s_0 + \Delta s$ as the new s_0 and $\hat{w}$ as the new w. This process, when repeated, converges quadratically to a point $\rho_\gamma(w)$ on ∂D.

3. Description of the iterative method. With the preliminary material in hand, the algorithm is quite simple to describe. Consider "guess" values σ_j^* and write $w_j^* = \gamma(\sigma_j^*)$. If these values were correct, i.e., if $w_j^* = w_j$, then by the result of [13] cited above, we may expect (if N is large) that all values of $P_{\vec{w}^*}(e^{i\theta})$ lie on or close to ∂D. This will be true in particular of the values $u_j^* = P_{\vec{w}^*}(\zeta^{j+1/2})$, i.e., for

$$\vec{u}^* = P_{\vec{w}^*}(\zeta^{1/2}\vec{\zeta}) = C\vec{w}^* \ . \tag{3.1}$$

On the other hand, if $\vec{w}^*$ is badly placed, then $P_{\vec{w}^*}(\partial D_0)$ may stray quite far from ∂D, as illustrated in Figure 1. Consider the set Σ_N of N-tuples $\vec{w} \in (\partial D)^N$ such that $\vec{u} = C\vec{w} \in (\partial D)^N$ and such that the interleaved points $w_0, u_0, w_1, u_1 \ldots, w_{N-1}, u_{N-1}$ lie in cyclic order, positively oriented, along ∂D. This is a topological circle lying as "diagonal" in $(\partial D)^N$, and we consider it to be the set of solutions to the (nonnormalized) mapping problem. The composition

$$\Phi_\gamma = \rho_\gamma \circ C^{-1} \circ \rho_\gamma \circ C \tag{3.2}$$

is defined on a neighborhood of $(\partial D)^N$ and fixes all elements of Σ_N. Conversely, as long as the interleaved values of $\vec{w}$ and $\vec{u}$ lie in cyclic order, and if D is such that the reasoning in section 5 applies, then any fixed point $\vec{w} \in (\partial D)^N$ of Φ in fact lies on Σ_N and thus is a solution.

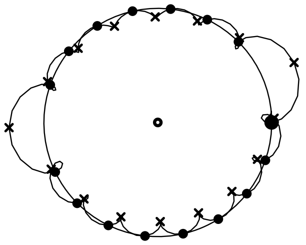


FIG. 1. *$P_{\vec{w}}(\partial D_0)$, where D is the unit disk, $N = 16$, and the values σ_j are displaced from the grid values $2\pi j/N$ by reducing $\sigma_{j+1} - \sigma_j$ by 10% for $j \neq 0, 7$ and increasing the remaining two intervals accordingly. The target points $w_j = e^{2\pi i\sigma_j}$ are marked with a spot (w_0 slightly larger), the origin with a hollow dot, and the intermediate values $u_j = P_{\vec{w}}(\zeta_N^{j+(1/2)})$ with an "x." Note how the increased space between the w_j tends to push u_j outwards, whereas decreasing the space moves it inwards.*

These considerations suggest the following algorithm: *beginning with $\vec{\sigma}^*$, define $\vec{w}^* = \gamma(\vec{\sigma}^*)$ and calculate the successive approximations*

$$(\Phi_\gamma)^n(\vec{w}^*)\ . \tag{3.3}$$

As mentioned in the introduction, the approximation of the mapping f via (3.3) invoves two types of truncation error: first, the discrepancy remaining between the nth iterate and its limiting value $\vec{w}$ (assuming convergence exists), and, second, the discrepancy between the best possible degree N polynomial approximation $P_{\vec{w}} \in \Sigma_N$ and f itself (i.e., our notion of "solution" Σ_N may not be adequate for a given problem).

3.1. Doubling. For many irregular domains, the mapping may be difficult to calculate because if one begins with an initial guess w_j^* too far from the correct values, the u_j^* may project to the boundary in the wrong order and the iteration will not approach the true mapping function. Often the difficulty can be avoided by beginning with a smaller value of N, applying Φ_γ one or more times, and then working with $2N$ points. Happily, we obtain the doubling at no cost, since the most recently calculated values of $\rho_\gamma(u_j)$, arrived at after half of the calculations involved in (3.2), can simply be interleaved in between those of w_j; that is, $\vec{w}$ is replaced by the "doubled" vector $(w_0, \rho_\gamma(u_0), w_1, \rho_\gamma(u_1), \ldots) \in \mathbb{C}^N$.

4. Normalization.

Repeated application of the operator Φ_γ of (3.2) does not need to conserve any reasonable normalization of the polynomials as mapping functions. We describe here two ways of obtaining normalized solutions to the mapping problem.

4.1. Boundary point normalization. Here we consider the normalization $f(1) = b \in \partial D$ (equivalently, $\sigma_0 = 0$ where $b = \gamma(0)$). Rather than complicate the iterative procedure by trying to make it produce $\Delta\vec{w} = 0$ automatically, we may rotate the vector $\vec{w}$ back to a normalized position via the operator $R^{[\beta]}$ of (2.3). The equation determining the appropriate value of β is nonlinear, hence it is convenient to use the following approximation procedure.

Consider the quadratic polynomial $q(t)$ determined by

$$q(-1) = w_{-1}, \quad q(0) = w_0, \quad q(1) = w_1.$$

There are in general two values t^+, t^- for which $q(t^\pm) = b$. They need not be real numbers; however, if all four points w_{-1}, w_0, w_1, b are close together on ∂D and if this boundary is reasonably smooth, there will be a unique root close to the real interval $[-1, 1]$. This leads us to define

$$\beta^* = \operatorname{Re} \frac{4(b - w_0)}{(w_1 - w_{-1}) \pm \sqrt{(w_1 - w_{-1})^2 + 8(b - w_0)(w_1 - 2w_0 + w_{-1})}}$$

(choosing the smaller of the two values) and then to replace $\vec{w}$ by $\rho_\gamma R^{[-\beta^*]}\vec{w}$. We repeat this until w_0 is sufficiently close to b. The limit is $\Psi_{\gamma,b}\vec{w}$, where we define

$$\Psi_{\gamma,b} = \rho_\gamma R^{[-\beta]} \tag{4.1}$$

for an appropriately chosen β, so that by construction $(\Psi_{\gamma,b}\vec{w})_0 = b$. There is no difficulty in implementing it computationally; the convergence of this operaton is quadratic.

This "correction by rotation" may be applied after each iteration, after every few iterations, or at the end. If the value w_0 has been allowed to wander very far, a preliminary applicaton of a power E^n of the shift operator, i.e., a cyclic rearrangement of the w_j, may be necessary as a preliminary step so that w_0 is as close to a as possible.

4.2. Normalized first derivative. The other normalization frequently considered, of interest especially for theoretical considerations, is $f'(0) > 0$. Writing $P'_{\vec{w}}(0) = \sum_j w_j Q'(0) = \sum w_j = a$, we see that $f(z) = P_{\vec{w}}((|a|/a)z)$ satisfies this normalization. The new grid values $f(\vec{\zeta})$ are thus obtained from $\vec{w} = P_{\vec{w}}(\vec{\zeta})$ as $\rho_\gamma R^{[-\beta]}\vec{w}$, where $\beta = (N/2\pi)\arg a$. It may be necessary to repeat the process until the desired accuracy is obtained.

5. Justification of convergence. Here we give an analysis, partly heuristic, of the effect on $\Phi_\gamma(\vec{w})$ caused by a small change $\Delta\vec{w}$ in $\vec{w}$. This will enable us to conclude that under reasonable conditions the iterative procedure defined in section 3 converges.

Suppose $\vec{w}^*$ is near the solution set Σ_N, and let $\vec{w} \in \Sigma_N$ be the closest element to $\vec{w}^*$ in the sense of minimizing the L_2 norm $\|\vec{w} - \vec{w}^*\|_2 = (\sum |w_j - w_j^*|^2)^{1/2}$. Write $\vec{w}^* = \vec{w} + \Delta\vec{w}$, $\vec{u} = C\vec{w}$, $\vec{u}^* = C\vec{w}^* = \vec{u} + \Delta\vec{u}$. By linearity, $\Delta\vec{u} = C\Delta\vec{w}$, and since C is unitary,

$$\|\Delta\vec{u}\|_2 = \|\Delta\vec{w}\|_2 \ . \tag{5.1}$$

Recall that $\rho_\gamma(u_j) = u_j$ by the definition of Σ_N. Write

$$\rho_\gamma\vec{u}^* = \vec{u} + \vec{d} \ ; \tag{5.2}$$

that is, d_j is the projection of the complex quantity Δu_j parallel to the tangent direction of ∂D at w_j.

We will suppose that N is so large that ∂D is approximated by a straight segment from $w_{j-\sqrt{N}}$ to $w_{j+\sqrt{N}}$. Recall that c_j is given by (2.4). It will be convenient to subdivide the indices from $-N/2$ to $N/2$ into the following sets:

$$\begin{aligned} &I_1 : 0 \le |j| < \sqrt{N} \ , \\ &I_2 : \sqrt{N} \le |j| < N/4 \ , \\ &I_3 : N/4 \le |j| \le N/2 \ . \end{aligned}$$

We may use as approximations the values

$$\begin{aligned} c_0 &= \frac{-1}{N} + \frac{2}{\pi}i + O(N^{-2}) \ , \\ c_{\sqrt{N}} &= \frac{-1}{N} + \frac{1}{\pi}\left(\frac{1}{\sqrt{N}} - \frac{1}{2N}\right)i + O(N^{-3/2}) \ , \\ c_{N/4} &= \frac{-1}{N} + \frac{i}{N} + O(N^{-2}) \ , \\ c_{N/2} &= \frac{-1}{N} + O(N^{-2}) \ . \end{aligned}$$

For our purposes it will be sufficient to obtain the remaining c_j from these values by linear interpolation over the corresponding j-intervals. With this approximation one finds that

$$\sum_{I_1} c_j = \frac{-2}{\sqrt{N}} + \frac{1}{\pi}\left(2\sqrt{N} + 1 - \frac{1}{2\sqrt{N}}\right) i + O(N^{-1}) \ ,$$

$$\sum_{I_2} c_j = \frac{-1}{2} + \frac{1}{8\pi}(2\sqrt{N} + 2\pi - 9)i + O(N^{-1/2}) \ ,$$

$$\sum_{I_3} c_j = \frac{-1}{2} + \frac{1}{4}i + O(N^{-1}) \ .$$

We will fix $j = 0$ for the moment for clarity. Thus $\Delta u_0 = \sum_{k=-N/2}^{N/2} c_{-k}\Delta w_k$. For $|k|$ near 0 modulo N, Δw_k is approximately parallel to Δw_0; there is no loss in generality in supposing that this direction is horizontal. We make the simplifying assumption that all the Δw_k are approximately the same size, say, $|\Delta w_k| = \delta$. Under this assumption we have

$$\frac{1}{\delta}\,|\Delta u_0| \geq \left|\sum_{I_1} c_{-k}\right| - \frac{1}{\delta}\left|\sum_{I_2 \cup I_3} c_{-k}\Delta w_k\right| \geq \frac{7}{4\pi}\sqrt{N} + O(1) \ .$$

Now we estimate $d_0 = \text{Re } \Delta u_0 = \sum_{k=-N/2}^{N/2} \text{Re } (c_{-k}\Delta w_k)$. For $k \in I_1$ we have

$$\frac{1}{\delta}\text{Re } \sum_{I_1} c_{-k}\Delta w_k = \frac{-2}{\sqrt{N}} + O(N^{-2})$$

from the assumption $\Delta w_k \in \mathbb{R}$. However, over I_2 and I_3 the directions of Δw_k may upset the cancellation in the sum $\sum c_j$ estimated earlier. We can at least say

$$\frac{1}{\delta}|d_0| \leq \frac{2}{\sqrt{N}} + \frac{1}{\delta}\left|\sum_{I_2 \cup I_3} c_{-k}\Delta w_k\right| \leq \frac{1}{4\pi}\sqrt{N} + O(1) \ .$$

(The dominant contribution of order $\sqrt{N}$ in these estimates comes from I_2, where there is likely to be a large amount of cancellation which has not been taken into account. This cancellation would be favorable to our estimates. On the other hand, cancellation in I_1, caused by Δw_k in opposing directions, would be unfavorable.)

Since the reasoning applies equally well for all j, we conclude from these upper and lower bounds (equating $|w_j|$ again with δ) that

$$|d_j| \leq A|\Delta u_j| \tag{5.3}$$

for any $A > 1/7$ if N is sufficiently large. In particular, $\|\vec{d}\|_2 \leq A\|\Delta\vec{u}\|_2$. Using (5.1), we may express this as

$$\|\rho_\gamma C\vec{w}^* - \vec{u}\|_2 \leq A\|\vec{w}^* - \vec{w}\|_2 \ ; \tag{5.4}$$

i.e., the operator $\rho_\gamma C$ reduces the L_2-distance from vectors to Σ_N by a factor less than 1. The same is evidently true of the operator $\rho_\gamma C^{-1}$, and hence by (3.2) of Φ_γ as well.

The above reasoning depends on several assumptions regarding the distribution of the w_j, their relative sizes, the value of N, etc. To the extent that these assumptions are valid, we infer that the distance of the sequence $\{\Phi_\gamma^n(\vec{w}^*)\}$ to the set Σ_N tends to zero. We can say more. Note that in fact

$$\Phi_\gamma(\vec{w}^*) = \rho_\gamma(\vec{w} + C^{-1}\vec{d}) .$$

Since projection onto ∂D does not increase distance,

$$\begin{aligned} &(5.5) \qquad & |\Phi_\gamma(\vec{w}^*)_j - w_j| &< |(C^{-1}d)_j| \le \|C^{-1}d\|_2 \\ &(5.6) \qquad & &\le A\|\Delta\vec{w}\|_2. \end{aligned}$$

We have argued that $\|\Delta\vec{w}\|_2$ decreases geometrically over successive iterations. Therefore the w_j form a Cauchy sequence as $n \to \infty$, and it follows that $\Phi_\gamma^n(w^*)$ tends in fact to an element of Σ_N.

6. Computational results. The numerical experiments reported here were carried out on a SPARC 10 workstation, programmed in C and in Mathematica. In order to take advantage of the FFT algorithm, N is always taken to be a power of 2. From the experiments made, the convergence is linear. For convenience we report the changes of values in the max norm $\|\vec{v}\|_\infty = \max|v_j|$; observations with the L_2 norm produced no significant differences. Where an explicit formula for the mapping function is available, we have used this data in the comparisons. Since this does not hold in most practical problems, we comment first on ways to ensure that the results obtained are reasonable. Mere convergence of $\vec{w}^{(n)} = \Phi^n(\vec{w}^*)$ to $\vec{w}$ is of course insufficient.

Observe that in the context we are working, the values w_j represent most naturally not the polygon with these points as vertices but rather the image of the unit circle under the polynomial $P_{\vec{w}}$. This image curve can be calculated exactly using the rotation operator $R^{[\beta]}$ of (2.3). We have done this for all of the figures presented in this article, choosing a number M $(= 10)$ of subdivisions for each of the N subarcs and applying the rotation matrices $R^{[m/M]}$ to $\vec{w}$ for $m = 1, \dots, M-1$. The elements of the resulting M N-tuples were then interleaved and then graphed.

Suppose $\vec{w}^{(n)} \to \vec{w} \in (\partial D)^N$. Whenever there is evidence that the image curves $P_{\vec{w}^{(n)}}(\partial D_0)$ tend to ∂D (and wrap only once around the domain), it follows from standard reasoning using the argument principle that the holomorphic functions $P_{\vec{w}^{(n)}}$ in fact converge to a normalized univalent polynomial P. Since clearly $P(\vec{\zeta}) = \vec{w}$, we have $P = P_{\vec{w}}$. As an alternative to checking graphically, one can examine the norm $\|(\vec{w}^{(n)})^{[\beta]} - \vec{w}^{[\beta]}\|_\infty$ of the discrepancies of the rotated points. At any rate, it is sufficent to check convergence of the half-click points $\vec{u}^{(n)}$ to verify that the iterations are approximating a point of Σ_N; we have marked these points in the figures with an "x".

Example 1. The disk $D = |w - \alpha| < 1$ admits as conformal mapping the normalized Möbius transformation $f(z) = \alpha - (\alpha - z)/(1 - \alpha z)$. Let $\alpha = 0.8$. We have applied successive doubling of points from $N = 4$ up to $N = 32$ after a single application of Φ_γ for each N and then continued to iterate from that point with N fixed. Table 1 shows the changes observed in $\vec{w}$ in the norm $\|\ \|_\infty$ (these are roughly proportional to the changes in σ) to each iteration from the previous one. For comparison, the differences between the approximations and the true values $w^\infty = f(z)$ are given in the rightmost column. These are also seen to decrease geometrically.

TABLE 1

Changes in $\vec{w}$ and discrepancy from true value for mapping to shifted disk, $\alpha = .8$, $N = 32$.

Iteration	$\|\Delta\vec{w}\|_\infty$	$\|\vec{w} - \vec{w}^\infty\|_\infty$
4	1.4	1.2
8	.22	.22
16	.21	.26
32	.098	.14
32	.027	.044
32	.011	.018
32	.0044	.0067
32	.0018	.0023
32	.00074	.0005

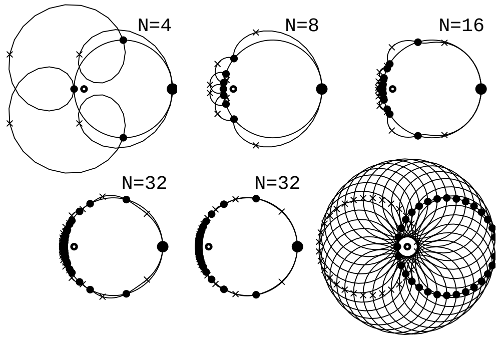


FIG. 2. *Successive doubling of grid points for mapping to a shifted disk. We begin with the* 4*th roots of unity as the initial guess. For $N = 4, 8, 16$ (top row) and* 32 *(bottom row, left) the image of the unit circle is shown after a single application of Φ_γ. The half-click points are interleaved into the N-tuple obtained, thus effectively doubling N for the next step. For $N = 32$ we also show the image after the second iteration. Convergence would not be obtained if the initial guess were the* 32*nd roots of unity; the last image depicts the corresponding image boundary.*

The images of ∂D_0 under the interpolating polynomials are drawn for the first five iterations in Figure 2. Note that for $N = 32$ the interpolating polynomial gives quite a good approximation even near the right-hand side of the domain, where extremely few grid points are mapped due to the crowding phenomenon.

Example 2. We consider the inverted ellipse with mapping function $f(z) = 2\alpha z/((1+\alpha) - (1-\alpha)z^2)$; the image domain is bounded by the curve $\gamma(\sigma) = (e^{i\sigma}(1 - (1-\alpha^2)\sin^2\sigma))^{1/2}$, $0 \le \sigma \le 2\pi$. We will use $\alpha = .25$.

Starting with the Nth roots of unity, $N = 32$, iteration of (3.2) yields the results given in Table 2. The images of ∂D_0 under some of the corresponding interpolating polynomials are shown in Figure 3. After 14 iterations we found the value of the first coordinate σ_0 to be less than 10^{-11}, which indicates that there has essentially been no wandering of the base point w_0, even though no normalization was applied. This was to be expected due to the symmetry of the domain. Upon further iteration, we found that while the successive differences $\|\Delta\vec{w}\|_\infty$ continued to manifest linear convergence, around the 20th iterate the true discrepancy $\|\vec{w} - \vec{w}^\infty\|_\infty$ stabilized at around 10^{-4}. This clearly reflects the fact that a polynomial of degree $N = 32$ cannot

TABLE 2
Accuracy measurements for inverted ellipse, $\alpha = .25$, $N = 32$.

Iteration	$\|\Delta\vec{w}\|_\infty$	$\|\vec{w} - \vec{w}^\infty\|_\infty$
1	.38	.58
2	.071	.23
3	.046	.16
4	.031	.11
5	.022	.081
6	.017	.058
7	.012	.042
8	.0094	.03
9	.0071	.021
10	.005	.014
11	.0033	.0085
12	.0021	.0052
13	.0013	.0032
14	.00079	.0019

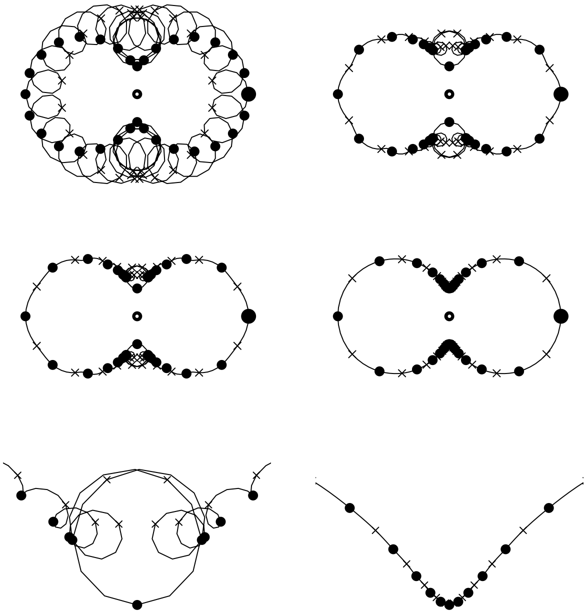


FIG. 3. *Image curves for inverted ellipse of Example* 2*: iterations* 1*,* 2*,* 3*, and* 14 *(top two rows). Magnifications of the middle row near the point $w = f(i)$ (bottom row) shows how the algorithm has straightened out "knotting" of the boundary.*

approximate this domain any better. When we doubled the grid size (using the $\vec{w}$ just obtained), the discrepancy was reduced to about 10^{-8} (see Figure 4). We have observed this type of improvement of the absolute accuracy with increased N in all examples tested.

Example 3. We give a nondifferentiable example, the square with vertices at $\pm 1 \pm i$. Here N is taken to be 32, and the original points are equally spaced along the perimeter. For the projection ρ_γ we have preferred to use an obvious ad-hoc function taking advantage of the easy geometry. The true mapping is given by an elliptic integral, $f(z) = az\int_0^1 (1 + r^4 z^4)^{-1/2} dr$, where $a > 0$ is chosen so that $f(1) = 1$. It can be seen from Figure 5 that a good approximation at the corners requires, apart from reducing $\|\Delta\vec{w}\|_\infty$, a large number of grid points. As Table 3 confirms, the approximation of f is not improved by iterating closer to Σ_N for fixed N.

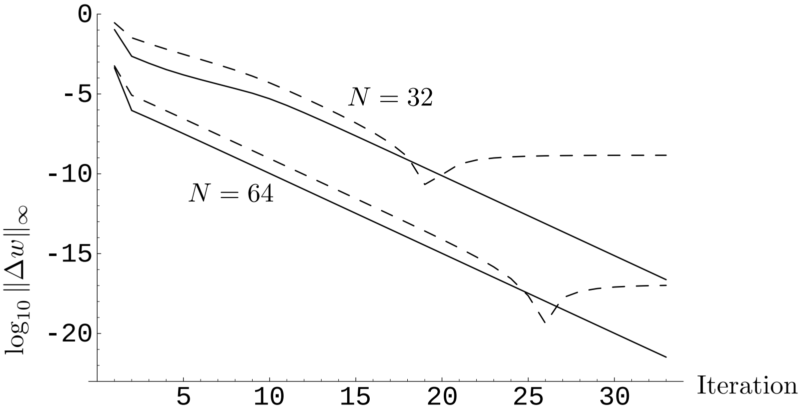


FIG. 4. *Comparison between successive differences (solid lines) and discrepancy from true value (dotted lines) for the inverted ellipse of Example* 2. *With $N = 32$ grid points, after $n = 20$ the mappings corresponding to $\Phi_\gamma^n(\vec{w}^*)$ no longer approach the true f but rather continue to tend towards the best possible approximation by a degree-N polynomial. For $N = 64$ the polynomials approximate f several orders of magnitude more closely.*

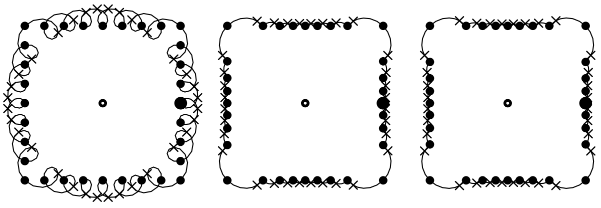


FIG. 5. *Polynomial approximation of conformal mapping to square domain. The rounding of the image of ∂D at the corners is not improved even as the iterates converge to a limiting value, as the number $N = 32$ of grid points is insufficient.*

Example 4. The last example we study is taken from Fornberg [6]. The boundary of D is defined to be the set of points w such that $F_\alpha(w) = 0$, where

$$(6.1)\ F_\alpha(w) = ((\mathrm{Re}\, w - .5)^2 + (\mathrm{Im}\, w - \alpha)^2)(1 - (\mathrm{Re}\, w - .5)^2 - (\mathrm{Im}\, w)^2) - 0.1\ .$$

In this example no parametrization of the boundary curve is explicitly needed; the projection to the boundary was carried out by Newton's method, that is, replacing w by $w - F(w)|\nabla F(w)|^{-2}$ where $\nabla F(w) \in \mathbb{C}$ is the gradient of F.

For α large, D approximates the disk $|w - .5| = 1$; see Figure 6. A more detailed depiction may be found in [6]. For $\alpha = .5$ the crowding phenomenon has reached the stage where the distance between consecutive grid points at the upper right edge of the domain is more than 500 times as large as at the upper left. One consequence of this is that an initial guess must be rather close to the sought-after values in order for the algorithm to converge.

To explain the comparative chart given in Table 4, it is necessary to summarize a few basic features of Fornberg's method. The initial guess is used to calculate a matrix and a vector; this step is referred to as an "outer iteration." Given this data, a series of "inner iterations" is carried out to arrive at essentially what we have called

Table 3
Approximations of square, $N = 32$.

Iteration	$\|\Delta\vec{w}\|_\infty$	$\|\vec{w} - \vec{w}^\infty\|_\infty$
1	.21	0.23
2	.0089	0.022
3	.00033	0.013
4	.000012	0.012
5	$.44\times10^{-6}$	0.012
6	$.16\times10^{-7}$	0.012
7	$.58\times10^{-9}$	0.012
8	$.21\times10^{-10}$	0.012
9	$.78\times10^{-12}$	0.012

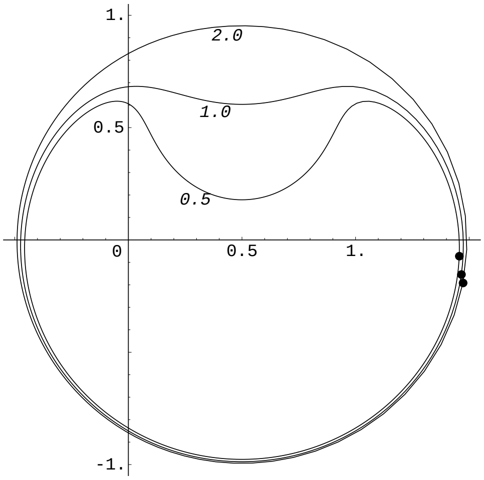


Fig. 6. *The domains defined by* (6.1) *for* $\alpha = 2.0, 1.0, .5$. *The number* N *of grid points is given in Table* 4. *The vast majority of the image points are concentrated in the upper left-hand area. Note the wandering of* w_0 *(marked by a spot near the right) allowed by the nonnormalizing algorithm.*

here $\Delta\vec{w}$. According to [6], the cost of one outer iteration with K inner iterations is $7 + 4K$ FFTs, whereas the cost of our Φ_γ is 4 FFTs.

The strategy in Fornberg's calculation, which we have applied here as well, is to let the values of α decrease gradually. We have also replicated the strategy of applying two Newton iterations to bring the points closer to the curve for each new α and have used a single Newton iteration as an approximation of ρ_γ in calculating Φ_γ. To compensate for the increased crowding with decreasing α, at certain moments the number of grid points is doubled (recall section 3). In [6] the doubling was done by a fourth-order polynomial approximation, requiring an additional type of calculation as compared to the present method.

We have found that under the above conditions, our method does not jump well from $\alpha = \infty$ to $\alpha = 2.0$ with $N = 128$ grid points. In [6] this was accomplished, starting from 128 equidistant points, in 11 outer iterations, corresponding to 225 FFTs. Instead we have doubled repeatedly from $N = 4$ to $N = 128$, utilizing 44 FFTs, and afterwards copied the decrements and doublings in [6]. The results are gathered in Table 4. To facilitate the comparison, we show the number of FFTs (each Φ_γ is 4 FFTs) used at each step.

Table 4

Comparative results for domain given by (6.1)*. Increments of the domain parameter α are paralleled by doubling of grid size N as in* [6]*; the cost in FFTs for the changes $\|\Delta\vec{w}\|_\infty$ and "Accuracy of Taylor coefficients" is shown for the two methods. The change $\|\Delta\vec{a}\|_\infty$ of the Taylor coefficients recalculated from the output of our method is also reported.*

		Iterating Φ_γ:			Method of [6]:	
α	N	FFTs	$\|\Delta\vec{w}\|_\infty$	$\|\Delta\vec{a}\|_\infty$	FFTs	Accuracy of Taylor coefs.
2.0	4	16	$.48\times10^{-1}$			
2.0	8	4	$.89\times10^{-1}$			
2.0	16	4	$.16\times10^{-1}$			
2.0	32	4	$.18\times10^{-2}$			
2.0	64	8	$.91\times10^{-5}$			
2.0	128	0	—			
1.5	128	24	$.13\times10^{-8}$	$.70\times10^{-9}$	96	$.14\times10^{-7}$
1.2	128	36	$.45\times10^{-8}$	$.37\times10^{-9}$	100	$.33\times10^{-5}$
1.2	256	0	—		50	$.97\times10^{-8}$
1.0	256	48	$.68\times10^{-7}$	$.63\times10^{-8}$	108	$.17\times10^{-5}$
1.0	512	0	—		50	$.40\times10^{-8}$
.9	512	44	$.53\times10^{-7}$	$.82\times10^{-8}$	116	$.26\times10^{-6}$
.9	1024	0	—		89	$.98\times10^{-10}$
.8	1024	80	$.35\times10^{-8}$	$.33\times10^{-9}$	155	$.42\times10^{-7}$
.8	2048	0	—		81	$.55\times10^{-11}$
.75	2048	112	$.41\times10^{-9}$	$.37\times10^{-10}$	143	$.58\times10^{-9}$
.72	2048	108	$.40\times10^{-8}$	$.35\times10^{-9}$	151	$.40\times10^{-8}$
.7	2048	108	$.95\times10^{-8}$	$.81\times10^{-9}$	120	$.30\times10^{-7}$
.7	4096	0	—		81	$.31\times10^{-11}$
.68	4096	148	$.12\times10^{-9}$	$.97\times10^{-11}$	151	$.38\times10^{-10}$
.66	4096	148	$.53\times10^{-9}$	$.42\times10^{-10}$	151	$.13\times10^{-9}$
.64	4096	136	$.92\times10^{-8}$	$.69\times10^{-9}$	159	$.29\times10^{-8}$
.62	4096	136	$.33\times10^{-7}$	$.23\times10^{-8}$	140	$.19\times10^{-7}$
.6	4096	144	$.48\times10^{-7}$	$.33\times10^{-8}$	148	$.84\times10^{-7}$
.6	8192	0	—		93	$.23\times10^{-10}$
.58	8192	168	$.19\times10^{-7}$	$.12\times10^{-8}$	167	$.78\times10^{-9}$
.56	8192	188	$.14\times10^{-7}$	$.83\times10^{-9}$	171	$.90\times10^{-9}$
.54	8192	180	$.11\times10^{-6}$	$.63\times10^{-8}$	171	$.22\times10^{-7}$
.54	16384	0	—		77	$.28\times10^{-10}$
.52	16384	180	$.46\times10^{-6}$	$.23\times10^{-7}$	178	$.99\times10^{-9}$
.5	16384	212	$.25\times10^{-6}$	$.11\times10^{-7}$	228	$.15\times10^{-7}$

Several observations are in order regarding the accuracy obtained. As in the previous examples, we have measured the error given for $\vec{w}$ as the largest change in w_j in the last iteration of Φ_γ calculated. The error reported in [6] is in terms of the Taylor coefficients, which is what that method produces directly; these range from 10^{-7} to 10^{-11} approximately for the values of α considered here. In each row of Table 4 from $\alpha = 1.5$ on, to provide an additional comparison we calculated at the beginning of the last iteration of Φ_γ the (Taylor) coefficients $\vec{a}^* = (a_1^*, \ldots, a_N^*)$ of $P_{\vec{w}^*}$ from the values obtained for $\vec{w}^*$. This was accomplished by applying the inverse Fourier transform and shifting to the left (not included in the FFT count in the table). After the iteration we similarly calculated the corresponding Taylor coefficients $\vec{a}$ and have included in Table 4 the maximal difference of $a_j - a_j^*$. However, it must be noted that comparison of this data with that of [6] is complicated by the fact that the results given there correspond only to the change caused by the "outer iterations." It may also be noted that an error of 10^{-7} in each of $N = 16{,}384$ coefficients of a polynomial evaluated for $|z| = 1$ may induce an error of as much as 10^{-3} in w^j, regardless of the algorithm used to evaluate the polynomial. Since in these calculations we have not

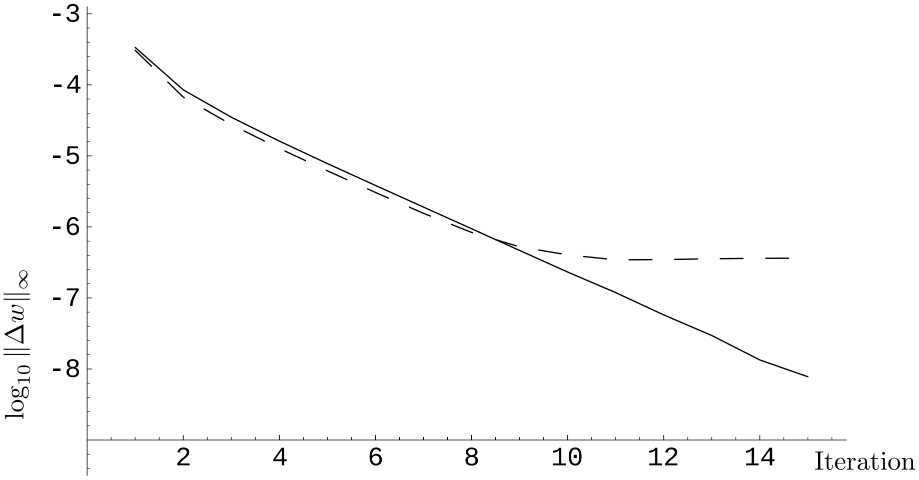


FIG. 7. *Comparison of decreasing of Δw for methods based on Φ_γ (solid line) and $\Psi_{\gamma,b}\Phi_\gamma$ (dotted line). Note that the renormalization by the rotation $\Psi_{\gamma,b}$ has not increased the accuracy of convergence for the first* 8 *rotations, but after that point it has impeded loss of convergence coming from floating of the base point.*

applied any normalization as described in section 4, w_0 is allowed to "float" along the boundary; its position is marked by spots near the right-hand side of Figure 6.

One may ask to what extent the changes observed in $\vec{w}$ may be due to this floating of the base point, which could thus mask how fast $\vec{w}$ is really approaching the set of solutions Σ_N. In another experiment, we took the 2048-point vector $\vec{w}$ corresponding to $\alpha = 0.75$ in Table 4 and constructed a new initial guess $\vec{w}^*$ by sampling from it $N = 256$ points defined by $w_j^* = w_{(8j)}$. When Φ_γ was applied, the differences $\Delta\vec{w}$ ceased to decrease at about 10^{-6}. Then we applied to the initial $\vec{w}^*$ the algorithm defined by $\Psi_{\gamma,b}\Phi_\gamma$. (After each Φ_γ, a single b was calculated as described in 4.1 to rotate the initial boundary point approximately to where it had been.) Now differences as small as 10^{-9} were obtained in the same number of iterations, as shown in Figure 7. Thus we see that in the Φ_γ algorithm, the floating base point phenomenon has destroyed convergence, which is recovered by renormalizing in each iteration. To discount the possibility that the sequence obtained with $\Psi_{\gamma,b}$ had better convergence merely due to a contribution of the additional projection involved in this operation, in another test several extra iterations of Newton's method were added to Φ_γ; these made no significant difference.

In contrast, in several other tests, in particular with the examples given in this paper, the $\Psi_{\gamma,b}\Phi_\gamma$ method produced little or no acceleration of convergence with indeed a higher cost in terms of FFTs.

7. Closing remarks. While we have studied exclusively the algorithm obtained by iterating the operator Φ_γ (with possible renormalization), we wish to suggest that the interpolating polynomial $P_{\vec{w}}$ is an object of intrinsic interest. Possibly more sophisticated algorithms could be developed, based on the ideas presented here, to obtain better convergence. This could involve using $R^{[\beta]}$ with values other than $\beta = 1/2$, or perhaps using the fact that the derivatives of the $P_{\vec{w}}$ at the points of $\vec{\zeta}$ are also readily available computationally.

The conjugate operator, also known as the Hilbert transform (see [7], [9], [10]), which transforms a real-valued function on ∂D_0 into its conjugate function, involves integration of the kernel $\cot(s-t)$. This looks suspiciously like the multipliers in the matrix C. It would be interesting to know if our iterative process is the discretization of an integral operator or some similar concept as $N \to \infty$. Unfortunately, letting $N \to \infty$ in the four operators composing ϕ_γ in (3.2) causes each of them to tend to the identity operator, which thus gives no information.

We close with the following question. Consider in general N points w_j on ∂D equally spaced according to arc length. These generate N intermediate points u_j as we have seen, not generally lying on ∂D. If the w_j are shifted by a fixed amount of arc length in the same direction, we obtain from them a new set of u_j. Since the linear operators C and E commute, the $\vec{u}$ thus obtained evidently trace out a curve, in some sense dual to ∂D. What curve is this in general, how does it depend on N, and what information can it give us regarding conformal mapping?

Acknowledgment. I am grateful to the referees who pointed out some serious errors in the first version of this article and supplied several valuable suggestions.

REFERENCES

[1] L. V. Ahlfors, *Complex Analysis, An Introduction to the Theory of Analytic Functions of One Complex Variable*, 3rd ed., Internat. Ser. Pure Appl. Math., McGraw-Hill, New York, 1978.

[2] S. Barnett, *Matrices: Methods and Applications*, Oxford Appl. Math. Comput. Sci. Ser., Clarendon Press, Oxford, UK, 1990.

[3] C. de Boor, *A Practical Guide to Splines*, Springer-Verlag, New York, 1978.

[4] T. K. Delillo, *The accuracy of numerical conformal mapping methods: A survey of examples and results*, SIAM J. Numer. Anal., 31 (1994), pp. 788–812.

[5] T. A. Driscoll and S. A. Vavasis, *Numerical conformal mapping using cross-ratios and Delaunay triangulation*, SIAM J. Sci. Comput., 19 (1998), pp. 1783–1803.

[6] B. Fornberg, *A numerical method for conformal mappings*, SIAM J. Sci. Statist. Comput., 1 (1980), pp. 386–400.

[7] M. H. Gutknecht, *Solving Theodorsen's integral equation for conformal maps with the fast Fourier transform and various nonlinear iterative methods*, Numer. Math., 36 (1981), pp. 405–429.

[8] P. Henrici, *Essentials of Numerical Analysis with Pocket Calculator Demonstrations*, John Wiley and Sons, New York, 1982.

[9] P. Henrici, *Applied and Computational Complex Analysis*, Vol. 3, John Wiley and Sons, New York, 1986.

[10] P. K. Kythe, *Computational Conformal Mapping*, Birkhäuser, Boston, 1998.

[11] R. Wegmann, *Convergence proofs and error estimates for an iterative method for conformal mapping*, Numer. Math., 44 (1984), pp 435–461.

[12] R. Wegmann, *On Fornberg's numerical method for conformal mapping*, SIAM J. Numer. Anal., 23 (1986), pp 1199–1213.

[13] R. Wegmann, *Discrete Riemann-Hilbert problems, interpolation of simply closed curves, and numerical conformal mapping*, J. Comput. Appl. Math., 23 (1988), pp. 323–352.

SIAM J. Sci. Comput.
Vol. 23, No. 3, pp. 1042–1049

APPLICATION OF AN INVERSE PROBLEM FOR SYMMETRIC PERIODIC POTENTIALS*

G. B. XIAO†, K. YASHIRO‡, N. GUAN‡, AND S. OHKAWA‡

Abstract. This paper gives an algorithm for constructing symmetric periodic potential $q(x)$ of Hill's equation from given roots of $\Delta(\lambda) - 2 = 0$. The problem arises from synthesizing dual-mode ring electrical circuits. In the presented method, the discriminant $\Delta(\lambda)$ of Hill's equation is determined from the roots of $\Delta(\lambda) - 2 = 0$ at first, and then $q(x)$ is constructed from the roots of $\Delta(\lambda) \pm 2 = 0$ and $\Delta(\lambda) = 0$. Examples are provided to verify the algorithm.

Key words. inverse problem, Hill's equation, dual-mode ring circuit

AMS subject classifications. 94C99, 78A46

PII. S1064827500376211

1. Introduction.

In analyzing lossless dual-mode ring circuits with distributed parameters, it is often required to solve the following Hill's equation [1]:

$$-y''(x) + q(x)y(x) = \lambda y(x), \tag{1.1}$$

where the real periodic potential function $q(x)$ is related to the characteristic impedance of the ring circuit and satisfies

$$q(1+x) = q(x), \qquad q(1-x) = q(x), \qquad x \in (0,1). \tag{1.2}$$

Let $y_1(x,\lambda)$ and $y_2(x,\lambda)$ denote the two linearly independent solutions of (1.1), where $y_1(0,\lambda) = 1, y_1'(0,\lambda) = 0$ and $y_2(0,\lambda) = 0, y_2'(0,\lambda) = 1$. Their Wronskian is given by $W(y_1,y_2) = y_1y_2' - y_1'y_2 = 1$, and the Hill's discriminant is defined by $\Delta(\lambda) = y_1(1,\lambda) + y_2'(1,\lambda)$. The roots of $\Delta(\lambda) - 2 = 0$, $\Delta(\lambda) + 2 = 0$, and $\Delta(\lambda) = 0$ are denoted by $\lambda_i (i = 0,1,2,\ldots)$, $\mu_i (i = 1,2,\ldots)$, and $\gamma_i (i = 1,2,\ldots)$, respectively. In intervals of $(-\infty, \lambda_0)$, $(\lambda_{2i-1}, \lambda_{2i})$, and (μ_{2i-1}, μ_{2i}), there exists $\mid \Delta(\lambda) \mid> 2$. These intervals are instability intervals known as forbidden bands because the solutions of (1.1) are unbounded on the line [2].

Especially in ring electrical circuits, $\sqrt{\lambda_i}$ stand for resonating frequencies. The synthesis of such ring circuits is equivalent to solving the potential function $q(x)$ from given resonating frequencies $\sqrt{\lambda_i}$. Generally, periodic potentials can be uniquely recovered from norming constants and the zeros of $y_2(1,\lambda)$ [2], [3], but in this practical case, the norming constants are not known in advance. This paper will show that although $q(x)$ cannot be uniquely constructed in this case, but under the condition of (1.2), $q(x)$ can belong only to a function set $\{q_n(x)\}$, which can be wholly determined. A practical algorithm is provided and is verified by examples.

*Received by the editors August 4, 2000; accepted for publication (in revised form) April 18, 2001; published electronically September 26, 2001.
http://www.siam.org/journals/sisc/23-3/37621.html

†Graduate School of Natural Science, Chiba University 1-33 Yayoi-cho, Inage-Ku, Chiba 263-8522, Japan and Hunan University, Changsha, 410082, People's Republic of China (xiao@knight.te.chiba-u.ac.jp).

‡Department of Electronics and Mechanical Engineering, Chiba University, 1-33 Yayoi-cho, Inage-ku, Chiba 263-8522, Japan (yashiro@cute.te.chiba-u.ac.jp, guan@cute.te.chiba-u.ac.jp, ohkawa@cute.te.chiba-u.ac.jp).

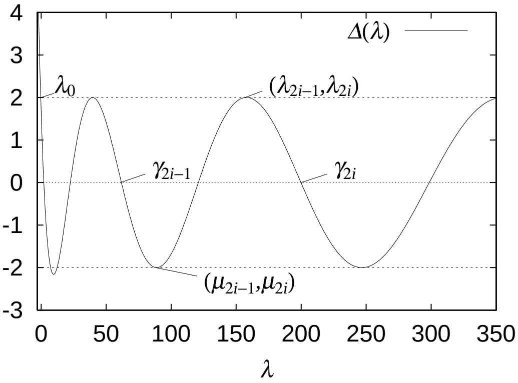


FIG. 1. *A general shape of* $\Delta(\lambda)$.

2. Recover $q(x)$ from λ_i's. Without loss of generality, we assume that $\int_0^1 q(s)ds = 0$ in the following discussions. A general shape of $\Delta(\lambda)$ is depicted in Figure 1. The roots are arranged as

$$\lambda_0 < \cdots < \gamma_{2i-1} < \mu_{2i-1} \leq \mu_{2i} < \gamma_{2i} < \lambda_{2i-1} \leq \lambda_{2i} < \cdots . \tag{2.1}$$

Let α_i, β_i denote the zeros of $y_1'(1,\lambda)$ and $y_2(1,\lambda)$, respectively. We recall some necessary results at first.

LEMMA 2.1. *When $q(x) \in L_2(0,1)$ and $\int_0^1 q(s)ds = 0$, we have the following asymptotic formulae for eigenvalues:*

$$\alpha_i = (i\pi)^2 + \nu_{1i}, \tag{2.2}$$

$$\beta_i = (i\pi)^2 + \nu_{2i}, \tag{2.3}$$

$$\lambda_{2i-1} = (2i\pi)^2 + \nu_{3i}, \tag{2.4}$$

$$\lambda_{2i} = (2i\pi)^2 + \nu_{4i}, \tag{2.5}$$

$$\mu_{2i-1} = [(2i-1)\pi]^2 + \nu_{5i}, \tag{2.6}$$

$$\mu_{2i} = [(2i-1)\pi]^2 + \nu_{6i}, \tag{2.7}$$

$$\gamma_i = [(i-0.5)\pi)]^2 + \nu_{7i}, \tag{2.8}$$

as $\lambda \uparrow \infty$, and $\sum_{i=1}^{\infty} \nu_{1i}^2, \ldots, \sum_{i=1}^{\infty} \nu_{7i}^2$ all converge. [4], [5].

LEMMA 2.2. *All entire functions of order $1/2$, type 1 can be expressed as a constant multiple of the canonical product formed from its roots* [6].

$\Delta(\lambda)$, $\Delta(\lambda) \pm 2$ are all entire functions of order 1/2 and type 1 [2], [6]. Thus, we can write

$$\Delta(\lambda) = C_1 \prod_{i=1}^{\infty} \left(1 - \frac{\lambda}{\gamma_i}\right), \tag{2.9}$$

$$\Delta(\lambda) - 2 = C_2 \prod_{i=0}^{\infty} \left(1 - \frac{\lambda}{\lambda_i}\right), \tag{2.10}$$

$$\Delta(\lambda) + 2 = C_3 \prod_{i=1}^{\infty} \left(1 - \frac{\lambda}{\mu_i}\right). \tag{2.11}$$

The estimates of $y_1(1,\lambda)$ and $y_2'(1,\lambda)$ are $y_1(\lambda) = \cos(\sqrt{\lambda})[1 + o(1)]$ and $y_2'(\lambda) = \cos(\sqrt{\lambda})[1 + o(1)]$ for $\lambda \downarrow -\infty$ [2]. It follows that $\Delta(\lambda) = 2\cos(\sqrt{\lambda})[1 + o(1)]$ for $\lambda \downarrow -\infty$. Therefore, we have

$$\lim_{\lambda \to -\infty} \frac{\Delta(\lambda)}{2\cos(\sqrt{\lambda})} = 1. \tag{2.12}$$

Using $\cos(\sqrt{\lambda}) = \prod_{i=1}^{\infty}(1 - \frac{\lambda}{\Gamma_i})$, $\Gamma_i = [(i - 0.5)\pi]^2$, the left-hand side of (2.12) can be spelled out as

$$\begin{aligned} \lim_{\lambda \to -\infty} \frac{\Delta(\lambda)}{2\cos(\sqrt{\lambda})} &= \lim_{\lambda \to -\infty} \frac{C_1 \prod_{i=1}^{\infty}(1 - \frac{\lambda}{\gamma_i})}{2\prod_{i=0}^{\infty}(1 - \frac{\lambda}{\Gamma_i})} = \lim_{\lambda \to -\infty} \frac{C_1}{2} \prod_{i=1}^{\infty} \frac{1 - \frac{\lambda}{\gamma_i}}{1 - \frac{\lambda}{\Gamma_i}} \\ &= \lim_{\lambda \to -\infty} \frac{C_1}{2} \prod_{i=1}^{\infty} \Big(1 + \frac{\lambda(\gamma_i - \Gamma_i)}{\gamma_i(\Gamma_i - \lambda)}\Big). \end{aligned} \tag{2.13}$$

From (2.8), $\gamma_i - \Gamma_i = o(1)$; therefore $\sum_{i=1}^{\infty} \frac{\lambda(\gamma_i - \Gamma_i)}{\gamma_i(\Gamma_i - \lambda)}$ is uniformly convergent at least in $(-\infty, 0)$, as is (2.13). This enables us to pass the limit $\lambda \to -\infty$ into the product in (2.13),

$$\lim_{\lambda \to -\infty} \frac{C_1}{2} \prod_{i=1}^{\infty} \Big(1 + \frac{\lambda(\gamma_i - (i - 0.5)^2)}{\gamma_i(\Gamma_i - \lambda)}\Big) = \frac{C_1}{2} \prod_{i=1}^{\infty} \frac{\Gamma_i}{\gamma_i} = 1; \tag{2.14}$$

hence,

$$C_1 = 2\prod_{i=1}^{\infty} \frac{\gamma_i}{\Gamma_i}. \tag{2.15}$$

In the same way,

$$C_2 = \lambda_0 \prod_{i=1}^{\infty} \frac{\lambda_{2i-1}\lambda_{2i}}{P_i^2}, \tag{2.16}$$

$$C_3 = 4\prod_{i=1}^{\infty} \frac{\mu_{2i-1}\mu_{2i}}{Q_i^2}, \tag{2.17}$$

where $P_i = (2\pi i)^2$, $Q_i = [(2i-1)\pi]^2$. As a result, (2.9), (2.10), (2.11) become

$$\Delta(\lambda) = 2\prod_{i=1}^{\infty} \frac{\gamma_i - \lambda}{\Gamma_i}, \tag{2.18}$$

$$\Delta(\lambda) - 2 = (\lambda_0 - \lambda)\prod_{i=1}^{\infty} \frac{(\lambda_{2i-1} - \lambda)(\lambda_{2i} - \lambda)}{P_i^2}, \tag{2.19}$$

$$\Delta(\lambda) + 2 = 4\prod_{i=1}^{\infty} \frac{(\gamma_{2i-1} - \lambda)(\gamma_{2i} - \lambda)}{Q_i^2}. \tag{2.20}$$

These equations show that λ_i, μ_i, γ_i, and $\Delta(\lambda)$ are actually equivalent pieces of information. If λ_i are provided, $\Delta(\lambda)$, μ_i, and γ_i are all obtainable.

We now prove the following theorem.

THEOREM 2.3. *$y_2'(1,\lambda) = y_1(1,\lambda) = \frac{1}{2}\Delta(\lambda)$ if $q(1-x) = q(x)$.*

Proof. When $q(1-x) = q(x)$, it is obvious that the eigenvalues corresponding to the boundary conditions of $y'(0,\lambda) = 0, y(1,\lambda) = 0$ and $y(0,\lambda) = 0, y'(1,\lambda) = 0$ are the same. This means that $y_1(1,\lambda)$ and $y_2'(1,\lambda)$ have the same zeros. Because $y_1(1,\lambda)$ and $y_2'(1,\lambda)$ are also entire functions of type 1, order 1/2, from Lemma 2.1, it is reasonable to write $y_1(1,\lambda) = Cy_2'(1,\lambda)$, where C is a constant, and from their estimates

$$C = \lim_{\lambda\to-\infty} \frac{y_1(1,\lambda)}{y_2'(1,\lambda)} = 1, \tag{2.21}$$

it leads to

$$y_2'(1,\lambda) = y_1(1,\lambda) = \frac{1}{2}\Delta(\lambda). \tag{2.22}$$

This proves the theorem. □

With it the following relation can be drawn from the Wronskian of y_1 and y_2:

$$y_1'(1,\lambda)y_2(1,\lambda) = y_2'(1,\lambda)y_1(1,\lambda) - 1 = \frac{1}{4}[\Delta(\lambda)+2][\Delta(\lambda)-2]. \tag{2.23}$$

Equation (2.23) illustrates that the zeros of $y_1'(1,\lambda)$ and $y_2(1,\lambda)$, i.e., α_i, β_i, can be only those values of the zeros of $\Delta(\lambda)-2$ or $\Delta(\lambda)+2$, i.e., λ_i or μ_i. In order to divide these zeros, more preparatory results are needed as follows.

LEMMA 2.4. *If $\int_0^1 q(s)ds = 0$, then $\lambda_0 < 0$ and $\alpha_0 < 0$, unless $q(x) = 0$ for all x[7].*

Proof. Let $y_0(x)$ denote the eigenfunction corresponding to λ_0 or α_0; then $y_0(x) \neq 0$ for all x because $y_0(x)$ has the least oscillation in all the eigenfunctions in both cases. Write (1.1) as

$$\frac{y_0''(x)}{y_0(x)} + \lambda_c - q(x) = 0, \tag{2.24}$$

where $\lambda_c = \lambda_0$, or α_0. Integrating it over (0,1), we get

$$\int_0^1 \frac{y_0''(x)}{y_0(x)}dx + \lambda_c = 0. \tag{2.25}$$

Forwarding the integration by part leads to

$$\frac{y_0'(x)}{y_0(x)}\bigg|_0^1 + \int_0^1 \left(\frac{y_0'(x)}{y_0(x)}\right)^2 dx + \lambda_c = 0. \tag{2.26}$$

Recall the boundary conditions of $y_0(1) = y_0(0), y_0'(1) = y_0'(0)$ for $\lambda_c = \lambda_0$, and $y_0'(1) = y_0'(0) = 0$ for $\lambda_c = \alpha_0$. In both cases, $\frac{y_0'(x)}{y_0(x)}\Big|_0^1 = 0$. Therefore $\lambda_c < 0$ unless $y_0'(x) = 0$ almost everywhere. In the latter instance, $y_0(x)$ must be a constant. From (1.1) and $\int_0^1 q(x)dx = 0$, we can see that $q(x) = 0$ must hold for all x.

This lemma shows that in (2.23) λ_0 ought to be assigned to α_0.

LEMMA 2.5. *The zeros of $y_1(1,\lambda)$ and those of $y_1'(1,\lambda)$ interlace; the zeros of $y_2(1,\lambda)$ and those of $y_2'(1,\lambda)$ interlace* [4].

It follows from Lemma 2.5 that μ_{2i-1} and μ_{2i} can be assigned only to one zero of $y_2(1,\lambda)$ and one zero of $y_1'(1,\lambda)$, as well as for λ_{2i-1} and λ_{2i}.

We conclude these points in Theorem 2.6.

THEOREM 2.6. *When $q(1-x) = q(x)$, the zeros of $y_1(1,\lambda)$ and $y_2'(1,\lambda)$ coincide with those of $\Delta(\lambda)$, while α_i and β_i satisfy*

$$\begin{aligned} &\alpha_0 = \lambda_0, \\ &(\alpha_{2i-1}, \beta_{2i-1}) \leftrightarrow (\mu_{2i-1}, \mu_{2i}), \\ &(\alpha_{2i}, \beta_{2i}) \leftrightarrow (\lambda_{2i-1}, \lambda_{2i}). \end{aligned} \tag{2.27}$$

Here we have used a symbolical expression $(\alpha_{2i-1}, \beta_{2i-1}) \leftrightarrow (\mu_{2i-1}, \mu_{2i})$, which means that we can choose either α_{2i-1} to be μ_{2i-1} and β_{2i-1} to be μ_{2i} or choose α_{2i-1} to be μ_{2i} and β_{2i-1} to be μ_{2i-1}.

By now, we have shown a method to generate the zeros of $y_1(1,\lambda)$, $y_1'(1,\lambda)$ and $y_2(1,\lambda)$, $y_2'(1,\lambda)$ from λ_i provided. It is known that to construct $q(x)$ from the zeros of $y_1(1,\lambda)$ and $y_1'(1,\lambda)$ or $y_2(1,\lambda)$ and $y_2'(1,\lambda)$ are both overdetermined inverse classical Sturm–Liouville problems that are uniquely solvable [6], [8]. Theoretically, the division according to (2.27) is infinite, so there exists an infinite function sequence $q_n(x)$ comprising a same $\Delta(\lambda)$. However, as the width of forbidden band decreases exponentially fast [3], only a limited number of forbidden bands (say, M) will cause significant influence on $q_n(x)$, or we need to consider only approximately 2^M $q_n(x)$ for given λ_i. Practically, if we just want to obtain a $q(x)$ which comprises the provided λ_i , not minding whether $q(x)$ is unique or not, we may simply choose a kind of combination from (2.27). Also, we can construct several $q_n(x)$ and from them select the most proper one for the practical problem at hand.

3. Algorithm and examples.

We detail our algorithm as follows.

(1) Generate $\Delta(\lambda)$ from given λ_i by (2.19).

It is inconvenient to use (2.19) directly in calculation because the canonical product contains infinite terms. From (2.4) we assume that $\lambda_{2i-1} \approx \lambda_{2i} \approx P_i$ when $i > N$. Then (2.19) becomes

$$\begin{aligned} \tilde{\Delta}(\lambda) - 2 &= (\lambda_0 - \lambda) \prod_{i=1}^{N} \frac{(\lambda_{2i-1} - \lambda)(\lambda_{2i} - \lambda)}{P_i^2} \prod_{i=N+1}^{\infty} \frac{(P_i - \lambda)(P_i - \lambda)}{P_i^2} \\ &= 4 \sin^2 \frac{\sqrt{\lambda}}{2} \frac{\lambda_0 - \lambda}{\lambda} \prod_{i=1}^{N} \frac{(\lambda_{2i-1} - \lambda)(\lambda_{2i} - \lambda)}{(P_i - \lambda)^2}, \end{aligned} \tag{3.1}$$

where $\sin\frac{\sqrt{\lambda}}{2} = \frac{\sqrt{\lambda}}{2}\prod_1^{\infty}(\frac{P_i-\lambda}{P_i})$ is used. Though (3.1) is an approximate formula for $\Delta(\lambda)-2$, it guaranteed that the first $2N$ zeros are actually those provided. This is enough in practical applications, where we often care only about a limited frequency range.

(2) Compute μ_i, γ_i ($i \le 2N$) from $\tilde{\Delta}(\lambda)$.

(3) Obtain α_i and β_i from (2.27), while the zeros of $y_1(1,\lambda)$ and $y_2'(1,\lambda)$ are the same as γ_i.

(4) Construct $q(x)$ from either the zeros of $y_1(1,\lambda)$ and $y_1'(1,\lambda)$ or from the zeros of $y_2(1,\lambda)$ and $y_2'(1,\lambda)$. Readers are suggested to [8] for the detailed procedure of solving these kinds of inverse problems.

We will verify this algorithm by two examples.

Example 3.1. The provided λ_i are those of $q^0(x) = cos(2\pi x)$. In this case, only the first forbidden band has significant width, so $q_n(x)$ contains two typical potential functions. The second forbidden band may slightly affect the reconstructed potentials. The four reconstructed potentials from the combination of the first two pairs of zeros in (2.27) are shown in Figure 2, while the effect of other forbidden bands are too small to yield a potential function that can be distinguished from these four potentials.

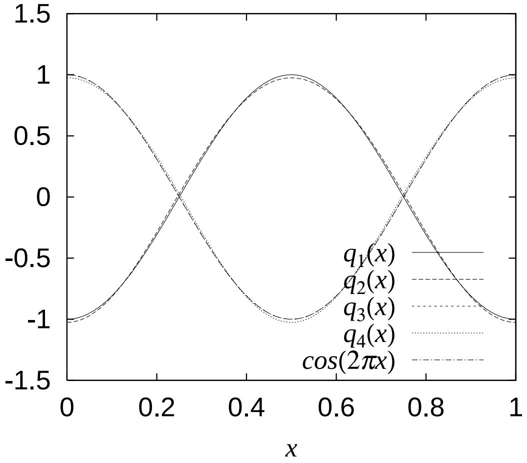


FIG. 2. *Example of continuous potential $q^0(x) = cos(2\pi x)$. $q_1(x) \sim q_4(x)$ are corresponding to different choices of the first two pairs of zeros in* (2.27).

Example 3.2. The provided λ_i are those of a rectangular pulse $q^0(x)$. In this case, about 15 forbidden bands have significant width, so we may have about 2^{15} different potentials that comprise the same λ_i's as $q^0(x)$ and can also be recovered. In Figure 3, we present only eight of them, which are corresponding to different choices for $i \le 3$ in (2.27), and probably the typical shapes in the set of $\{q_n(x)\}$. $q^a(x)$ in the figures are constructed from the zeros of $y_1(1,\lambda)$ and $y_1'(1,\lambda)$, while $q^b(x)$ are from the zeros of $y_2(1,\lambda)$ and $y_2'(1,\lambda)$. $q^a(x)$ and $q^b(x)$ agree very well in all these cases.

We have adopted a parameter $E_r(\lambda_i)$ to examine the errors of the presented algorithm. $E_r(\lambda_i)$ is defined as

$$E_r(\lambda_i) = \Big|\frac{\lambda_i - \tilde{\lambda_i}}{\lambda_i}\Big|, \tag{3.2}$$

where λ_i are those provided values, and $\tilde{\lambda_i}$ are calculated from (1.1) by applying

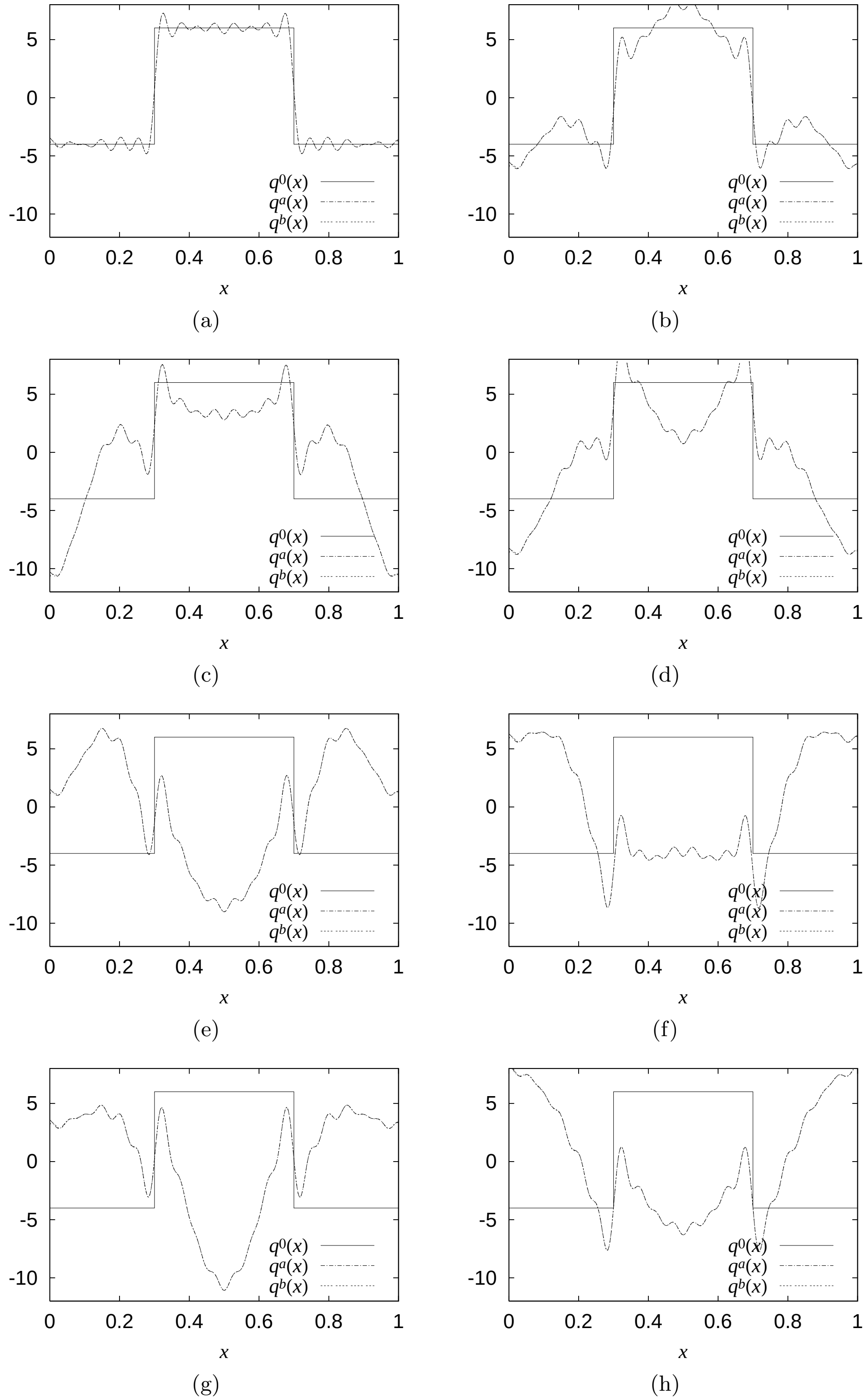


FIG. 3. *$q^0(x)$ is a rectangular pulse. Only 2^3 $q_n(x)$ are constructed here.*

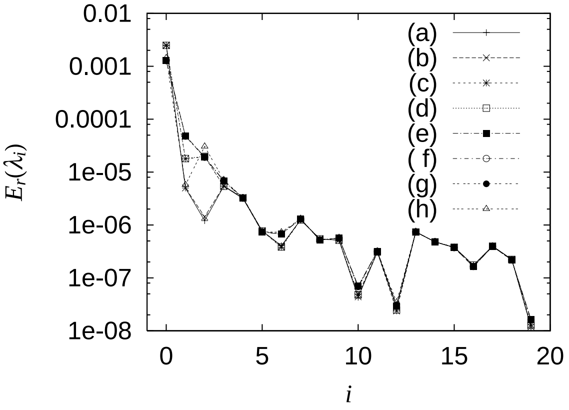


FIG. 4. $E_r(\lambda_i)$ *for* $0 \le i \le 20$.

constructed $q_n(x)$. Figure 4 shows the $E_r(\lambda_i)$ of all the $(a) \sim (h)$ cases in Example 3.2.

4. Conclusion. The inverse problem of constructing $q(x)$ of Hill's equation from given roots of $\Delta(\lambda) - 2 = 0$ has been changed to an inverse classical Sturm–Liouville problem of constructing $q(x)$ from two eigenvalue sequences, which can be solved by several established procedures. With the presented algorithm, it is possible to synthesize a ring electrical circuit that resonates at given frequencies.

REFERENCES

[1] D. C. YOULA, *Analysis and synthesis of arbitrarily terminated lossless nonuniform lines*, IEEE Trans. Circuit Theory, 11 (1964), pp. 363–372.

[2] H. P. MCKEAN AND E. TRUBOWITZ, *Hill's operator and hyperelliptic function theory in the presence of infinitely many branch points*, Comm. Pure Appl. Math., 29 (1976), pp. 143–226.

[3] E. TRUBOWITZ, *The inverse problem for periodic potentials*, Comm. Pure Appl. Math., 30 (1977), pp. 321–337.

[4] V. A. MARCHENKO, *Sturm–Liouville Operators and Applications*, Oper. Theory Adv. Appl. 22, Birkhäuser Verlag, Basel, 1986.

[5] J. PÖSCHEL AND E. TRUBOWITZ, *Inverse Spectral Theory*, Academic Press, Boston, 1987.

[6] B. M. LEVITAN AND M. G. GASYMOV, *Determination of a differential equation by two of its spectra*, Usephi Mat. Nauk, 19 (1964), pp. 3–63.

[7] H. HOCHSTADT, *The Functions of Mathematical Physics*, Dover, New York, 1986.

[8] W. RUNDELL AND P. E. SACKS, *Reconstruction techniques for classical inverse Sturm–Liouville problems*, Math. Comp., 58 (1992), pp. 161–183.

SIAM J. SCI. COMPUT.
Vol. 23, No. 3, pp. 1050–1051

A NOTE ON PRECONDITIONING NONSYMMETRIC MATRICES[*]

ILSE C. F. IPSEN[†]

Abstract. The preconditioners for indefinite matrices of KKT form in [M. F. Murphy, G. H. Golub, and A. J. Wathen, *SIAM J. Sci. Comput.*, 21 (2000), pp. 1969–1972] are extended to general nonsymmetric matrices.

Key words. preconditioner, minimal polynomial

AMS subject classifications. 65F10, 15A23

PII. S1064827500377435

In [2] preconditioners for real indefinite matrices of KKT form

$$\mathcal{A} \equiv \begin{pmatrix} A & B^* \\ C & 0 \end{pmatrix}$$

are presented.[1] The preconditioners $\mathcal{P}$ are of the following form:

$$\begin{pmatrix} A & B^* \\ 0 & \pm CA^{-1}B^* \end{pmatrix}, \quad \begin{pmatrix} A & B^* \\ C & 2CA^{-1}B^* \end{pmatrix}, \quad \begin{pmatrix} A & 0 \\ 0 & CA^{-1}B^* \end{pmatrix}.$$

The preconditioned matrices $\mathcal{P}^{-1}\mathcal{A}$ have minimal polynomials of degree at most 4. Hence a Krylov subspace method like GMRES applied to a preconditioned linear system with coefficient matrix $\mathcal{P}^{-1}\mathcal{A}$ converges in 4 iterations or less, in exact arithmetic.

We extend the preconditioners $\mathcal{P}$ in [2] to general matrices $\mathcal{A}$ by deriving them from LU decompositions of $\mathcal{A}$. As before, the preconditioned matrices $\mathcal{P}^{-1}\mathcal{A}$ and $\mathcal{A}\mathcal{P}^{-1}$ have minimal polynomials of degree at most 4.

Let

$$\mathcal{A} \equiv \begin{pmatrix} A & B^* \\ C & D \end{pmatrix}$$

be a complex, nonsingular matrix where the leading principal submatrix A is nonsingular. Let $S \equiv D - CA^{-1}B^*$ be the Schur complement with respect to A. Since $\mathcal{A}$ is nonsingular, so is S. The idea is to factor $\mathcal{A} = \mathcal{L}\mathcal{D}\mathcal{U}$ such that the preconditioned matrix $\mathcal{L}^{-1}\mathcal{A}\mathcal{U}^{-1} = \mathcal{D}$ has a minimal polynomial of small degree.

PROPOSITION 1 (extension of Remark 2 in [2]). *If*

$$\mathcal{P} \equiv \begin{pmatrix} A & B^* \\ 0 & S \end{pmatrix},$$

then

$$\mathcal{A}\mathcal{P}^{-1} = \begin{pmatrix} I & 0 \\ CA^{-1} & I \end{pmatrix},$$

and $\mathcal{P}^{-1}\mathcal{A}$ and $\mathcal{A}\mathcal{P}^{-1}$ have the minimal polynomial $(\lambda - 1)^2$.

[*] Received by the editors September 1, 2000; accepted for publication (in revised form) October 18, 2000; published electronically September 26, 2001.
http://www.siam.org/journals/sisc/23-3/37743.html

[†] Center for Research in Scientific Computation, Department of Mathematics, North Carolina State University, P.O. Box 8205, Raleigh, NC 27695-8205 (ipsen@math.ncsu.edu, http://www4.ncsu.edu/~ipsen/). This research was supported in part by NSF grant DMS-9714811.

[1] The superscript * denotes the conjugate transpose.

PROPOSITION 2 (extension of (5) in [2]). *If*

$$\mathcal{P} \equiv \begin{pmatrix} A & B^* \\ 0 & -S \end{pmatrix},$$

then

$$\mathcal{A}\mathcal{P}^{-1} = \begin{pmatrix} I & 0 \\ CA^{-1} & -I \end{pmatrix},$$

and $\mathcal{P}^{-1}\mathcal{A}$ *and* $\mathcal{A}\mathcal{P}^{-1}$ *have the minimal polynomial* $(\lambda - 1)(\lambda + 1)$.

The preconditioned matrix below is the same, up to permutations, as the one in [1, section 2.1].

PROPOSITION 3. *If*

$$\mathcal{P}_1 \equiv \begin{pmatrix} I & 0 \\ CA^{-1} & -I \end{pmatrix}, \qquad \mathcal{P}_2 \equiv \begin{pmatrix} A & B^* \\ 0 & S \end{pmatrix},$$

then

$$\mathcal{P}_1^{-1}\mathcal{A}\mathcal{P}_2^{-1} = \begin{pmatrix} I & 0 \\ 0 & -I \end{pmatrix}.$$

The preconditioned matrix is also similar to $\mathcal{P}^{-1}\mathcal{A}$ and $\mathcal{A}\mathcal{P}^{-1}$, where

$$\mathcal{P} \equiv \begin{pmatrix} A & B^* \\ C & D - 2S \end{pmatrix},$$

which is an extension of the preconditioner in [2, p. 7].

REMARK 1. *Extending the preconditioner in* [2, Proposition 1] *to general matrices gives*

$$\mathcal{P} \equiv \begin{pmatrix} A & \\ & -S \end{pmatrix}.$$

It can be derived from the scaled LU decomposition $\mathcal{A} = \mathcal{L}\mathcal{U}\mathcal{D}$, *where*

$$\mathcal{L} \equiv \begin{pmatrix} I & \\ CA^{-1} & I \end{pmatrix}, \qquad \mathcal{U} \equiv \begin{pmatrix} I & -B^*S^{-1} \\ & -I \end{pmatrix}, \qquad \mathcal{D} \equiv \begin{pmatrix} A & \\ & -S \end{pmatrix}.$$

The preconditioned matrix is

$$\mathcal{T} \equiv \mathcal{A}\mathcal{P}^{-1} = \mathcal{L}\mathcal{U} = \begin{pmatrix} I & -B^*S^{-1} \\ CA^{-1} & -DS^{-1} \end{pmatrix}.$$

If $\mathcal{A}$ *is of KKT form with* $D = 0$, *then*

$$\mathcal{T}^2 - \mathcal{T} = \begin{pmatrix} -B^*S^{-1}CA^{-1} & 0 \\ 0 & I \end{pmatrix}.$$

Since $(\mathcal{T}^2 - \mathcal{T})^2 = \mathcal{T}^2 - \mathcal{T}$, *the preconditioned matrix* $\mathcal{T}$ *has a minimal polynomial of degree* 4.

REFERENCES

[1] P. E. GILL, W. MURRAY, D. B. PONCELEÓN, AND M. A. SAUNDERS, *Preconditioners for indefinite systems arising in optimization*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 292–311.

[2] M. F. MURPHY, G. H. GOLUB, AND A. J. WATHEN, *A note on preconditioning for indefinite linear systems*, SIAM J. Sci. Comput., 21 (2000), pp. 1969–1972.

SIAM J. SCI. COMPUT.
Vol. 23, No. 4, pp. 1053–1076

CONSTRUCTION OF NEARLY ORTHOGONAL NEDELEC BASES FOR RAPID CONVERGENCE WITH MULTILEVEL PRECONDITIONED SOLVERS[*]

DIN-KOW SUN[†], JIN-FA LEE[‡], AND ZOLTAN CENDES[†]

Abstract. This paper presents a systematic approach to constructing high-order tangential vector basis functions for the multilevel finite element solution of electromagnetic wave problems. The new bases allow easy computation of a preconditioner to eliminate or at least weaken the indefiniteness of the system matrix and thus reduce the condition number of the system matrix. When these bases are used in multilevel solutions, where the multilevels correspond to the order of the basis functions, the resulting p-multilevel-ILU preconditioned conjugate gradient method (MPCG) provides an optimal rate of convergence. We first derive an admissible set of vectors of order p, and decompose this set into two subspaces—rotational and irrotational (gradient). We then reduce the number of vectors by making them orthogonal to all previously constructed lower-order bases. The remaining vectors are made mutually orthogonal in both the vector space and in the range space of the curl operator. The resulting vector basis functions provide maximum orthogonality while maintaining tangential continuity of the field. The zeroth-order space is further decomposed using a scalar-vector formulation to eliminate convergence problems at extremely low frequencies. Numerical experiments show that number of iterations needed for the solution by MPCG is basically constant, regardless of the order of the basis or of the matrix size. Computational speed is improved by several orders of magnitude due to the fast matrix solution of MPCG and to the high accuracy of the higher-order bases.

Key words. *P*-multilevel finite element methods, Maxwell's equations, multilevel preconditioned conjugate gradient method, tangential vector basis functions

AMS subject classifications. 65N22, 65N55, 65F10, 83C50

PII. S1064827500367531

1. Introduction.

In 1980, Nedelec defined a set of vector finite element basis functions having the properties that they are complete to order p in the range space of the curl operator, that they impose tangential but not normal continuity of the vector, and that they are unisolvent [1].

In this paper, we will call finite elements based on these functions Nedelec elements and denote the function space spanned by these elements as $E^p(curl)$. Nedelec elements have been shown to be important in the solution of electromagnetic field problems [2], [3]. In particular, [4], [5], [6], [7] showed that these elements eliminate the problem of spurious modes that plague conventional node based approximations of the vector wave equation derived from Maxwell's equations. Numerous authors have derived alternative forms of the Nedelec bases, both for low-order "edge elements" [3], [7], [8], [9], [10], [11], [12] and for high-order tangential vector elements [5], [6], [13], [14], [15], [16], [17], [18], [19], [20], [21]. The goal of these constructions has often been to make the Nedelec bases interpolatory or hierarchical. Hiptmair [22] recently provided a general abstract framework for the systematic construction of these vector bases. With such an abundance of methods, the best choice of bases is often debated. All of the high-order bases are complete to order p-1 in the range space of the curl

[*] Received by the editors February 9, 2000; accepted for publication (in revised form) December 1, 2000; published electronically October 11, 2001.
http://www.siam.org/journals/sisc/23-4/36753.html

[†] Ansoft Corporation, 4 Station Square, Pittsburgh, PA 15219 (dinsun@earthlink.net, Zol@ansoft.com).

[‡] ECE Department, Ohio State University, Columbus, OH (inlee@ee.eng.ohio-state.edu).

operator, so their numerical solutions have the same order of accuracy. This means that they are all optimal in the sense of requiring the minimum number of unknowns to obtain a pth-order convergence rate. Nevertheless, the question arises: Does any particular basis set have advantages over the others?

To answer this question, we need to examine more closely the meaning of the word "optimal" in the tangential vector finite element framework. In terms of accuracy, an infinite number of optimal, nonunique bases may be constructed in this space. However, the overall performance of these elements varies greatly when one considers matrix solution speed. Researchers have found that the convergence of iterative matrix solution algorithms such as the conjugate gradient algorithm varies considerably when the system of equations is created using different tangential vector finite elements [23], [24]. These papers study the condition number of various basis sets and compare the convergence behavior of the conjugate gradient algorithm with preconditioning and without preconditioning. Contrary to common belief, some badly conditioned matrices converge more rapidly than other better conditioned matrices. Some preconditioned matrices converge slower than others without preconditioning, while others converge faster with preconditioning; still others take too many iterations to converge at all. What is going on?

Let us first note that many positive definite systems of equations do not converge rapidly without preconditioning. Also, note that although it may take considerable work, even a matrix with a large condition number can be preconditioned to form a nearly identity matrix. The convergence rate of the conjugate gradient algorithm is determined by the matrix after preconditioning and is inversely proportional to the condition number of the preconditioned matrix, not that of the original matrix. This explains why the performance of some vector sets is dramatically improved after preconditioning, while the others are not. Further, in the case of the vector wave equation, the matrix equation is not positive definite and the conjugate gradient method is not applicable. However, as shown in [25], [26], [27], if the indefinite nature of the matrix can be eliminated or at least greatly reduced, then the conjugate gradient method can be applied. The varying convergence results indicate the sensitivity of the condition number of the preconditioned matrix to both the preconditioner and the choice of basis functions. Therefore, the question of which basis to use is intimately tied to the need to find, with reasonable computational cost, a good preconditioner for the matrix generated by that basis.

This leads us to answer the "optimal by what criteria" question. We propose that some high-order tangential vector basis sets are better than others because they are more amenable to preconditioning. Hence we adopt the following criteria for choosing the optimum tangential vector basis. *The optimum tangential vector finite element basis set is the one that results in the fastest convergence when the resulting system of equations is solved by the conjugate gradient algorithm.*

A matrix perfectly preconditioned into the identity matrix allows the conjugate gradient algorithm to converge in one iteration. A mutually orthogonal set of vectors, with the additional property of mutually orthogonal curls of the vectors, generates a diagonal system matrix for the vector wave equation. Such vectors are, of course, the eigenvectors of the associated generalized matrix eigenvalue problem and are too expensive to compute in general. Even if expense were not a factor, the resulting system eigenvectors span the entire problem domain, unlike finite element basis functions that are local in nature. However, we may readily compute the eigenvectors of the element matrices. This leads to the question: Is it possible to have high degrees of local orthogonality without violating the property of tangential continuity?

The current work extends Babuska, Griebel, and Pitkaranta [28], where orthogonal scalar bases are constructed with nodal elements. Further, the construction of preconditioners for the p-version of scalar finite elements is given in [29], [30], [31]. Here we construct orthogonal vector bases based on the reduction of the number of negative eigenvalues and the condition number of the system matrix. Unlike previous work, we optimize the performance of the iterative solver by explicitly exploiting the connection between the basis vectors, the system matrix, and the preconditioner.

In the following analysis, we focus on the vector wave equation defined in terms of the curl-curl operator and requiring tangential continuity. Different differential equations have different requirements of orthogonality and continuity; an optimal basis set for one differential equation may not be optimal for another. We begin in section 2 by deriving the system equation from Maxwell equations. In section 3, we enumerate our strategy for achieving maximum orthogonality, and in section 4 we describe how to accomplish each strategy. In section 5, we count the numbers of bases in various spaces. Section 6 provides details of the multilevel-ILU preconditioned conjugate gradient method (MPCG) algorithm used to solve the system equation. Numerical results that demonstrate the effectiveness of the newly constructed set are presented in section 7, and conclusions are drawn in section 8.

2. System equation. From Maxwell's equations, the field in a three-dimensional discontinuity region Ω satisfies the vector wave equation

$$\nabla \times \frac{1}{\mu_r} \nabla \times \vec{E} - k^2 \varepsilon_r \vec{E} = 0 \quad \text{in } \Omega \tag{1}$$

with the boundary conditions

$$\begin{aligned} \vec{H} \times \vec{n} &= \vec{H}_t \quad \text{on } \Gamma_H, \\ \vec{E} \times \vec{n} &= 0 \qquad \text{on } \Gamma_E. \end{aligned}$$

Here $\vec{E}$ and $\vec{H}$ are the electric and magnetic fields, $\vec{H}_t$ is the excitation magnetic field on the boundary, $\vec{n}$ is the outward unit normal to the boundary, ε_r and μ_r are the relative permittivity and permeability of the material, respectively, and k is the free space wave-number.

Applying Galerkin's method to (1) gives the system equation

$$Ae = b, \tag{2}$$

where

$$\begin{aligned} A &= S - k^2 T, \\ S_{ij} &= \int \frac{1}{\mu_r} \nabla \times \vec{\alpha}_i \cdot \nabla \times \vec{\alpha}_j \, d\Omega, \\ T_{ij} &= \int \varepsilon_r \vec{\alpha}_i \cdot \vec{\alpha}_j \, d\Omega, \end{aligned} \tag{3}$$

and $\{\vec{\alpha}_i\}$ is a set of vector basis functions.

3. Strategy. Two vectors $\vec{\alpha}_i$ and $\vec{\alpha}_j$ are orthogonal in a tetrahedra if they satisfy

$$\vec{\alpha}_i \perp \vec{\alpha}_j \;\; \equiv \;\; \int_{tet} \vec{\alpha}_i \cdot \vec{\alpha}_j \, d\Omega = 0 \quad \text{when } i \neq j.$$

To derive orthogonal bases, we employ the following strategy.

3.1. Admissible space. We first define the maximum space of admissible functions that satisfies tangential continuity across the faces of the tetrahedron and then search the functions in this space to obtain maximum orthogonality.

3.2. Subspace decomposition. The primary reason for nonconvergence of the conjugate gradient algorithm is that the coefficient matrix A is either nearly singular or has many negative eigenvalues [25]. As defined in (3), a number of the eigenvalues of S are zero, generating many negative eigenvalues in A. The combination of zero and nonzero eigenmodes from S makes it hard to compute a good approximate inverse matrix. However, if we decompose the basis vectors into two orthogonal subspaces, rotational and irrotational (gradient), the contribution of S from the gradient space vanishes [26], [27]. Thus that part of A becomes negative definite and can be roughly inverted by an incomplete Cholesky factorization.

To avoid confusion, we note that the rotational subspace we construct is only approximately rotational because one cannot construct a normally continuous basis in the tangential vector space. However, for the sake of convenience, we use the term “rotational” throughout this paper to describe the component remaining after removing the gradient space.

3.3. Model tetrahedron. An orthogonal vector set over one tetrahedron shape is not necessarily orthogonal in other shapes. However, we cannot have different orthogonal basis functions over different elements because the continuity requirement across the common face of two adjacent tetrahedra would be violated. Therefore, we use the regular tetrahedron as the canonical shape by which to construct the orthogonal bases, and we use the same bases in all elements. Most tetrahedra in a well-made mesh are close to being regular so that we achieve near orthogonality in these elements. Numerical experiments indicate that using the regular tetrahedron as the canonical tetrahedron shape works well in practice.

3.4. Symmetrization. Symmetrized basis vectors are independent of tetrahedron orientation and avoid mixing degenerate modes. Symmetrized or antisymmetrized basis vectors provide unique sets of basis functions.

3.5. ST-orthogonal. The construction we present is hierarchical in the geometry of the element: edge-associated vectors are made orthogonal to edge-associated vectors first, and then to face-associated vectors if there are extra degrees of freedom. A face-associated vector is made orthogonal to face-associated vectors first, and then to volume-associated vectors if there are extra degrees of freedom. However, volume-associated vectors are made orthogonal only to themselves. Taking vector products between admissible vectors and the orthogonalized vectors generates an underdetermined matrix equation. This equation is solved using singular value decomposition to find the basis vectors.

To avoid mixing gradient vectors with rotational vectors, it is possible only to make the gradient vectors orthogonal to the other gradient vectors. On the other hand, rotational vectors can be made orthogonal to both rotational and gradient vectors. It is crucial to preserve the complete separation of the gradient space as discussed in section 3.2. The remaining edge-associated, face-associated, and volume-associated vectors are made mutually orthogonal not only in the vector space itself, but also in the range space of the curl operator. This is done by forming S and T matrices for the vector set and then computing the eigenvectors of the (S, T) pair. We call the newly formed eigenvector set ST-orthogonal, because the local S and T matrices computed from this basis set are diagonal. This leads to diagonal matrices for the local individual blocks within the global matrix.

3.6. $\boldsymbol{E^0(curl)}$ decomposition. We further decompose $E^0(curl)$, the lowest-order tangential vector space, into rotational and gradient spaces by the scalar-vector formulation of [32]. Without this decomposition, A becomes singular in the low frequency limit [32], [33], [34]. With this decomposition, A can be scaled by frequency into a well-conditioned matrix even at low frequencies. This completes the separation of rotational and gradient spaces.

4. Orthogonal tangential vectors.

Let us define the following terms to be used in the remainder of the paper.

- G_e^i, G_f^i, G_v^i are additional edge-associated, face-associated, and volume-associated gradient spaces when the order increases from $i-1$ to i, respectively.
- R_e^i, R_f^i, R_v^i are similar to G_e^i, G_f^i, G_v^i for rotational spaces.
- E_e^i, E_f^i, E_v^i : $E_e^i = G_e^i \cup R_e^i$. E_f^i, E_v^i are similar to E_e^i.
- G_e^I, G_f^I, G_v^I are gradient spaces of order i, i.e., $G_e^I = G_e^i \cup G_e^{i-1} \ldots \cup G_e^1$. G_f^I, G_v^I are similar to G_e^I.
- R_e^I, R_f^I, R_v^I are similar to G_e^I, G_f^I, G_v^I.
- E_e^I, E_f^I, E_v^I are similar to G_e^I, G_f^I, G_v^I.
- E^i is the tangential vector space for a single tetrahedron with function space complete to order i.
- $E^i(curl)$ is the tangential vector space of a single tetrahedron with range space complete to order i.
- F^i is the tangential finite element space of a contiguous set of tetrahedra with function space complete to order i.
- $F^i(curl)$ is the tangential finite element space of a contiguous set of tetrahedra with range space complete to order i.
- P^i is a polynomial of degree i.
- $\overline{P}^i$ is a homogeneous polynomial of degree i.

$$P_e^i, P_f^i, P_v^i : P_e^i = P^i(\lambda_0, \lambda_1), P_f^i = P^i(\lambda_0, \lambda_1, \lambda_2), P_v^i = P^i(\lambda_0, \lambda_1, \lambda_2, \lambda_3).$$

- $\#(A)$ is the number of vectors in A.
- $\#(e)$, $\#(f)$, $\#(v)$ are the numbers of edges, faces, and tetrahedra for a mesh, respectively.
- $A \perp B$ basis vectors in A are orthogonal to those in B.

4.1. Admissible tangential vectors of order p.

4.1.1. Edge-associated vectors. A vector along edge $\vec{l}_{01}$ of a triangle can be written as

$$\vec{v} = f(\lambda_0, \lambda_1)\nabla\lambda_0 + g(\lambda_0, \lambda_1)\nabla\lambda_1,$$

where the λ_i's are simplex coordinates in the triangle, and f and g are polynomials of degree p. To preserve tangential continuity, $\vec{v}$ must vanish on the other two edges of the triangle, i.e., on $\vec{l}_{02}$ and $\vec{l}_{12}$. On edge $\vec{l}_{02}$

$$\vec{l}_{02} \cdot \vec{v}\Big|_{\lambda_1=0} = 0.$$

Since

$$\vec{l}_{ij} \cdot \nabla\lambda_k = \begin{cases} 0, & i \neq j \neq k, \\ 1, & j = k, \\ -1, & i = k, \end{cases}$$

f must include a factor of λ_1. By the same reasoning, along $\vec{l}_{12}$, g must include a factor of λ_0. Consequently, the admissible edge-associated vector has the form

$$\vec{v} = s(\lambda_0, \lambda_1)\lambda_1 \nabla\lambda_0 + t(\lambda_0, \lambda_1)\lambda_0 \nabla\lambda_1, \tag{4}$$

where s and t are polynomials of degree $p-1$. The tangential components of (4) are seen to vanish on the faces of the tetrahedron except for the faces sharing edge $\vec{l}_{01}$. As long as the coefficients of (4) are common on the edge shared by adjacent tetrahedra, (4) can be added to a tangential vector without altering its tangential continuity. The number of independent variables in (4) is $\#(P_e^{p-1}) \times 2 = 2\binom{p+1}{2}$.

4.1.2. Face-associated vectors. A vector on face Δ_{012} of a tetrahedron can be written as

$$\vec{v} = f(\lambda_0, \lambda_1, \lambda_2)\nabla\lambda_0 + g(\lambda_0, \lambda_1, \lambda_2)\nabla\lambda_1 + h(\lambda_0, \lambda_1, \lambda_2)\nabla\lambda_2.$$

To preserve tangential continuity, $\vec{v}$ must vanish on the other three faces of the tetrahedron. On face Δ_{123}

$$\vec{v} \times \nabla\lambda_0 \Big|_{\lambda_0=0} = 0.$$

Since

$$\nabla\lambda_i \times \nabla\lambda_j = \begin{cases} 0, & i = j, \\ \vec{l}_{kl}/V, & i \neq j, \end{cases}$$

where V is the volume of the tetrahedron, this gives

$$\frac{1}{V}(g\vec{l}_{32} + h\vec{l}_{13})|_{\lambda_0=0} = 0.$$

Therefore, g and h must include a factor of λ_0. By the same reasoning, on face Δ_{023}, f and h must include a factor of λ_1, and on Δ_{013} f and g must have a factor of λ_2. Consequently, the admissible face-associated vector has the form

$$\vec{v} = s(\lambda_0, \lambda_1, \lambda_2)\lambda_1\lambda_2\nabla\lambda_0 + t(\lambda_0, \lambda_1, \lambda_2)\lambda_0\lambda_2\nabla\lambda_1 + u(\lambda_0, \lambda_1, \lambda_2)\lambda_0\lambda_1\nabla\lambda_2, \tag{5}$$

where s, t,and u are polynomials of degree $p-2$. The tangential components of (5) vanish at all points on all of the faces of the tetrahedron except for the interior region of face Δ_{012}. As long as the coefficients of (5) are common on the face shared by the two adjacent tetrahedra, (5) can be added to a tangential vector without altering its tangential continuity. The number of independent variables in (5) is $\#(P_f^{p-2}) \times 3 = 3\binom{p+1}{2}$.

4.1.3. Volume-associated vectors. Along the lines of the above derivation, the admissible volume-associated vector in the tetrahedron has the form

$$\begin{aligned} \vec{v} &= s(\lambda_0, \lambda_1, \lambda_2, \lambda_3)\lambda_1\lambda_2\lambda_3\nabla\lambda_0 + t(\lambda_0, \lambda_1, \lambda_2, \lambda_3)\lambda_0\lambda_2\lambda_3\nabla\lambda_1 \\ &\quad + u(\lambda_0, \lambda_1, \lambda_2, \lambda_3)\lambda_0\lambda_1\lambda_3\nabla\lambda_2 + w(\lambda_0, \lambda_1, \lambda_2, \lambda_3)\lambda_0\lambda_1\lambda_2\nabla\lambda_3, \end{aligned} \tag{6}$$

where s, t, u, and w are polynomials of degree $p-3$. Since the tangential components of (6) vanish on all points on all of the faces of the tetrahedron, (6) can be added to a tangential vector without altering its tangential continuity. Vector (6) has no direct interaction with the adjacent tetrahedra and hence can be eliminated to reduce the total number of unknowns [37].

For reasons to be discussed later, in three dimensions we will use only the following subspace of the vectors in (6):

$$\begin{aligned}\vec{v} &= \bar{s}(\lambda_1,\lambda_2,\lambda_3)\lambda_1\lambda_2\lambda_3\nabla\lambda_0 + \bar{t}(\lambda_0,\lambda_2,\lambda_3)\lambda_0\lambda_2\lambda_3\nabla\lambda_1 \\ &\quad + \bar{u}(\lambda_0,\lambda_1,\lambda_3)\lambda_0\lambda_1\lambda_3\nabla\lambda_2 + \bar{w}(\lambda_0,\lambda_1,\lambda_2)\lambda_0\lambda_1\lambda_2\nabla\lambda_3 \\ &\quad + \lambda_0\lambda_1\lambda_2\lambda_3 \begin{bmatrix} \bar{f}(\lambda_0,\lambda_1,\lambda_2,\lambda_3)(\nabla\lambda_0 - \nabla\lambda_1) \\ + \bar{g}(\lambda_0,\lambda_1,\lambda_2,\lambda_3)(\nabla\lambda_2 - \nabla\lambda_3) \\ + \bar{h}(\lambda_0,\lambda_1,\lambda_2,\lambda_3)(\nabla\lambda_2 + \nabla\lambda_3) \end{bmatrix},\end{aligned} \tag{6'}$$

where $\bar{s}$, $\bar{t}$, $\bar{u}$, and $\bar{w}$ are homogeneous polynomials of degree $p-3$, and $\bar{f}$, $\bar{g}$, and $\bar{h}$ are homogeneous polynomials of degree $p-4$. The number of independent variables in (6′) is

$$\#(\bar{P}_f^{p-3}) \times 4 + \#(\bar{P}_v^{p-4}) \times 3 = 4\binom{p-1}{2} + 3\binom{p-1}{3}.$$

4.2. Admissible gradient vectors of order p.

4.2.1. Edge-associated vectors. Edge-associated gradient vectors can be written as $\nabla P(\lambda_0,\lambda_1)$, where P is a polynomial of degree $p+1$. Comparing $\nabla P(\lambda_0,\lambda_1) = \frac{\partial P}{\partial\lambda_0}\nabla\lambda_0 + \frac{\partial P}{\partial\lambda_1}\nabla\lambda_1$ with (4), we must have $\frac{\partial P}{\partial\lambda_0} = s\lambda_1$ and $\frac{\partial P}{\partial\lambda_1} = t\lambda_0$. Therefore, P must include factors of λ_0 and λ_1. Thus admissible edge-associated gradient vectors have the form

$$\nabla P(\lambda_0,\lambda_1) = \nabla(\lambda_0\lambda_1 h(\lambda_0,\lambda_1)), \tag{7}$$

where h is a polynomial of degree $p-1$. The number of independent variables in (7) is $\#(P_e^{p-1}) = \binom{p+1}{2}$.

4.2.2. Face-associated vectors. By the same reasoning, admissible face-associated gradient vectors have the form

$$\nabla P(\lambda_0,\lambda_1,\lambda_2) = \nabla(\lambda_0\lambda_1\lambda_2 h(\lambda_0,\lambda_1,\lambda_2)), \tag{8}$$

where h is a polynomial of degree $p-2$. The number of independent variables in (8) is $\#(P_f^{p-2}) = \binom{p+1}{2}$.

4.2.3. Volume-associated vectors. In the same way, admissible volume-associated gradient vectors have the form

$$\nabla P(\lambda_0,\lambda_1,\lambda_2,\lambda_3) = \nabla(\lambda_0\lambda_1\lambda_2\lambda_3 h(\lambda_0,\lambda_1,\lambda_2,\lambda_3)), \tag{9}$$

where h is a polynomial of degree $p-3$. As was the case with volume-associated admissible vectors, instead of (9) we use the subspace vectors

$$\nabla P(\lambda_0,\lambda_1,\lambda_2,\lambda_3) = \nabla(\lambda_0\lambda_1\lambda_2\lambda_3)\bar{h}(\lambda_0,\lambda_1,\lambda_2,\lambda_3), \tag{9'}$$

where $\bar{h}$ is a homogeneous polynomial of degree $p-3$. The number of independent variables in (9′) is $\#(\bar{P}_v^{p-3}) = \binom{p}{3}$.

4.3. Symmetrization. Let us define two operators, $O_s(i,j)$ and $O_a(i,j)$. $O_s(i,j)$ symmetrizes a token by adding its symmetric complement as follows:

$$O_s(i,j)f(\dots\lambda_i,\dots\lambda_j,\dots) = f(\dots\lambda_i,\dots\lambda_j,\dots) + f(\dots\lambda_j,\dots\lambda_i,\dots);$$

$O_a(i,j)$ antisymmetrizes a token by subtracting its symmetric complement as follows:

$$O_a(i,j)f(\dots\lambda_i,\dots\lambda_j,\dots) = f(\dots\lambda_i,\dots\lambda_j,\dots) - f(\dots\lambda_j,\dots\lambda_i,\dots).$$

4.3.1. Edge-associated vectors. Given (4) or (7), we compute $O_s(0,1)\ \vec{v}\ (\lambda_0,\lambda_1)$ and $O_a(0,1)\ \vec{v}\ (\lambda_0,\lambda_1)$.

4.3.2. Face-associated vectors. Given (5) or (8), we compute $O_s(0,1)\ \vec{v}\ (\lambda_0,\lambda_1,\lambda_2)$ and $O_a(0,1)\ \vec{v}\ (\lambda_0,\lambda_1,\lambda_2)$. The face-associated vectors are not symmetrized or antisymmetrized with respect to λ_2.

4.3.3. Volume-associated vectors. Given (6) or (9), we have the following four possibilities:

$$\begin{gathered}
O_s(0,1)O_s(2,3)\ \vec{v}\ (\lambda_0,\lambda_1,\lambda_2,\lambda_3),\\
O_s(0,1)O_a(2,3)\ \vec{v}\ (\lambda_0,\lambda_1,\lambda_2,\lambda_3),\\
O_a(0,1)O_s(2,3)\ \vec{v}\ (\lambda_0,\lambda_1,\lambda_2,\lambda_3),\\
O_a(0,1)O_a(2,3)\ \vec{v}\ (\lambda_0,\lambda_1,\lambda_2,\lambda_3).
\end{gathered}$$

4.4. Procedure for constructing hierarchical vector bases of order p.

4.4.1. The procedure. The following algorithm may be used to generate the correct vector basis:

For $i = 1$ to p do {

1(a) If $i >= 3$, start with $(9')$ and construct G_v^i such that

$$G_v^i \perp G_v^{I-1}. \tag{10}$$

1(b) If $i >= 3$, start with $(6')$ and construct R_v^i such that

$$R_v^i \perp (G_v^i \cup E_v^{I-1}). \tag{11}$$

2(a) If $i >= 2$, start with (8) and construct G_f^i such that

$$G_f^i \perp (G_f^{I-1} \cup G_v^I). \tag{12}$$

2(b) If $i >= 2$, start with (5) and construct R_f^i such that

$$R_f^i \perp (G_f^i \cup E_f^{I-1} \cup E_v^I). \tag{13}$$

3(a) Start with (7) and construct G_e^i such that

$$G_e^i \perp (G_e^{I-1} \cup G_f^I). \tag{14}$$

3(b) Start with (4) and construct R_e^i such that

Table 1
Edge-associated basis components.

	Coeff. of $\lambda_1\nabla\lambda_0$	Coeff. of $\lambda_0\nabla\lambda_1$
R^1	1	-1
G^1	1	1
G^2	$2\lambda_0 - \lambda_1$	$-(2\lambda_1 - \lambda_0)$
G^3	$1086\lambda_0^2 + 362\lambda_1^2 - 1008\lambda_0\lambda_1$ $-234\lambda_0 - 117\lambda_1 - 16$	$1086\lambda_1^2 + 362\lambda_0^2 - 1008\lambda_0\lambda_1$ $-234\lambda_1 - 117\lambda_0 - 16$

$$R_e^i \perp (G_e^i \cup E_e^{I-1} \cup E_f^I). \tag{15}$$

}

Note the following points about this procedure.

1. One needs to construct the volume-associated basis first, because the face-associated basis is constructed by making it orthogonal to the volume-associated one. Similarly, the face-associated basis is constructed before the edge-associated one.

2. As mentioned before, to maximize orthogonality, the constructed G_e^i, R_e^i, G_f^i, R_f^i, G_v^i, and R_v^i are all made ST-orthogonal in each step. This is done by computing all of the eigenvectors of the local generalized (S, T) eigenvalue problem and using the full set of eigenvectors in the local element as the new basis. This new basis makes both the local S and T matrices diagonal. By the definition of S and T, this means that we generate a double orthogonality: not only is the new basis orthogonal over the tetrahedron, the curls of the new basis functions are also orthogonal. For a gradient space, the S matrix vanishes, and orthogonality is required only between the vectors themselves. However, in the case of two dimensional waveguide problems where the field consists of transverse and longitudinal components, one employs vector bases to approximate the transverse components and scalar bases to approximate the longitudinal components. In this case, one may want to derive gradient vector bases from a set of mutually orthogonal scalar bases.

3. The volume-associated vector basis is constructed differently from the edge-associated and the face-associated ones. It starts with a homogeneous polynomial instead of a polynomial of complete degree. This restricts its orthogonality with higher-dimensional spaces, i.e., four-dimensional space, but does not affect its orthogonality with lower-dimensional ones. If desired for some reason, one could construct the four-dimensional components of the basis first and then construct the volume-associated components using (6) and (9).

The computed bases up to third order are given in Tables 1–6. Notice in Table 1 that the coefficients of G_e^3 are not homogeneous. If one starts with a homogeneous polynomial, it is not possible to simultaneously make it orthogonal to the gradient space of lower order and to the higher dimensional gradient space.

4.4.2. Finite element vector spaces. Since a tetrahedron has six edges, four faces, and one volume, E^i can be expressed as

$$E^i = 6E_e^I \cup 4E_f^I \cup E_v^I. \tag{16}$$

The accuracy of S is two orders lower than that of T; thus S dominates the accuracy of A. Since missing gradient bases does not affect the accuracy of S, one may without

Table 2
Face-associated basis components.

	Coeff. of $\lambda_1\lambda_2\nabla\lambda_0$	Coeff. of $\lambda_0\lambda_2\nabla\lambda_1$	Coeff. of $\lambda_0\lambda_1\nabla\lambda_2$
G^2	1	1	1
R^2	1	1	-2
	1	-1	0
G^3	$2\lambda_0 + \lambda_1 - 2\lambda_2$	$2\lambda_1 + \lambda_0 - 2\lambda_2$	$\lambda_0 + \lambda_1 - 4\lambda_2$
	$2\lambda_0 - \lambda_1$	$-(2\lambda_1\lambda_0)$	$\lambda_0 - \lambda_1$
R^3	$\lambda_1 - \lambda_2$	$-(\lambda_0 - \lambda_2)$	$\lambda_0 - \lambda_1$
	$393\lambda_0 + 80\lambda_1 - 212\lambda_2$	$-(393\lambda_1 + 80\lambda_0 - 212\lambda_2)$	$-292\lambda_0 + 292\lambda_1$
	$-131\lambda_0 + 168\lambda_1 - 124\lambda_2$	$-131\lambda_1 + 168\lambda_0 - 124\lambda_2$	$-44\lambda_0 - 44\lambda_1$ $+262\lambda_2$

Table 3
Volume-associated basis components.

	Coeff. of $\lambda_1\lambda_2\lambda_3\nabla\lambda_0$	Coeff. of $\lambda_0\lambda_2\lambda_3\nabla\lambda_1$	Coeff. of $\lambda_0\lambda_1\lambda_3\nabla\lambda_2$	Coeff. of $\lambda_0\lambda_1\lambda_2\nabla\lambda_3$
G_3	1	1	1	1
R_3	1	1	-1	-1
	0	0	1	-1
	1	-1	0	0

penalty decrease the accuracy of T by removing a gradient space of order i. This results in an associated space, $E^{i-1}(curl)$:

$$E^{i-1}(curl) = E^i - (6G_e^i \cup 4G_f^i \cup G_v^i). \tag{17}$$

Similarly, for a finite element mesh, we have

$$\begin{aligned} F^i &= \#(e)E_e^I \cup \#(f)E_f^I \cup \#(v)E_v^I, \\ F^{i-1}(curl) &= F^i - (\#(e)G_e^i \cup \#(f)G_f^i \cup \#(v)G_v^i). \end{aligned} \tag{18}$$

4.5. Further decomposition of $\boldsymbol{E^0(curl)}$. $E^0(curl)$ is equal to R_e^1. Strictly speaking, R_e^1 is only locally rotational, and it can have a global gradient subspace. By using the scalar-vector formulation in [32], this space may be split into two subspaces R_{ce}^1 and $\nabla\phi^1$, where R_{ce}^1 is a cotree subspace of R_e^1, and ϕ^1 is the first-order nodal basis

$$R_e^1 = R_{ce}^1 \cup \nabla\phi^1. \tag{19}$$

The definition of the cotree space and the reason that it resolves convergence problems at low frequency is provided in [34]. We emphasize that separating the gradient space from the rotational space for additional orders empowers us to deal with only R_e^1. In addition, $\nabla\phi^1$ is not orthogonal to R_{ce}^1 in (19), and R_{ce}^1 still contains significant low frequency components. Incomplete Cholesky decomposition is not a good preconditioner because of this.

5. Basis count. Since the higher dimensional space is constructed before the lower dimensional space, we count the number of basis vectors in three dimensions first.

Table 4
The effects of mesh refinement and increasing basis order on the number of MPCG iterations at 10 *GHz.*

Element size	Basis order			
	0	1	2	3
h	1	4	9	14
h/2	1	10	14	17
h/4	1	11	16	20
h/8	1	12	13	16

Table 5
Number of nonzero entries per row.

	Basis order			
Element size	0	1	2	3
h	2.4	12.8	32.6	63.2
h/2	10.6	28	56.2	102.8
h/4	16.9	39	71.2	124.8
h/8	21.4	44.6	78.6	135.4
Eq. (22)	26	50	90	149

Table 6
Number of unknowns per tetrahedra.

	Basis order			
Element size	0	1	2	3
h	0.3	2.8	10.8	27
h/2	0.6	4.4	14.4	33.5
h/4	0.9	5.3	16.4	37.0
h/8	1.0	5.8	17.4	38.8
Eq. (20)	1.16	6.32	18.48	40.64

5.1. Volume-associated vectors.

5.1.1. G_v^i. Since the volume-associated gradient space is constructed from (9′) and satisfies (10), it follows that

$$\begin{aligned}\#(G_v^i) &= \#(\bar{P}_v^{i-3}) - \#(\bar{P}_v^{i-4}) \\ &= \binom{i}{3} - \binom{i-1}{3} \\ &= \frac{1}{2}(i-1)(i-2)\end{aligned}$$

and

$$\#(G_v^I) = \binom{i}{3}.$$

5.1.2. $\boldsymbol{R_v^i}$. Since the volume-associated rotational space is constructed from (6′) and satisfies (11), it follows that

$$\begin{aligned}\#(R_v^i) &= \#(\bar{P}_f^{i-3}) \times 4 + \#(\bar{P}_v^{i-4}) \times 3 - [\#(G_v^i) + \#(\bar{P}_f^{i-4}) \times 4 + \#(\bar{P}_v^{i-5}) \times 3] \\ &= i(i-2)\end{aligned}$$

and

$$\#(E_v^I) = \frac{1}{2}(i-2)(i-1)(i+1).$$

5.2. Face-associated vectors.

5.2.1. $\boldsymbol{G_f^i}$. Since the face-associated gradient space is constructed from (8) and satisfies (12), it follows that

$$\begin{aligned}&\#(G_f^i) = \#(P_f^{i-2}) - [\#(G_f^{I-1}) + \#(G_v^I)] \\ &\Rightarrow \#(G_f^I) = \frac{1}{2}i(i-1) \\ &\Rightarrow \#(G_f^i) = i-1.\end{aligned}$$

5.2.2. $\boldsymbol{R_f^i}$. Since the face-associated rotational space is constructed from (5) and satisfies (13), it follows that

$$\begin{aligned}&\#(R_f^i) = \#(P_f^{i-2}) \times 3 - [\#(G_f^I) + \#(E_f^{I-1}) + \#(E_v^I)] \\ &= \ 3\binom{i+1}{3} - [\#(G_f^I) + \#(R_f^{I-1}) + \#(E_v^I)] \\ &\quad \Rightarrow \ \#(R_f^I) = \frac{1}{2}(i-1)(i+2) \\ &\quad \Rightarrow \ \#(R_f^i) = i\end{aligned}$$

and

$$\#(E_f^I) = (i-1)(i+1).$$

5.3. Edge-associated vectors.

5.3.1. $\boldsymbol{G_e^i}$. Since the edge-associated gradient space is constructed from (7) and satisfies (14), it follows that

$$\begin{aligned}&\#(G_e^i) = \#(P_e^{i-1}) - [\#(G_e^{I-1}) + \#(G_f^I)] \\ &\Rightarrow \#(G_e^I) = i \\ &\Rightarrow \#(G_e^i) = 1.\end{aligned}$$

5.3.2. $\boldsymbol{R_e^i}$. Since the edge-associated rotational space is constructed from (4) and satisfies (15), it follows that

$$\begin{aligned}&\#(R_e^i) = \#(P_e^{i-1}) \times 2 - [\#(G_e^i) + \#(E_e^{I-1}) + \#(E_f^I)] \\ &= \ 2\binom{i+1}{2} - [\#(G_e^I) + \#(R_e^{I-1}) + \#(E_f^I)] \\ &\Rightarrow \#(R_e^I) = 1 \\ &\Rightarrow \#(R_e^i) = \begin{cases} 1, & i = 1, \\ 0, & i \geq 2, \end{cases}\end{aligned}$$

and

$$\#(E_e^I) = i + 1.$$

5.4. E^i and $E^{i-1}(curl)$. From (16), we see that

$$\begin{aligned}\#(E^i) &= 6 \times \#(E_e^I) + 4 \times \#(E_f^I) + \#(E_v^I)\\ &= \frac{1}{2}(i+1)(i+2)(i+3),\end{aligned}$$

and also from (17) we observe

$$\begin{aligned}\#(E^{i-1}(curl)) &= \#(E^i) - [6 \times \#(G_e^i) + 4 \times \#(G_f^i) + \#(G_v^i)]\\ &= \frac{1}{2}i(i+2)(i+3).\end{aligned}$$

The numbers of basis functions thus computed are identical to those of Nedelec [1].

5.5. F^i and $F^{i-1}(curl)$. From (18), we find that

$$\begin{aligned}&\#(F^i) = \#(e) \times \#(E_e^I) + \#(f) \times \#(E_f^I) + \#(v) \times \#(E_v^I),\\ &\#(F^{i-1}(curl)) = \#(F^i) - [\#(e) \times \#(G_e^i) + \#(f) \times \#(G_f^i) + \#(v) \times \#(G_v^i)].\end{aligned}$$

According to [7], ignoring Dirichlet boundaries, an ideal Delaunay tessellation in three dimensions obeys the following approximate relationships:

$$\begin{aligned}\#(e) &\approx \#(v) \times 1.16,\\ \#(f) &\approx \#(v) \times 2.\end{aligned}$$

Consequently, the total number of degrees of freedom is linearly proportional to $\#(v)$ and can be estimated from the following equations:

$$\begin{aligned}\#(F^i) &\approx \#(v) \times \frac{1}{2}(i+1)(i^2+i+0.32),\\ \#(F^{i-1}(curl)) &\approx \#(v) \times \frac{1}{2}i(i^2+i+0.32).\end{aligned} \tag{20}$$

As discussed earlier, using F^i does not result in a more accurate solution. $F^{i-1}(curl)$ is favored because it contains fewer unknowns. Notice, however, that the ratio $(i+1)/i$ is not significant with higher orders.

When the volume unknowns are eliminated, $\#(F^i)$ and $\#(F^{i-1}(curl))$ are greatly reduced to

$$\begin{aligned}\#(F^i) &\approx \#(v) \times (i+1)(2i-0.84),\\ \#(F^{i-1}(curl)) &\approx \#(v) \times i(2i-0.84).\end{aligned} \tag{21}$$

One can also estimate the number of nonzero entries by assuming an ideal Delaunay tessellation, where an edge is shared by 5 connecting tetrahedra forming a saucer. This edge interacts with the surrounding 26 edges (including itself), 15 faces, and 5 tetrahedra. Because a face is shared by 2 tetrahedra, it interacts with 9 edges, 7 faces (including itself), and 2 tetrahedra. A volume unknown interacts with 6 edges, 4 faces, and itself. Therefore, we have the following formula for the number of nonzero entries for $\#(F^{i-1}(curl))$:

$$NZ \approx \left(1.16i \times NZ_e + 2(i-1)i \times NZ_f + \frac{1}{6}(i-1)(i-2)(2i+3) \times NZ_v\right) \#(v). \tag{22}$$

NZ_e, NZ_f, and NZ_v and are calculated as

$$\begin{aligned} NZ_e &= 26i + 15(i-1)i + \frac{5}{6}(i-1)(i-2)(2i+3), \\ NZ_f &= 9i + 7(i-1)i + \frac{2}{6}(i-1)(i-2)(2i+3), \\ NZ_v &= 6i + 4(i-1)i + \frac{1}{6}(i-1)(i-2)(2i+3). \end{aligned}$$

6. Multilevel preconditioned conjugate gradient method.

In this work, we employ Schwarz methods, often used in the domain decomposition area, and apply them to form an efficient preconditioner for the conjugate gradient algorithm with p-type finite elements. In particular, we use a multiplicative Schwarz preconditioner. This structure can be viewed as an overlapping block Gauss–Seidel preconditioner. Even without conjugate gradient acceleration, the multiplicative method can take far fewer iterations than the additive version. This theory is provided in [35]. Conventionally, multilevel methods are associated with a nested grid that employs a multilevel of grids [36]. In this paper, the multilevel algorithm employs a single grid but a multilevel of basis functions. We may think of the approach presented here as a p-refinement multilevel method [37], [38] instead of as the more traditional h-refinement multilevel method, where p refers to the order of the element and h refers to the element size. We employ Schur factorization to obtain an approximate inverse of the system matrix and treat it as a preconditioner in the conjugate gradient method. It can be proved that the current approach is equivalent to a V-cycle multigrid method. An advantage of the current approach is that it provides a better understanding of the approximation made in computing the preconditioner. We will call the resulting procedure the MPCG method.

Numbering unknowns from the lowest level to the highest, i.e., from 1 to p, the system matrix A_P has the following block structure:

$$A_P = \begin{bmatrix} A_{1,1} & A_{1,2} & \dots & A_{1,p} \\ A_{2,1} & A_{2,2} & & \vdots \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & \\ A_{p,1} & \dots & \dots & A_{p,p} \end{bmatrix}.$$

As in section 4, capital index letters denote the entire hierarchy, and lower case letters denote the additional unknowns. To utilize the multilevel concept, we write the above in two-level recursive form:

$$A_P = \begin{bmatrix} A_{P-1} & A_{P-1,p} \\ A_{p,P-1} & A_{p,p} \end{bmatrix}.$$

The above matrix is factorized by Schur factorization:

$$A_P = \begin{bmatrix} A'_{P-1} & A_{P-1,p} \\ 0 & A_{p,p} \end{bmatrix} \begin{bmatrix} I & 0 \\ A_{p,p}^{-1} A_{p,P-1} & I \end{bmatrix}, \tag{23}$$

$$A'_{P-1} = A_{P-1} - A_{P-1,p} A_{p,p}^{-1} A_{p,P-1}.$$

When A'_{P-1} is approximated by $B_{P-1} = A_{P-1}$ and the $A_{p,p}$'s are approximated by an incomplete Cholesky decomposition with thresholding ILU, i.e., $A_{p,p} \approx B_{p,p} = LDL^T$, we obtain a multiplicative preconditioner

$$B_P = \begin{bmatrix} B_{P-1} & A_{P-1,p} \\ 0 & B_{p,p} \end{bmatrix} \begin{bmatrix} I & 0 \\ B_{p,p}^{-1} A_{p,P-1} & I \end{bmatrix}.$$

We use ILU preconditioners to take advantage of the orthogonality of the basis functions. When $A_{P-1,p}$ and $B_{p,p}^{-1} A_{p,P-1}$ are further removed, we obtain a less effective additive preconditioner. Although the multiplicative preconditioner requires approximately one additional matrix multiplication in each conjugate gradient iteration (three matrix vector multiplications versus two for the additive preconditioner), the saving on the number of conjugate gradient iterations well compensates this cost. Numerical experiments show that for $A_{p,p}$, a thresholding constant of 0.01 is a good choice.

Matrix A_p has two sets of negative eigenvalues: physical and nonphysical [25], [26], [27]. Nonphysical ones correspond to the gradient space. Physical ones are the resonance modes of the system with resonance frequency less than the prescribed frequency. As discussed in section 3.2, nonphysical ones from p greater or equal to 1 can be easily preconditioned to positive eigenvalues. However, for $E^0(curl)$, the complete separation of gradient and rotational spaces is not done. In order for MPCG to converge for indefinite problems, $A_{1,1}$ has to be decomposed exactly. Otherwise, the preconditioned matrix will remain highly indefinite such that MPCG will not converge. Also, the mesh has to be fine enough such that $A_{1,1}$ accurately contains the low frequency spectrum below the prescribed frequency.

7. Numerical results.

Since $E^p(curl)$ is computationally advantageous, we will focus on studying it numerically. All computations are performed with a Pentium II, 550 MHz processor PC. First, we demonstrate the accuracy of the elements by computing the lowest eigenvalue of a rectangular cavity of dimensions $8\times10\times16$ mm^3. We choose this problem because the fields in the cavity are smooth. The eigenvalues are computed using the Lanczos algorithm implemented with MPCG. The cavity is first split into two boxes along the longest dimension, and each box is then broken into six tetrahedra. This gives the coarse mesh. Splitting the boxes into eight boxes of equal size gives the next finer mesh, and so on. Therefore, the tetrahedra in the mesh are not perfectly regular, but they are well shaped. To speed computation of the system matrix, we precomputed the set of constant matrices [21]. For MPCG, we start with a zero initial guess and allow the residual norm to go down to less than 1e-4 of the norm of the right-hand side vector. If the discretization error of solution is less than 1e-4, we let the residuals go two orders of magnitude less than the discretization error. The results in Figures 1–5 show the normalized error in the computed eigenvalues varying the basis order and the element size. P-refinement is clearly superior to h-refinement and is therefore preferred wherever the solution is

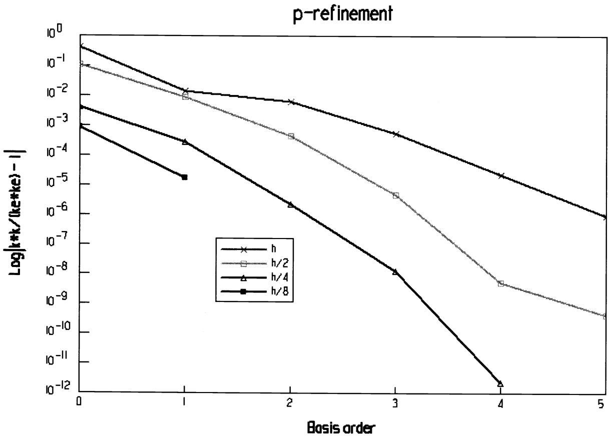


FIG. 1. *Convergence with respect to p-refinement, k_e is the exact eigenvalue.*

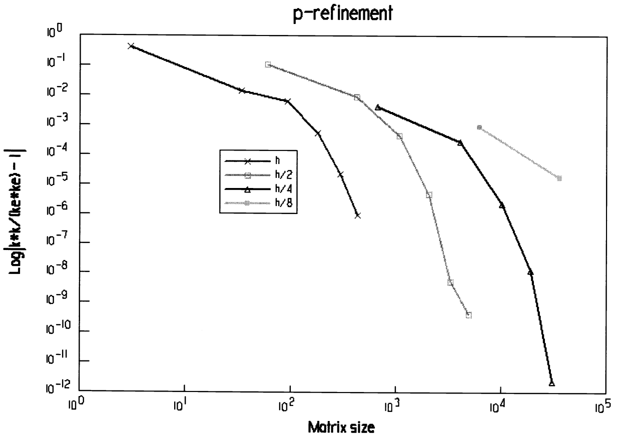


FIG. 2. *Convergence of p-refinement plotted versus matrix size.*

smooth. As shown in Figure 5 and also expressed in (21), the cost of going to higher order is small, especially at already high orders. The convergence rates of each basis order can be computed from the slopes of Figure 3. These agree with the optimal theoretical rates of $2 \times p + 2$.

Next, we demonstrate the effectiveness of MPCG. The rectangular box of the previous example is now treated as a waveguide by sending the TE_{01} mode through the faces with dimensions 8×16 mm^2. As shown in Figure 6, although the residual curve is bumpy, it reaches machine precision without stagnation. Table 7 exhibits an almost constant number of iterations, regardless of the mesh fineness or of the basis order. In Figure 7, we demonstrate the effect of increasing frequency by several or-

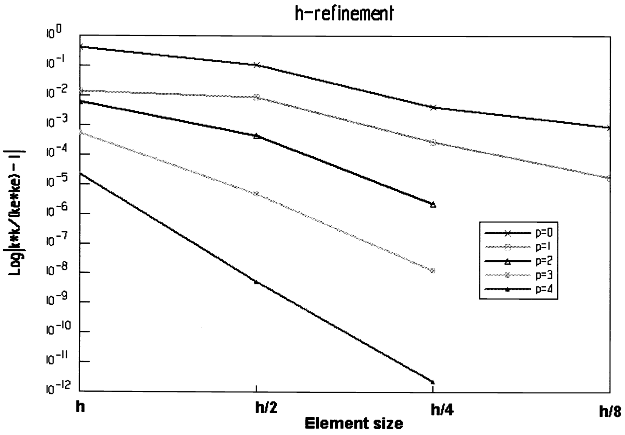


FIG. 3. *Convergence versus h-refinement.*

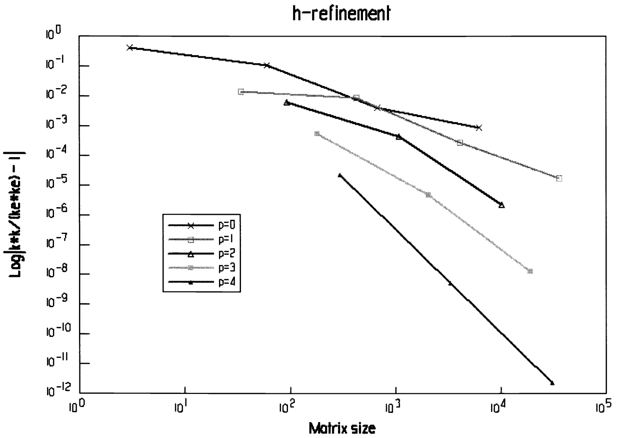


FIG. 4. *Convergence of h-refinement plotted versus matrix size.*

ders of magnitude. As expected, at low frequencies, the number of iterations is almost constant. However, at higher frequencies, when the element size exceeds a quarter wavelength, MPCG starts to oscillate irregularly. At extremely high frequencies, the finite element solutions are, of course, meaningless since the elements are relatively large compared to the wavelength of the field. Nevertheless, the matrix A approaches the positive definite matrix T, and MPCG again converges quickly. Figure 8 compares h- and p-refinement in terms of CPU time. Again, p-refinement is clearly superior. In Figure 9, we separate the portions of time spent on CG iterations and on the decomposition of $A_{1,1}$. This figure shows that another advantage of p-refinement is that the time spent on the full decomposition of $A_{1,1}$ stays constant as p increases. With

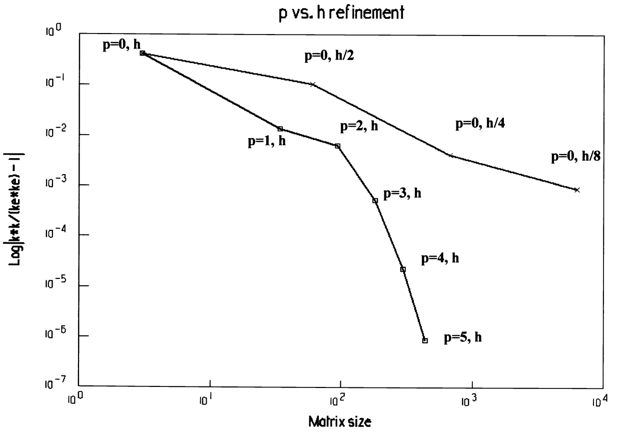


FIG. 5. *Comparison of p- and h-refinement for the cavity problem.*

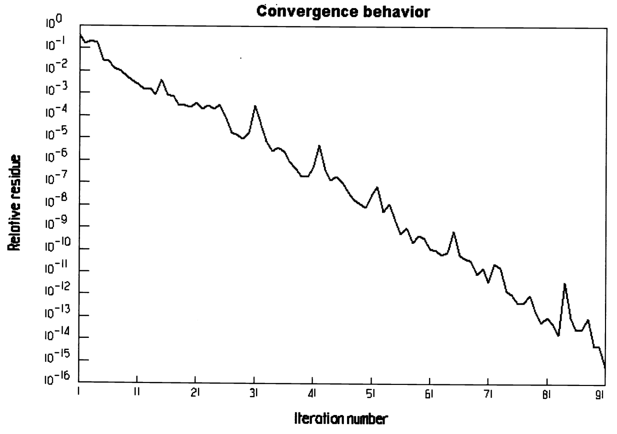


FIG. 6. *The convergence behavior of MPCG for E^1(curl) at* 10 *GHz.*

the problem sizes reported here, CG dominates the computational time. However, the decomposition of $A_{1,1}$ will become the bottleneck with much larger problems. Research on eliminating full decomposition of $A_{1,1}$ is underway.

Going to higher orders is not without cost. The matrices are denser as seen from Table 5. Despite this, the higher rates of convergence are worthwhile. We compare the estimated number of nonzeros per row from (22) with real ones. Real ones approach the estimated ones for a larger mesh, where the unknowns on Dirichlet boundaries are only a small portion of the entire domain. The number of unknowns per tetrahedron is also compared to the estimate from (20) in Tables 3–6.

To examine the reasons why MPCG works so well for indefinite systems, we re-

TABLE 7
Comparison between the condition numbers of the original (O) *and preconditioned* (P) *matrices for a well-shaped and for a distorted mesh at* 10 *GHz.*

	Basis order					
	0		1		2	
Element size	O	P	O	P	O	P
h	3.1	1	1.6e2	8.2	3.2e6	1.5e1
h/2	2.2e2	1	4.4e2	1.5e1	9.6e6	6.0e1
distorted	1.4e4	1	3.3e4	4.5e1	1.0e9	1.4e2

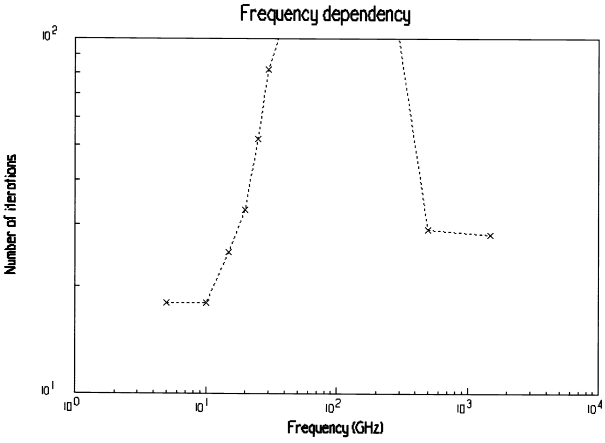


FIG. 7. *The effect of varying frequency on the number of MPCG iterations for $E^1(curl)$.*

visit the last problem and study the spectrum of original and preconditioned matrices. The results are presented in Table 7 and in Figures 10 and 11. Although the original matrix is highly indefinite due to the presence of the gradient space, at 10 GHz it is preconditioned into a positive-definite matrix. The condition number of preconditioned matrix grows at a much slower rate than those of the original matrix, as shown in Table 7. The eigenvalues of the original matrix are real because it is a real symmetric matrix. On the other hand, the preconditioned matrix is not symmetric, and therefore its eigenvalues are either real or in complex conjugate pairs. The three clusters in Figure 10(b) help to explain why MPCG converges better than what would be expected from the computed condition numbers. At higher frequencies, i.e., at 35 GHz in Figure 11, the preconditioned matrix is no longer positive definite, and therefore MPCG starts to oscillate irregularly.

Our derivation is based on a regular tetrahedron, so that the basis vectors derived above are in general not orthogonal in tetrahedra of arbitrary shape. In complex problems, the tetrahedra are rarely regular. Thus, the matrix generated by using the derived basis functions is not less sparse than that obtained by using nonorthogonal bases. However, we have found empirically that the fill-in generated by the incomplete Cholesky decomposition is greatly reduced by using the orthogonal bases. Thus, while

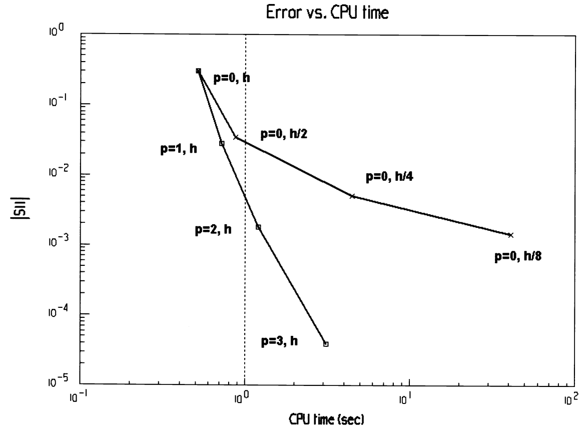


FIG. 8. *Comparison of h- and p-refinement for a waveguide problem.*

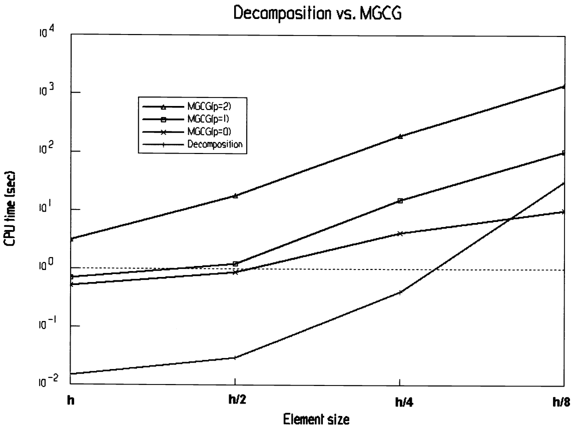
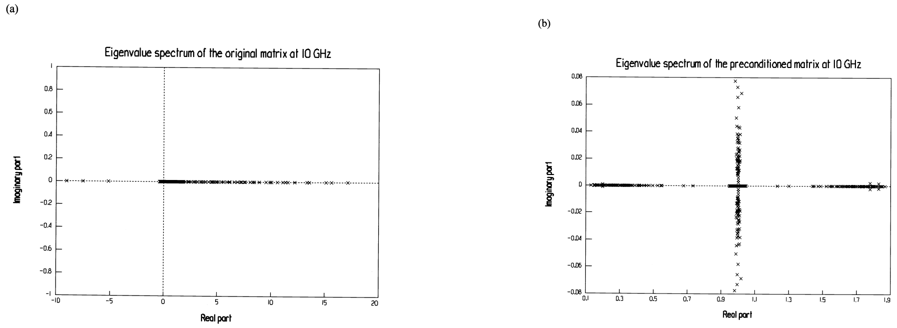


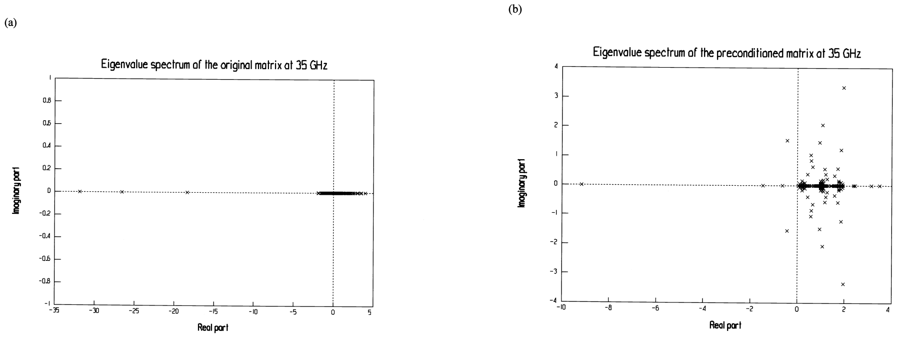
FIG. 9. *Computational complexity.*

we have not obtained diagonal matrices, we have obtained matrices that are much more diagonally dominant. Even with an arbitrary mesh, the number of iterations increases only slightly from low to high order, suggesting that the basis vectors are quite orthogonal although not perfectly so. In addition, refining the mesh does not increase the number of iterations. On the contrary, the number of iterations is reduced with mesh refinement in most cases. The reason for this is that the $E^0(curl)$ solution improves with mesh refinement and so fewer conjugate gradient iterations are required for convergence with the higher-order solutions. However, if one overrefines the mesh, the increase in the condition number can make MPCG nonconverging.



(a) (b)

FIG. 10. *Spectrum of the original and preconditioned matrices for the h/2 mesh and* $p = 1$ *at* 10 *GHz.*

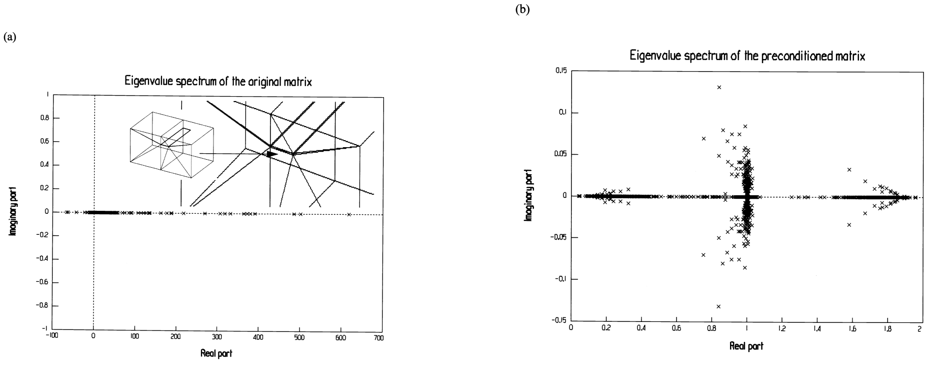


(a) (b)

FIG. 11. *Spectrum of the original and preconditioned matrices for a h/2 mesh and* $p = 1$ *at* 35 *GHz.*

Finally, we study the effect of severely distorted tetrahedra. We distort the mesh by inserting an artificial, rectangular strip of 0.1 mm thickness into the previous waveguide. As shown in the inserts in Figure 12(a), the surface mesh on the port is very badly distorted. Although we are able to precondition the problem into a positive definite matrix, its spectrum, shown in Figure 12(b), spreads out greatly. The condition numbers of the original matrices increase two orders of magnitude, but those of the preconditioned matrices do not, as shown in Table 7. This explains why the number of iterations only increases slightly, from 10 for the $h/2$ mesh, to 14 for $p = 1$, and up to 17 for $p = 2$. We compare the distorted mesh with the $h/2$ mesh because numbers of unknowns are closest in these cases.

If the mesh is not fine enough, the resulting preconditioned matrix is indefinite, and in this case, MPCG fails. There are two remedies for this problem: one is to refine the mesh, and the other is to include the $E^1(curl)$ bases in $A_{1,1}$. With mesh refinement or with enlargement of the lowest level matrix, the nonconverging problem is overcome. We monitor the diagonal entries in the Cholesky matrix to see if they have the opposite sign to the corresponding entries in the original matrix. If they do, this indicates that the Cholesky matrix has different spectral characteristics from the original matrix. Therefore, the decomposition is unstable, and a refinement or



(a) (b)

FIG. 12. *Spectrum of the original and preconditioned matrices for a distorted mesh and $p = 1$ at* 10 *GHz.*

enlargement is performed.

8. Conclusions. Tangential vector finite element basis functions are not unique and can be defined in many ways. This paper suggests that the best choice—indeed the optimal choice—is that basis set which results in the fastest convergence when the resulting system of equations is solved by using the conjugate gradient algorithm. The goal of constructing tangential vector finite element basis functions should therefore be to allow efficient preconditioning, and thus reduce the condition number of the system matrix. Computational efficiency results by using basis sets and associated preconditioners that speed up the conjugate gradient process.

We have shown that the optimal rate of convergence is obtained by using bases computed using a multistep process in which the admissible basis set is decomposed into gradient and rotational subspaces, and each vector is made orthogonal to the others by computing the eigenvectors of the element submatrix. This orthogonalization is performed on a regular tetrahedron and is therefore not strictly valid on tetrahedra of arbitrary shape. Empirical results, however, demonstrate that the orthogonality is largely preserved with meshes of arbitrary shape, and the resulting global finite element matrix is therefore well suited for solution with the conjugate gradient method. Numerical experiments show that the number of iterations needed for solution by MPCG is basically a constant, regardless of the order of the basis or of the matrix size. Computational speed is improved by several orders of magnitude due to the fast matrix solution of MPCG and to the high accuracy of the higher-order bases.

REFERENCES

[1] J. C. NEDELEC, *Mixed finite elements in R^3*, Numer. Math., 35 (1980), pp. 315–341.

[2] A. BOSSAVIT AND J. C. VERITE, *A mixed FEM-BIEM method to solve three-dimension eddy current problem*, IEEE Trans. Magn., 18 (1982), pp. 431–435.

[3] M. L. BARTON AND Z. J. CENDES, *New vector finite elements for three-dimensional magnetic field computations*, J. Appl. Phys., 61 (1987), pp. 3919–3921.

[4] S. H. WONG AND Z. J. CENDES, *Combined finite element modal solution of three-dimensional eddy current problems*, IEEE Trans. Magn., 24 (1988), pp. 2685–2687.

[5] S. H. WONG AND Z. J. CENDES, *Numerically stable finite element methods for the Galerkin solution of eddy current problems*, IEEE Trans. Magn., 25 (1989), pp. 3019–3021.

[6] Z. J. Cendes, *Vector finite elements for electromagnetic field computation*, IEEE Trans. Magn., 27 (1991), pp. 3953–3966.

[7] J.-F. Lee, D.-K. Sun, and Z. J. Cendes, *Tangential vector finite elements for electromagnetic field computation*, IEEE Trans. Magn., 27 (1991), pp. 4032–4035.

[8] D. Boffi, *Fortin operator and discrete compactness for edge elements*, Numer. Math., 87 (2000), pp. 229–246.

[9] D. Boffi, P. Fernandes, L. Gastaldi, and I. Perugia, *Computational models of electromagnetic resonators: Analysis of edge element approximation*, SIAM J. Numer. Anal., 36 (1999), pp. 1264–1290.

[10] S. Caorsi, P. Fernandes, and M. Raffetto, *On the convergence of Galerkin finite element approximations of electromagnetic eigenproblems*, SIAM J. Numer. Anal., 38 (2000), pp. 580–607.

[11] M. Costabel, M. Dauge, and S. Nicaise, *Singularities of Maxwell interface problems*, M2AN Math. Model. Numer. Anal., 33 (1999), pp. 627–649.

[12] P. Monk and L. Demkowicz, *Discrete compactness and the approximation of Maxwell's equations in R^3*, Math. Comp., 70 (2001), pp. 507–523.

[13] J. P. Webb and B. Forghani, *Hierarchical scalar and vector tetrahedra*, IEEE Trans. Magn., 29 (1993), pp. 1495–1498.

[14] P. Monk, *On the p and hp-extension of Nedelec's conforming elements in three dimensions*, J. Comput. Appl. Math., 53 (1994), pp. 117–137.

[15] D. Sun, J. Manges, X. Yuan, and Z. Cendes, *Spurious modes in finite element methods*, IEEE Antennas and Propagation Magazine, 37 (1995), pp. 12–24.

[16] J. S. Savage and A. F. Peterson, *Higher-order vector finite elements for tetrahedral cells*, IEEE Trans. Microwave Theory Tech., 44 (1996), pp. 874–879.

[17] R. Graglia, D. R. Wilton, and A. F. Peterson, *Higher order interpolatory vector bases for computational electromagnetics*, IEEE Trans. Antennas and Propagation, 45 (1997), pp. 329–342.

[18] L. S. Anderson and J. L. Volakis, *Hierarchical tangential vector finite elements for tetrahedra*, IEEE Microwave and Guide Wave Letter, 8 (1998), pp. 127–129.

[19] L. Demkowicz and L. Vardapetyan, *Modeling of electromagnetic absorption/scattering problems using hp-adaptive finite elements*, Comput. Methods Appl. Mech. Engrg., 152 (1998), pp. 103–124.

[20] L. Demkowciz, P. Monk, L. Vardapetyan, and W. Rachowicz, *De Rham Diagram for hp Finite Element Spaces*, Comput. Math. Appl., 39 (2000), pp. 29–38.

[21] J. P. Webb, *Hierarchal vector basis functions of arbitrary order for triangular and tetrahedral finite elements*, IEEE Trans. Antennas and Propagation, 47 (1999), pp. 1244–1253.

[22] R. Hiptmair, *Canonical construction of finite elements*, Math. Comp., 68 (1999), pp. 1325–1346.

[23] L. S. Anderson and J. L. Volakis, *Condition numbers for various FEM matrices*, in Proceedings of the IEEE Antennas and Propagation Society International Symposium, Orlando, FL, 1999, pp. 2614–2617.

[24] Y. Jiang and A. Martin, *Comparison of iterative convergence in higher-order FEM modeling of periodic structures*, in Proceedings of the URSI Radio Science Meeting, Orlando, FL, 1999, p. 312.

[25] R. Dyczij-Edlinger and O. Biro, *A joint vector and scalar potential formulation for driven high frequency problems using hybrid edge and nodal elements*, IEEE Trans. Microwave Theory and Techniques, 44 (1996), pp. 15–23.

[26] G. Peng, R. Dyczij-Edlinger, and J.-F. Lee, *Hierarchical methods for solving matrix equations from TVFEMs for microwave components*, IEEE Trans. Magn., 35 (1999), pp. 1474–1477.

[27] R. Dyczij-Edlinger, G. Peng, and J.-F. Lee, *Efficient finite element solvers for the Maxwell equations in the frequency domain*, Comput. Methods in Appl. Mech. and Engrg., 169 (1999), pp. 297–309.

[28] I. Babuska, M. Griebel, and J. Pitkaranta, *The problem of selecting the shape functions for a p-type finite element*, Internat. J. Numer. Methods Engrg., 28 (1989), pp. 1891–1908.

[29] N. Hu, X.-Z. Guo, and I. Katz, *Multi-p preconditioners*, SIAM J. Sci. Comput., 18 (1997), pp. 1676-1697.

[30] J. Mandel, *An iterative solver for p-version finite elements in three dimensions*, Comput. Methods Appl. Mech. Engrg., 116 (1994), pp. 175–183.

[31] P. S. Vassilevski and J. Wang, *Stabilizing the hierarchical basis by approximate wavelets* II: *Implementation and numerical results*, SIAM J. Sci. Comput., 20 (1998), pp. 490–514.

[32] R. Albanese and C. Rubinacci, *Solution of three dimensional eddy current problems by integral and differential methods*, IEEE Trans. Magn., 24 (1988), pp. 98–101.

[33] I. Bardi, O. Biro, K. Preis, G. Vrisk, and K. Richter, *Nodal and edge element analysis of inhomogeneously loaded* 3*D cavities*, IEEE Trans. Magn., 28 (1992), pp. 1142–1145.

[34] J. P. Webb and B. Forghani, *The low-frequency performance of* H-ϕ *and* T-Ω *methods using edge elements for* 3*D eddy current problems*, IEEE Trans. Magn., 29 (1993), pp. 2461–2463.

[35] J. H. Bramble, J. E. Pasciak, J. Wang, and J. Xu, *Convergence estimates for product iterative methods with applications to domain decompositions and multigrid*, Math. Comp., 57 (1991), pp. 1–21.

[36] R. Hiptmair, *Multigrid method for Maxwell's equations*, SIAM J. Numer. Anal., 36 (1998), pp. 204–225.

[37] I. Babuska, A. Craig, J. Mandel, and J. Pitkaranta, *Efficient preconditioning for the p-version finite element method in two dimensions*, SIAM J. Numer. Anal., 28 (1991), pp. 624–661.

[38] I. Babuska, B. Szabo, and I. N. Katz, *The p-version of the finite element method*, SIAM J. Numer. Anal., 18 (1981), pp. 515–545.

SIAM J. SCI. COMPUT.
Vol. 23, No. 4, pp. 1077–1093

ITERATIVE MULTIGRID REGULARIZATION TECHNIQUES FOR IMAGE MATCHING[*]

STEFAN HENN[†] AND KRISTIAN WITSCH[‡]

Abstract. In this paper, we consider the problem of matching images, i.e., to find a deformation u, which transforms a digital image into another such that the images have nearly equal gray values in every image element. The difference of the two images is measured by their L_2-difference, which should be minimized. This yields a nonlinear ill conditioned inverse problem for u, so the numerical solution is quite difficult. A Tikhonov regularization method is considered to rule out discontinuous and irregular solutions to the minimization problem. An important problem is a proper choice of the regularization parameter α. For the practical choice of α, we use iterative regularization methods based on multigrid techniques. To obtain a suitable initial guess, we use an approach similar to the full multigrid (FMG) developed by Brandt [*Math. Comp.,* 31 (1977), pp. 333–390]. The algorithms have optimal complexity: the amount of work is proportional to the number of picture elements. Finally, we present some experimental results for synthetic and real images.

Key words. image processing, multigrid methods, nonlinear ill-posed problems, Tikhonov regularization

AMS subject classifications. 65J20, 65N55, 68U10

PII. S106482750037161X

1. Introduction.

An important problem in two- and three-dimensional medical image analysis is to match two similar images resulting from different imaging modalities. In brain research, which was our starting point, it is necessary for the analysis of the organization and variation in the structure of human brains. A good survey of a part of the practical applications is given in [6]. In this paper, we consider a gray-level based mapping method of two similar images. The description of the problem and the numerical algorithms for its solution will be given for the two-dimensional problem. The extension to three dimensions is obvious and has been done.

In two dimensions, the matching leads to the following problem: find a displacement vector $u = (u_1, u_2)^t$ (whose components are functions of the variables $x = (x_1, x_2)^t$) that transforms one image T, called the template, into another image R, called the reference, so that

$$R(x) = T(x - u(x)). \tag{1.1}$$

In continuous two-dimensional variables, the reference R and the template T can be represented by functions $T, R : \Omega \subset \mathbb{R}^2 \longrightarrow \mathbb{R}^{\geq 0}$ which associate with the pixel (picture element) $(x_1, x_2) \in \mathbb{R}^2$ its intensity (gray level), $T(x_1, x_2)$, and $R(x_1, x_2)$, respectively. In our example, Ω will simply be the unit square $[0,1]^2 \subset \mathbb{R}^2$. A digital image B^h is a finite-dimensional approximation of a continuous image B with $B^h : \Omega_h \to [0, g_{max}] \subset \mathbb{N}^{\geq 0}$ and maximal gray-level g_{max}.

We define a map $F = T \circ \phi$, with $\phi(x) = x - u(x)$, from the space of displacements

[*] Received by the editors May 2, 2000; accepted for publication (in revised form) June 5, 2001; published electronically October 11, 2001.
http://www.siam.org/journals/sisc/23-4/37161.html

[†] Institut für Mathematische Optimierung, Heinrich-Heine Universität Düsseldorf, Universitätsstraße 1, D-40225 Düsseldorf, Germany (henn@am.uni-duesseldorf.de).

[‡] Institut für Angewandte Mathematik, Heinrich-Heine Universität Düsseldorf, Universitätsstraße 1, D-40225 Düsseldorf, Germany (witsch@am.uni-duesseldorf.de).

$\mathcal{X}$ into the set $\mathcal{Y}$ of images

$$F : \mathcal{D}(F) \subset \mathcal{X} \longrightarrow \mathcal{Y}, \qquad F : u \longmapsto T(x - u(x)). \tag{1.2}$$

We consider the subset

$$\mathcal{D}(F) = \left\{ u \in \mathcal{X} \;\middle|\; x - u(x) \in \Omega \quad \forall x \in \Omega \right\}$$

of $\mathcal{X}$. This means that T, and therefore F, are defined. (This restriction can be avoided by extending T to a larger domain.) Thus, the gray-value matching problem (1.1) can be identified with an inverse problem

find u in (1.2), so that (1.1) holds approximately.

In general, this problem is ill-posed in the sense of Hadamard. The information provided from R and the model (1.1) are not sufficient to ensure existence, uniqueness, and stability of a solution u.

In practical applications, only a noisy version R^δ of the exact data R is given with

$$||R - R^\delta|| \leq \delta.$$

In our application, δ is unknown, and we make no assumptions about statistics of the noise. In the case of perturbed data R^δ, the computation of a proper approximation u^δ of (1.1) requires regularization methods.

In section 2, we will present more precisely the minimization problem studied here, as well as the assumptions to impose on the model. We use a Tikhonov regularization for the so-called output least squares functional. It uses a regularization term which favors smooth transformations. This leads to a nonlinear variational problem similar to many other approaches in digital image processing or pattern recognition; cf. [1], [3], [7], [11], [12], [13], and [15].

The first attempts to deal with the issue of medical image matching can be traced back to Bajcsy and Kovacic [5]. These ideas were further developed in [8] using successive overrelaxation for a linearized model and in [1] using Fourier and wavelet representations of a nonlinear functional. In [11] a Landweber iteration scheme was developed, including the multigrid correction scheme (CS) and a trust region method for one-dimensional minimization.

The minima of the Tikhonov functional depend significantly on the choice of the regularization parameter α. Theoretically, decreasing α should give a better agreement of the images, but in practice a too small α leads to strong artifacts because of the influence of high-frequency structures in the image data. Increasing α removes these artifacts and allows only smoother transformations. For further increasing α, the matching of the images becomes worse.

Our aim is to develop methods for the efficient and robust solution of the matching problem and to find a good value of the regularization problem without a priori information.

These methods are described in section 4. In a first approach, a standard multigrid method and the choice of a suitable regularization parameter are combined. The second method follows the solution curve for decreasing α. One starts with a relative large α, which is helpful for the solution method. Then one computes for decreasing α minimal solutions of the functional consisting of the output least squares part and a regularizing part applied to the difference from the previous solution. This process is stopped at the point where the output least squares functional increases.

Another important idea is to use a nested iteration or full multigrid (FMG) approach as introduced by Brandt [4]. The solution process starts with the same problem but a coarser resolution. It is easier to solve the nonlinear problem there and to find global minima. This solution is the starting point for the next finer resolution and usually needs only relatively small corrections.

The Euler–Lagrange equations of the Tikhonov functional are a coupled system of nonlinear PDEs which depend on the regularization parameter. Its discretization and approximation is described in section 3. Finally, we present some experimental results for synthetic and real images in section 6.

2. Tikhonov regularization of the nonlinear inverse problem.

To find an approximate solution of (1.1), we measure the $L_2(\Omega)$ difference of the two images $h(u) = T(x - u) - R(x) = F(u) - R(x)$ by

$$D(u) = \langle h(u), h(u)\rangle_{L_2(\Omega)} = ||h(u)||^2_{L_2(\Omega)} = \int_\Omega (F(u(x)) - R(x))^2 d\Omega.$$

Here, $D(u)$ is the so-called output least squares functional with given data $R(x)$. The inverse problem will be solved by approximatively minimizing $D(u)$, which is a standard method for inverse problems. We can add further terms related, for example, to edges without essential changes in the algorithm presented here.

2.1. Tikhonov regularization.

To rule out discontinuous and irregular solutions to the minimization problem, it is necessary to introduce a smoothing or regularizing term $\alpha||u||^2_{\mathcal{X}}$ with a parameter $\alpha > 0$ for the deformation u. This means that one minimizes the Tikhonov functional

$$J_\alpha(u) = ||h(u)||^2_{L_2(\Omega)} + \alpha||u||^2_{\mathcal{X}} \quad \text{for} \quad u \in \mathcal{D}(F) \tag{2.1}$$

with an appropriate norm $||\cdot||_{\mathcal{X}}$ for the space $\mathcal{X}$.

The statistical model proposed by Amit, Grenander, and Piccioni [2] for image restoration leads us to choose $||u||^2_{\mathcal{X}} = a(u, u)$ with a bilinear form a and to search for a solution among the minima of the energy

$$J_\alpha(u) = ||h(u)||^2_{L_2(\Omega)} + \alpha a(u, u) \quad \text{over} \quad \mathcal{D}(F). \tag{2.2}$$

The parameter $\alpha \in \mathbb{R}^+$ allows us to balance the influence of both terms in the Tikhonov functional $J_\alpha(u)$. For the brain data set, we use a regularizing term of the form

$$a(u, v) = \int_\Omega \left[\sum_{i,j=1}^{2} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} \right) \right] d\Omega \tag{2.3}$$

defined on $\mathcal{X} \subset H^1(\Omega) \times H^1(\Omega)$. This model assumes that the difference between the images comes from an elastic deformation of the brain. The elastic constants are chosen so that the changes in volume are maximal, which corresponds to longitudinal stretch without lateral shrink. The bilinear form measures the energy of the elastic deformation, is isotropic in the directions, and is neutral with respect to translations and rotations but penalizes these transformations by the boundary conditions introduced in section 2.2.

This is an appropriate model for the case that the template image T is obtained by slicing and scanning the brain as described in [5]. Other bilinear forms on $H^1(\Omega) \times H^1(\Omega)$ as the Laplacian would give similar results but have no coupling between the directions as (2.3). Regularization in $H^2(\Omega) \times H^2(\Omega)$ as by the biharmonic operator enforces too much smoothness of the displacements.

Total variation based regularization as in [7], [13] is not appropriate in this case, but useful for other applications, and will be investigated in the future.

2.2. Minimization by a variational equation. With the described form of regularization in the functional J_α and $f(u) = -\frac{1}{\alpha} h'(u) \cdot h(u)$, the Euler–Lagrange equation for the minimizer u is given by the nonlinear variational equation

$$a(u, \phi) = \langle f(u), \phi \rangle_{L_2(\Omega)} \quad \forall \phi \in (H^1(\Omega))^2 = H^1(\Omega) \times H^1(\Omega). \tag{2.4}$$

For α very small, this is a singular perturbed problem which is difficult to solve.

Equation (2.4) is the weak form of a nonlinear coupled PDE. For $\mathcal{X} = (H^1(\Omega))^2$, one obtains Neuman boundary conditions. In our application, it is appropriate to use $\mathcal{X} = (H_0^1(\Omega))^2$ which yields Dirichlet boundary conditions

$$u(x) = 0 \quad \text{for} \quad x \in \partial\Omega.$$

(In principle, one has to enforce $u \in \mathcal{D}(F)$, but this was no problem in our application.) The minimization of the Tikhonov functional is therefore equivalent to finding a solution of the boundary value problem

$$E(u(x)) = \begin{cases} -\alpha(\Delta u(x) + \operatorname{grad}(\operatorname{div} u(x))) - f(u(x)) & = & 0 & \text{for } x \in \Omega, \\ u(x) & = & 0 & \text{for } x \in \partial\Omega. \end{cases} \tag{2.5}$$

3. Discretization, approximation, and solution methods of the nonlinear PDE. For the purpose of the numerical solution of (2.5), we approximate the infinite-dimensional space $\mathcal{X}$ by a finite-dimensional space $\mathcal{X}_n$.

3.1. Discretization. The image T^h obtained from the continuous image T can be considered as a function which is piecewise constant on each pixel of a regular square grid G_h. For the discretization of the PDE (2.5), we use G_h with the pixel-centered gridpoints as in Figure 1:

$$G_h = \{x \in \mathbb{R}^2 : x = (x_i, x_j) = (h/2 + ih, h/2 + jh),\ i, j = 0, 1, \ldots, n-1\}.$$

The displacement vector is a grid function $u_h(x_h) = (u_{1,h}, u_{2,h})^t \in (\mathcal{F}(G_h))^2 = \mathcal{F}(G_h) \times \mathcal{F}(G_h)$ on G_h. (In three-dimensional magnetic resonance (MR), there is a sampling difference between the $x-y$-plane and the z-direction. In practice, it is not necessary to take this into account for the adaptation process. It could be done, but then the multigrid (MG) components would have to be modified.)

3.2. Computation of the nonlinear functional $f(u(x))$. The nonlinear functional $f(u(x))$ in (2.5) has to be evaluated by discretization. With $\phi(u) = \phi(x, u) = x - u(x)$, one gets

$$f(u) = h'(u) \cdot h(u) = J_{T^h}(\phi(u)) \cdot J_\phi(u) \cdot h(u) = -\nabla T^h(\phi(u)) \cdot h(u),$$

since $J_\phi(u) = -I_2$, the 2×2 identity, and $J_{T^h}(x) = T^{h\prime}(x) = \nabla T^h(x)^t$. For determining a second order approximation of ∇T^h, the right-hand side $f_h(u_h) = (f_{1,h}, f_{2,h})^t \in$

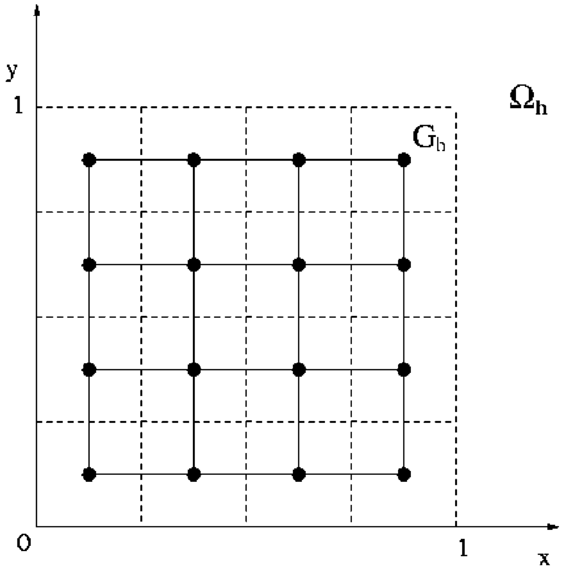


FIG. 1. *The pixel-centered grid G_h in two-dimensions for* 4×4 *pixels.*

$\mathcal{F}(G_h) \times \mathcal{F}(G_h)$ for an inner point (x_i, x_j) of G_h is computed with central differences by

$$f_h(u_h(x_i, x_j)) = \frac{1}{2} \left(\frac{T^h_{i+1,j} - T^h_{i-1,j}}{h}, \frac{T^h_{i,j+1} - T^h_{i,j-1}}{h} \right)^t (T^h_{i,j} - R^h_{i,j}), \tag{3.1}$$

where t denotes the transpose, $T^h_{i,j} = T^h(x_i - u_{1,h}(x_i, x_j), x_j - u_{2,h}(x_i, x_j))$ the deformed template image, and $R^h_{i,j} = R^h(x_i, x_j)$ the reference image.

3.3. Approximation of the PDE. The PDE (2.5) is discretized by the finite difference method using second order approximations. This is done by replacing the partial derivatives by corresponding difference quotients. This yields the following second order stencils:

$$-\Delta u = -u_{xx} - u_{yy} = \frac{1}{h^2} \begin{bmatrix} 0 & -1 & 0 \\ -1 & +4 & -1 \\ 0 & -1 & 0 \end{bmatrix} u_h + \mathcal{O}(h^2)$$

and

$$-u_{xy} = \frac{1}{4h^2} \begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix} u_h + \mathcal{O}(h^2).$$

Therefore, the discrete operator $L_h = -\Delta_h - \text{grad}_h(\text{div}_h)$ is represented by

$$L_h(u_h) = \begin{cases} \frac{1}{h^2} \begin{bmatrix} 0 & -1 & 0 \\ -2 & 6 & -2 \\ 0 & -1 & 0 \end{bmatrix} u_{1,h} + \frac{1}{4h^2} \begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix} u_{2,h}, \\ \\ \frac{1}{h^2} \begin{bmatrix} 0 & -2 & 0 \\ -1 & 6 & -1 \\ 0 & -2 & 0 \end{bmatrix} u_{2,h} + \frac{1}{4h^2} \begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix} u_{1,h} \end{cases} \tag{3.2}$$

for $u_h = (u_{1,h}, u_{2,h})^t \in (\mathcal{F}(G_h))^2$. By using (3.2) for all inner gridpoints $x_h \in G_h$ and $u_h = 0$ for all boundary gridpoints, together with the discretized form of the nonlinear functional $f(u)$ as described in (3.1), one obtains a nonlinear system for the unknown values of the translation vector

$$E_h(u_h(x)) = \alpha L_h(u_h(x)) - f_h(u_h(x)) = 0. \tag{3.3}$$

This is the discretized form of the boundary value problem (2.5). The nonlinear system in two dimensions for typical images with 256 × 256 pixels has 2^{17} equations and unknowns. The corresponding system in three dimensions with 256 × 256 × 128 voxel has 3×2^{23} equations and unknowns. This makes clear that it is extremely important to use very efficient solvers. Otherwise, in particular the three-dimensional problem cannot be solved.

3.4. Coarse grid problems and intergrid transfers. In the discretized PDE (3.3), the nonlinear functional $f_h(u_h)$ is naturally defined on the fine grid G_h by the pixel size of the digital images T^h and R^h (section 3.2). To use efficient solution methods like the multigrid full approximation scheme (FAS) [4], [14], for the nonlinear PDE we have to define the functional $f_h(u_h)$ also on coarse grids.

The basic idea of the FAS is to smooth the errors of the solution such that they can be approximated on a coarser grid. On the coarse grid, a defect equation is solved, and then the coarse grid corrections are interpolated back to the fine grid, where the errors are smoothed again. The components of the FAS for this problem are described in the following.

By coarsening the digital images, a corresponding nonlinear functional $f_H(u_H)$ can be defined on coarse grids $G_{2^l h}$ with $l > 0$. This means that we approximate the infinite-dimensional spaces $\mathcal{X}$ and $\mathcal{Y}$ by a sequence of finite-dimensional subspaces $\{\mathcal{Y}_l\}_{l=0,1,2,...}$ with $\mathcal{Y}_l = \mathcal{Y}_{2^l h}$ for the images and $\{\mathcal{X}_l\}_{l=0,1,2,...}$ with $\mathcal{X}_l = \mathcal{X}_{2^l h}$ for the displacements. The sequences have the properties

$$\mathcal{X}_L \subset \mathcal{X}_{L-1} \subset \cdots \subset \mathcal{X}_0 \subset \cdots \subset \mathcal{X} \quad \text{and} \quad \mathcal{Y}_L \subset \mathcal{Y}_{L-1} \subset \cdots \subset \mathcal{Y}_0 \subset \cdots \subset \mathcal{Y}.$$

The images are coarsened by collecting several picture elements into one coarse picture element. This can be described by the operator $R^l_{l-1} : \mathcal{Y}_{l-1} \to \mathcal{Y}_l$. This corresponds to a lowpass filter in digital imaging. This procedure yields a sequence of coarser and coarser images with containing only information about corresponding coarse structure. On all grids, the discretization of the PDE is done in the same way as on the finest grid as described in section 3.1; cf. Figure 2.

Standard components for the intergrid transfers (injection, half weighting, and full weighting for restriction and linear and bilinear interpolation) in the FAS were implemented.

4. Approximate solution methods.

We now introduce two different methods for obtaining approximate solutions to the nonlinear problem. We consider the finite-dimensional Tikhonov functional $J_{\alpha,h} : G_h \times G_h \to \mathbb{R}^{\geq 0}$. The functional is positive and continuous. Therefore the Tikhonov functional has at least one global minimum on $\mathcal{X}$ because $J_{\alpha,h} \to \infty$ for $||u||_{\mathcal{X}} \to \infty$.

In general, the minimum is not unique. Moreover, a solution of $J_{\alpha,h}$ depends crucially on the regularization parameter α. The aim of our methods is to find a suitable solution of the minimization problem, hopefully a global one, without any a priori information about the image data and about a suitable choice of α in (2.2).

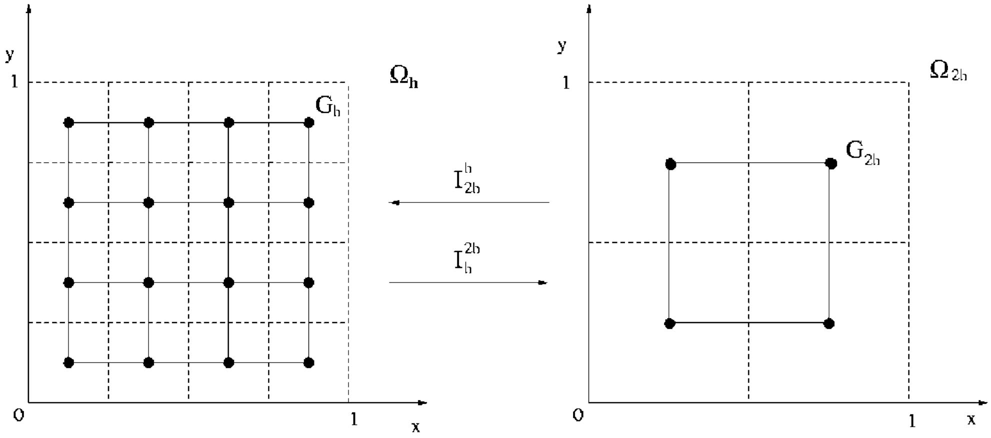


FIG. 2. *Intergrid transfers.*

The regularization parameter α determines the ratio between the influence of the image least squares differences $D(u)$ and the smoothness $a(u, u)$ of the displacements. For α too large, the influence of $D(u)$ is damped, and the matching is quite bad since the images behave as stiff materials. For α too small, the displacements are not smooth and are useless. Then, the numerical solution is difficult or even impossible. In particular, for $\alpha = 0$, each pixel could be mapped onto any other pixel with the same gray level.

In the first algorithm, we try to choose a deformation which is relatively independent of α. In the second algorithm, we solve a sequence of subproblems with decreasing parameter α. For each subproblem, we determine the parameter α within the iteration.

4.1. Algorithm 1: Modified FAS for minimizing output least squares functional. In this section, we will describe a method which incorporates the solution process and the choice of a suitable regularization in a modified FAS algorithm. We combine the multigrid idea and the minimization of the functional as follows.

The main components of every multigrid process are the relaxation and the coarse grid correction. As in standard multigrid methods, the coarse grid correction is determined by restricting the defect and the current approximation to the coarse grid. The coarse grid problem is solved approximately, the solution is interpolated to the fine grid and the solution is added to the fine grid approximation. However, in our case a proper scaling of the coarse grid correction and of the result of the relaxation yields significantly faster convergence, in particular for smaller α. Therefore, we proceed as follows.

On the fine grid an intermediate approximation $\overline{u}_h^{(k+1)}$ is computed by a nonlinear relaxation method for (2.5). We use a nonlinear Jacobi relaxation. With respect to the discretization and approximation of (2.5), we get

$$\frac{6}{h^2} u_{1,h}^{(n+1)} + f_{1,h}(u^{(n+1)}) = \frac{1}{h^2} \begin{bmatrix} & 1 & \\ 2 & & 2 \\ & 1 & \end{bmatrix} u_{1,h}^{(n)} + \frac{1}{4h^2} \begin{bmatrix} -1 & & 1 \\ & & \\ 1 & & -1 \end{bmatrix} u_{2,h}^{(n)}$$

$$\frac{6}{h^2} u_{2,h}^{(n+1)} + f_{2,h}(u^{(n+1)}) = \frac{1}{h^2} \begin{bmatrix} & 2 & \\ 1 & & 1 \\ & 2 & \end{bmatrix} u_{2,h}^{(n)} + \frac{1}{4h^2} \begin{bmatrix} -1 & & 1 \\ & & \\ 1 & & -1 \end{bmatrix} u_{1,h}^{(n)}$$

with initial guess $u_h^{(0)} \in \mathcal{D}(F)$. (On the coarse grids, the restricted defect of the next finer grid has to be added to the right-hand side.)

Due to the dependence on α, this approximation may result in a bad matching, as described above. To correct this, the approximation is modified by

$$u_h^{(k+1)} = u_h^{(k)} + \omega \cdot \left(\overline{u}_h^{(k+1)} - u_h^{(k)} \right).$$

The parameter ω is the solution of the one-dimensional minimization problem

$$\text{find } \omega \in \mathbb{R} \quad \text{so that} \quad \omega = \arg \min_{\overline{\omega} \in \mathbb{R}^{>0}} D \left(u_h^{(k)} + \overline{\omega} \cdot \left(\overline{u}_h^{(k+1)} - u_h^{(k)} \right) \right).$$

This approximation and its defect

$$d_h = rhs_h - E_h \left(\overline{u}_h^{(k+1)} \right)$$

are transferred to the coarse grid as in standard multigrid. On the fine grid, the right-hand side is given by $rhs_h = 0$ and as described in (4.2) on the coarse grid. On the coarse grid, we first choose a nonlinear relaxation scheme

$$\overline{u}_H = u_H + v_H = \mathcal{R}_H(u_H, v_H, d_H) \tag{4.1}$$

for the resulting nonlinear discrete system

$$E_H(u_H + v_H) = d_H + E_H(u_H) \tag{4.2}$$

with

$$u_H = \left(u_1^{(1,1)}, \ldots, u_1^{(n-1,n-1)}, u_2^{(1,1)}, \ldots, u_2^{(n-1,n-1)} \right) \in \mathcal{F}(G_H) \times \mathcal{F}(G_H)$$

and defect $d_H \in \mathcal{F}(G_H) \times \mathcal{F}(G_H)$ on the coarse grid to compute the coarse grid correction v_H in the full approximation scheme.

Again, we choose a parameter ω as the solution of the one-dimensional minimization-problem

$$\text{find } \omega \in \mathbb{R} \quad \text{so that} \quad \omega = \arg \min_{\overline{\omega} \in \mathbb{R}^{>0}} D(u_H + \overline{\omega} \cdot v_H). \tag{4.3}$$

Then the coarse grid correction is

$$\overline{v}_H = \omega \cdot v_H$$

and

$$u_h^{new} = u_h + I(\overline{v}_H).$$

In standard multigrid, a similar scaling is known, but $J_{\alpha,h}$ would be minimized. Here, the regularization term is used in determining the direction v_H but omitted in choosing ω. This reduces the influence of α on the final solution.

This two-level algorithm can be converted into a multigrid algorithm in the same way as in a standard FAS by recursive solution of the coarse grid problem by the two grid method.

4.2. Algorithm 2: Iterative Tikhonov regularization. In the previous section, we have described an algorithm which computes an approximation for a given parameter α but which reduces the influence of α on the solution by a one-dimensional minimization problem within a FAS.

In this section, we present an algorithm which is more complicated than algorithm 1 and is in some respect similar to the solution with the L-curve criterion investigated by Hansen [9] and Hansen and O'Leary [10]. The L-curve criterion is a practical method for choosing the regularization parameter α for linear inverse problems. It was observed that the log-log-graphic presentation of the curve $L = \{x(\alpha), y(\alpha)\}$ with the norm $x(\alpha)$ of the regularized solution $u_\alpha(x)$ of (2.2) and the corresponding residual norm $y(\alpha)$ can be useful in studying the problem.

The so-called L-curve method derives its name from the plot of the graph $\{x(\alpha), y(\alpha)\}$, which resembles the shape of the letter "L." Here, the basic idea is that a point on the graph near the "corner" of the "L" represents a reasonable compromise between data fit and smallness of the regularizing term. In our situation, the use of the L-curve criterion is dissatisfying due to the following reasons:

- The description of the L-curve criterion is for linear problems. (This means that F in (1.2) is a linear operator.) The idea can be generalized to nonlinear problems, but in general we saw no useful "L."
- For the iterative solution and the singular perturbation problems, we cannot compute solutions for small α.
- An important role for minimizing an nonlinear functional is the choice of an initial guess. This is not taken into account within the L-curve criterion. In contrast to linear problems, there is no optimal regularization parameter for an optimal solution.
- The L-curve criterion seems to prefer too smooth solutions in our case with nonoptimal matching of the images; see Figure 3.

In the following, we introduce a method which may be viewed as an iterated Tikhonov regularization. The method consists of a sequence of minimization subproblems

$$\left\{ \min_{u\in\mathcal{D}(F)} \{||h(u)||^2_{L_2(\Omega)} + \alpha_k||u - u^{(k)}||^2_{\mathcal{X}}\} \right\}_{k\in\mathbb{N}} \tag{4.4}$$

whose solutions $u = u_\alpha^{(k+1)}$ depend on the parameter $\alpha_k > 0$ and on the initial guess $u^{(k)}$. Each subproblem is well posed for α sufficiently large and can be solved efficiently by a FAS.

There are two ideas: The first is, not to use a very small α in order to simplify the solution process. The second is the use of $u - u^{(k)}$ in the regularization term diminishes the influence of the regularization since $u^{(k)} \approx u$ but allows only reasonable solutions. An important issue herein is the determination of the first iterate.

For the numerical implementation of the iterative Tikhonov regularization, we compute the solution $u_\alpha^{(1)}$ with relative large α and an initial guess $u^{(0)} \in \mathcal{D}(F)$; cf. section 5. This is done by solving the nonlinear Euler–Lagrange equation (2.5) with a FAS and initial guess $u^{(0)}$. For the next iteration, we reduce α by a factor $0 < \kappa < 1$ (e.g., $\kappa = 1/2$) and take the solution $u_\alpha^{(1)}$ as the initial guess.

We get a sequence of displacement fields $\{u^{(k)}\}_{k\in\mathbb{N}}$ which successively reduces the value of the nonlinear functional $D(u)$. We stop the iteration if $D(u^{(k+1)}) > D(u^{(k)})$ and get the following algorithm.

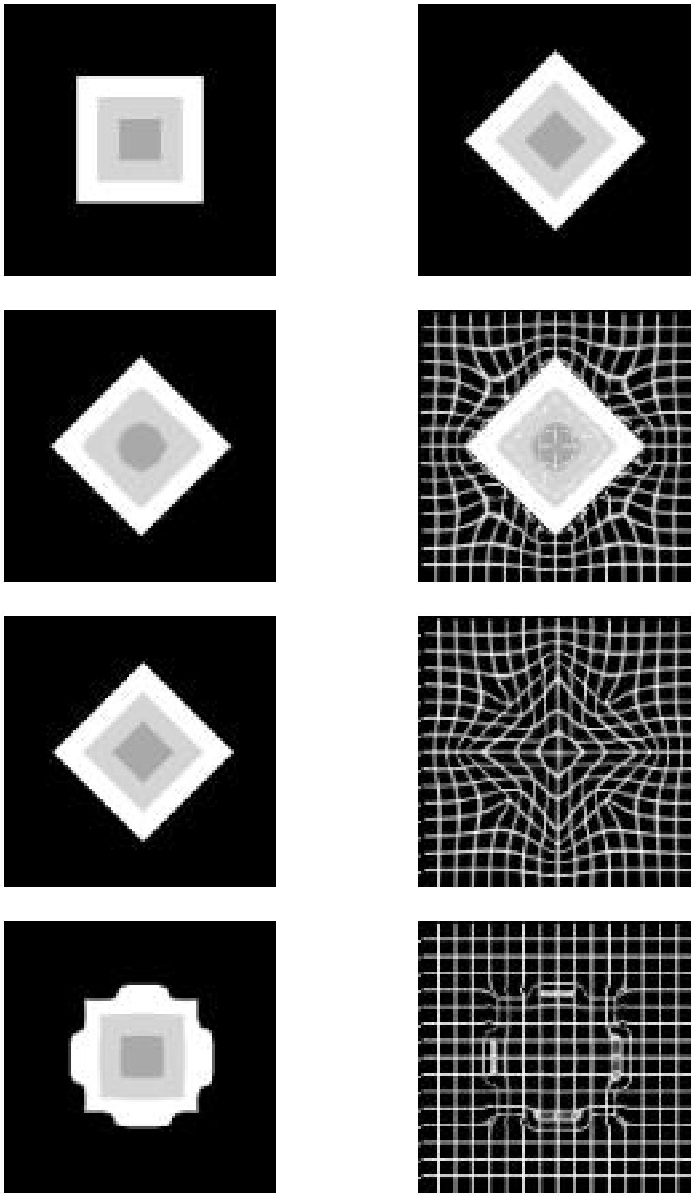


FIG. 3. *From left to right:* 1. *Template image* T. 2. *Reference image* R. 3. *Template deformed by the solution of algorithm* 1 *introduced in section* 4.1. 4. *Deformed template and superimposed deformed uniform grid for showing the deformation applied to the template.* 5. *Template deformed by the solution of algorithm* 2 *introduced in section* 4.2. 6. *Uniform grid deformed by the solution.* 7. *Template deformed by the solution of the L-curve criterion.* 8. *Uniform grid deformed by the solution of the L-curve criterion.*

Algorithm 2: Iterative Tikhonov regularization for minimizing $D(u)$.

```
$k = 0$; $u^{(k)} = u^*$; $\alpha_k = N \gg 0$;
repeat
  compute $u^{(k+1)} = \arg\min_{u \in \mathcal{D}(F)} ||h(u)||_2^2 + \alpha_k ||u^{(k)} - u^*||_{\mathcal{X}}^2$
  reduce $\alpha = \kappa \cdot \alpha$;
until $\big(D(u^{(k+1)}) > D(u^{(k)})\big)$.
```

5. Multiresolution minimization and multilevel image matching. Within the minimization of the Tikhonov functional (2.1) in sections 4.1 and 4.2, the subsets $\mathcal{X}_h \subset \mathcal{X}$ and $\mathcal{Y}_h \subset \mathcal{Y}$ are given by the number of pixels. This pixel size h is also the step size of the discretization of the PDE in section 3.

The use of a small discretization parameter h causes two difficulties:

- The adaption process mainly adapts structures with size h and neglects coarser structures. Therefore the minimization process finds local minima far from the global one.
- The minimization process requires many iterations with a fine resolution h.

Both difficulties can be avoided to a large extent by considering the Tikhonov functional also on coarser resolutions. The basic principle of this approach is that the image structures will be matched on that resolution on which they can be represented. This means that we approximate the infinite-dimensional spaces $\mathcal{X}$ and $\mathcal{Y}$ by a sequence of finite-dimensional subspaces, as in section 3.4. We get a sequence of minimization problems

$$(M_l) \qquad \min_{u_l \in \mathcal{X}_l} \left\{ ||h(u)||_2^2 + \alpha a(u, u) \right\}$$

defined on the spaces $\mathcal{X}_l$. With an interpolation operator $I_l^{l-1} : \mathcal{X}_{2^l h} \to \mathcal{X}_{2^{(l-1)}h}$, we transform the solution $u_l \in \mathcal{X}_l$ onto the next finer resolution. Then $I_l^{l-1}(u_l) \in \mathcal{X}_{l-1}$ is a suitable initial guess for the minimization of M_{l-1}.

Multilevel image matching.

```
$u_L = 0; l = L$;
repeat
  if $(l = L)$
    compute $u_L^*$ with initial approximation $u_L$;
    $l = l - 1$;
  else
    $u_{l-1} = I_l^{l-1}(u_l^*)$;
    compute $u_{l-1}^*$ with initial approximation $u_{l-1}$;
  endif
  $l = l - 1$;
until l = 1.
```

6. Results. In order to conclude this paper, we present experimental results for synthetic as well as MR images. To demonstrate the principle and reliability of the iterative Tikhonov iteration scheme and the modified FAS described in sections 4.1 and 4.2, we consider the synthetic images shown in Figure 3. These images are quite different but can be adapted quite well, depending on the required smoothness.

Comparing the synthetic template image (leftmost in Figure 3) with the synthetic reference image (second left in Figure 3) shows a difference of $\frac{1}{N}||T(x) - R(x)||_{L_2} \approx 71$ with $N = 128^2$ pixels. After the iteration described in sections 4.1 and 4.2 on a

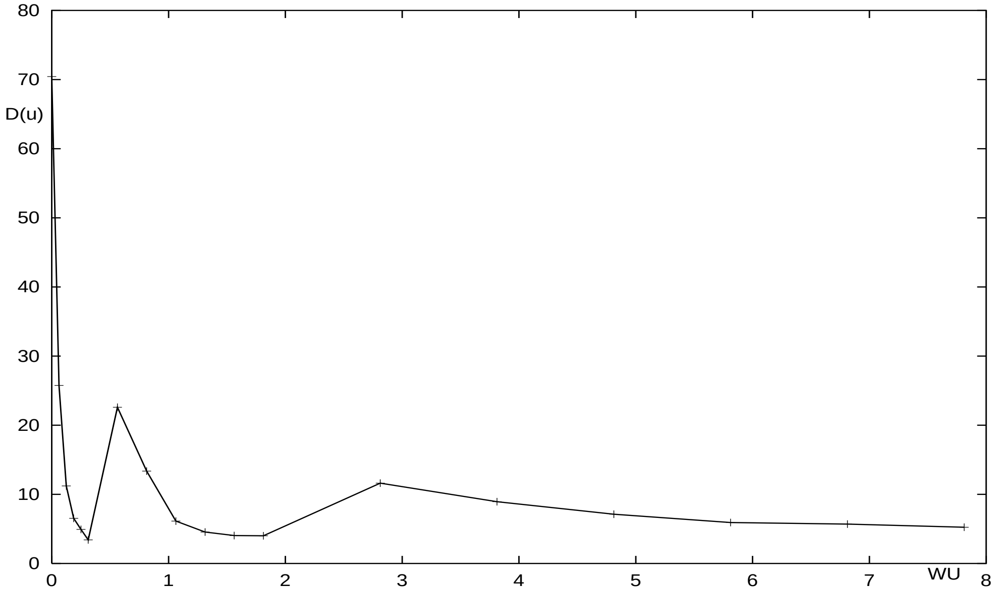


FIG. 4. *Least squares difference $D(u^{(k)})$ after each FAS-iteration (algorithm* 1 *in section* 4.1*) step for the example in Figure* 3.

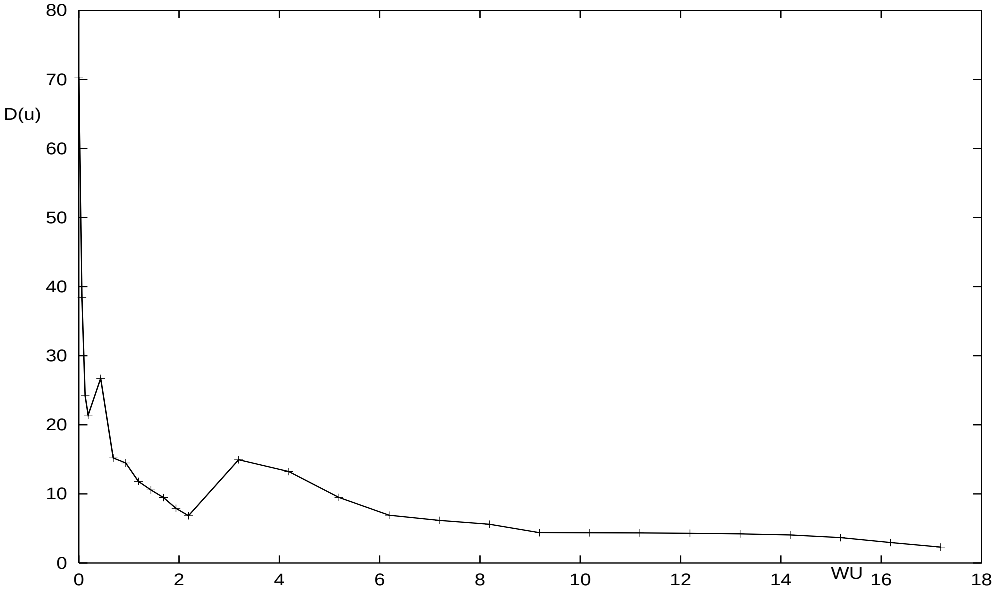


FIG. 5. *Least squares difference $D(u^{(k)})$ after each iteration step of the iterative Tikhonov regularization algorithm (algorithm* 2 *in section* 4.2*) for the example in Figure* 3.

sequence of the three subspaces $X_{1/32} \subset X_{1/64} \subset X_{1/128}$, we get a sequence of deformation fields $u^{(k)}$ on every subspace which reduces the difference $\frac{1}{N}||T(x - u^{(k)}(x)) - R(x)||_{L_2}$ within the iteration. Figures 4 and 5 show for the algorithms the decreasing least squares difference on the three different resolutions applied to the example in

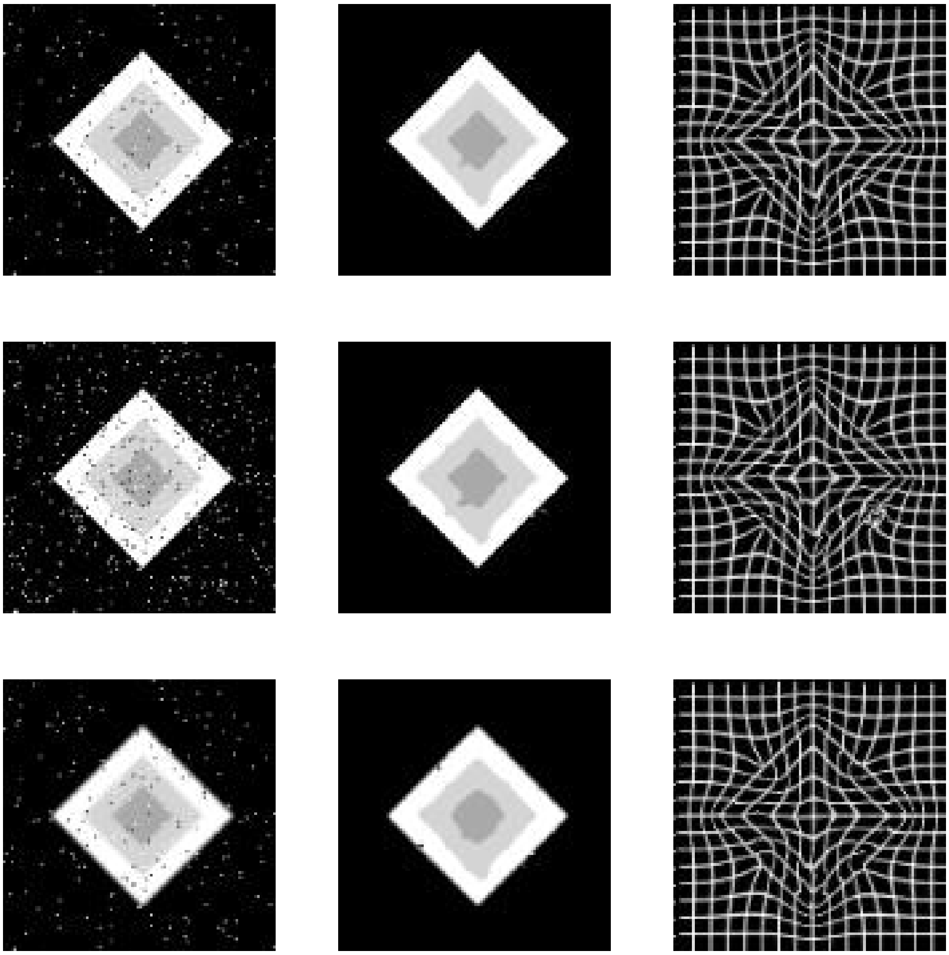


Fig. 6. *Images of the three experiments with different noise levels. Left: The reference with added noise R^δ (added noise from top to bottom:* 1. 2% *salt and pepper noise* 2. 5% *salt and pepper noise* 3. *blurred with* 2% *salt and pepper noise). Middle: Template image deformed by the solution of algorithm* 2. *Right: Uniform grid deformed by the solution of algorithm* 2.

Figure 3. The graphs are scaled by work units (WUs) on the finest grid, corresponding to the effort of a multigrid cycle.

In the examples, a WU of the first algorithm corresponds to approximately a $\frac{1}{3}$ WU of the second one. In all examples, we used a FAS with one relaxation before and one after the coarse grid correction for smoothing the error on three levels: full weighting restriction, bilinear interpolation, and the injection operator for the restriction of the image data and the approximate solutions.

In the next experiment, we analyze the dependence of the solution on the noise level $||R^\delta - R|| \leq \delta$ of the reference image R. For this, we added different types of noise to the synthetic reference images which appear in Figure 3. These images are shown in Figure 6. We observe that the results (in the middle of Figure 6) obtained by algorithm 2 are similar to the reference shown in Figure 3, except for artifacts caused by the added noise. The graphs of the decreasing defect functional $D(u)$ for these experiments scaled by WUs on the finest grid are shown in Figure 7. As expected, the

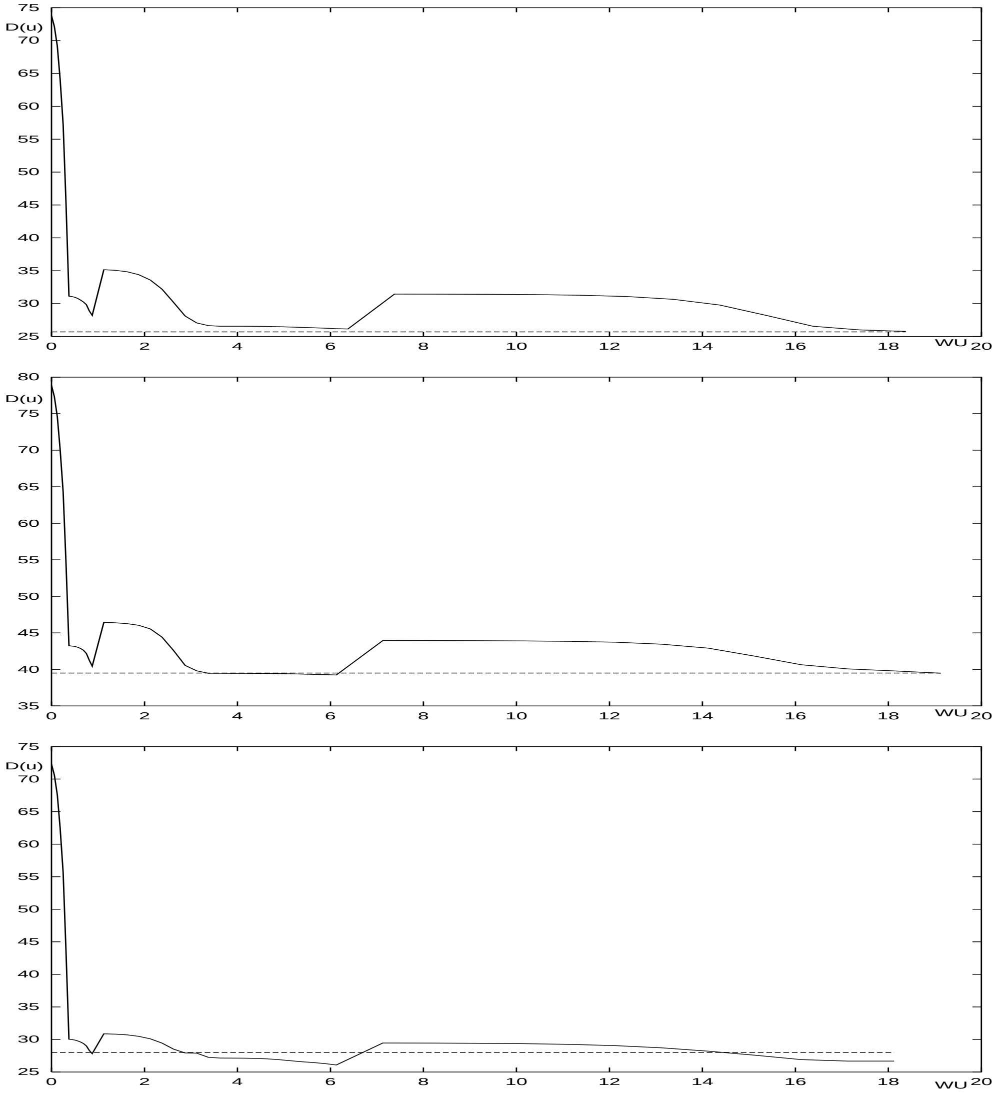


FIG. 7. *Least squares difference* $D(u^{(k)})$ *after each iteration step of algorithm* 2 *for the three experiments shown in Figure* 6 *and the noise level* δ *parallel to the abscissa.*

decreasing least squares functional on the three different resolutions can be reduced down to the noise level δ.

For estimation of the registration quality on MR slices, the iterative minimization algorithms for image matching are applied to a deformed MR template image $T(x)$ and a MR reference image $R(x)$, shown in Figures 8 and 9. Figure 10 shows the calculated result $T(x - u^*(x))$ deformed by the solution u^* of the minimization problem. The minimization problem is also solved on three subsets $X_{1/64} \subset X_{1/128} \subset X_{1/256}$. One can easily see in Figure 10 how the deformation field produces an image similar to the reference image. This can be stressed by the graphs in Figures 11 and 12. They show the decreasing least squares difference $D(u)$ between the images after each iteration step of the algorithms on different resolutions. The graphs are scaled by WUs on the finest grid.

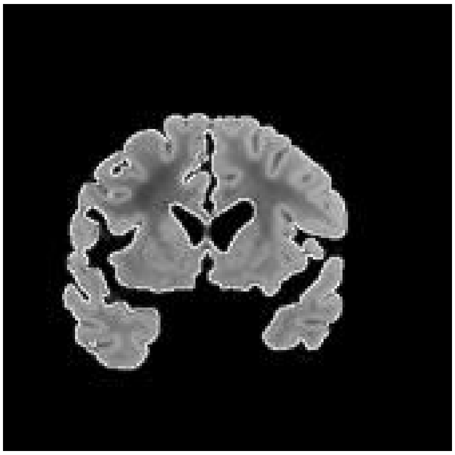


FIG. 8. *MR reference image with superimposed contour.*

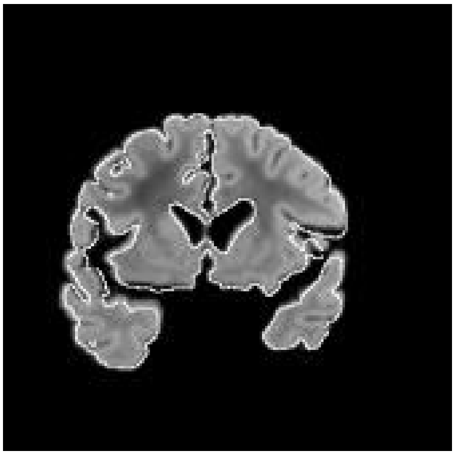


FIG. 9. *MR template slice with superimposed contour of the reference slice in Figure* 8.

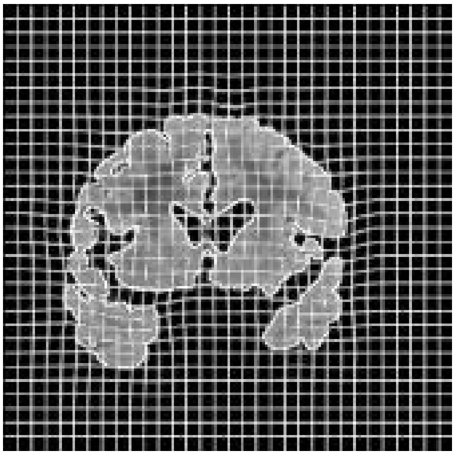


FIG. 10. *Deformed template with superimposed contour of the reference slice in Figure* 8 *and superimposed deformed uniform grid for showing the deformation applied to the template.*

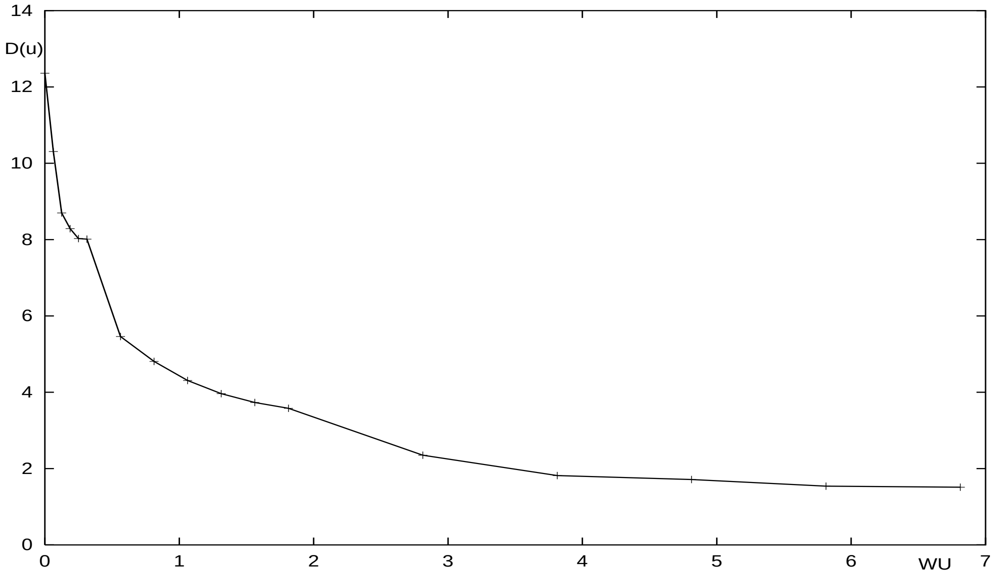


FIG. 11. *Least squares difference $D(u^{(k)})$ after each iteration step of algorithm* 1 *for the example in Figures* 8–10.

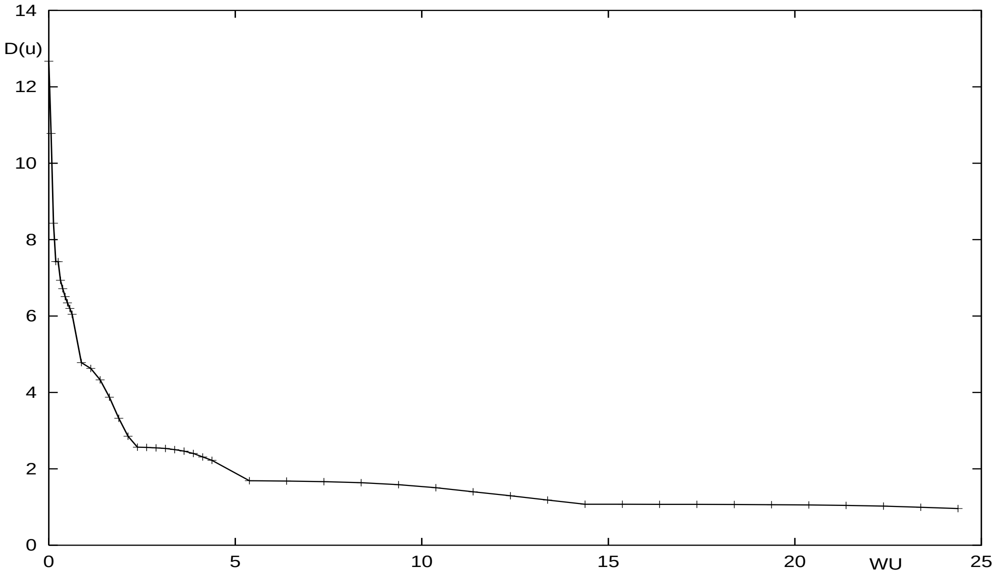


FIG. 12. *Least squares difference $D(u^{(k)})$ after each iteration step of algorithm* 2 *for the example in Figures* 8–10.

7. Conclusion. In this paper, we have introduced a nonlinear model for digital image matching. This inverse problem is ill-posed and its treatment requires careful use of regularizing techniques. For Tikhonov regularization, which aims at smooth solutions, a bilinear form is added. In section 4 we have presented two efficient methods for solving the problem.

The main advantages of these methods, compared to other known methods, are the self-controlled choices of the regularization parameter. In other papers [5], [1], this problem is not discussed, and therefore in practical applications "trial an error" methods for the parameter choice are inevitable. On the other hand, the use of a FAS for solving the nonlinear Euler–Lagrange equations is an essential benefit of the presented methods. This results in computing time which is linear in the number of the picture elements. This optimal scaling is important for the treatment of large, in particular three-dimensional, problems which are in practice unsolvable only with relaxation methods. In order to compare our methods with other methods in literature, Bajcsy and Kovacic [5] use the Jacobi method to determine deformations of a template and Christensen et al. [8] use successive overrelaxation (SOR) with checkerboard update to compute viscous fluid deformations of a template. In practice, due to the h-dependence convergence rates of the Jacobi method and the SOR, these methods are rather slow. For instance, the method in [8] has an execution time of 7 days with a MIPS R440 processor and 10 hours using a massively parallel DECmpp 128×64 MasPar computer with a resolution of $128 \times 128 \times 100$ voxel. With this resolution, our algorithm takes (with a nonoptimized implementation in C++) approximately one hour on an Ultra-Sparc Workstation.

Acknowledgment. We would like to thank the referees for useful remarks on the first version of the manuscript.

REFERENCES

[1] Y. Amit, *A nonlinear variational problem for image matching*, SIAM J. Sci. Comput., 15 (1994), pp. 207–224.

[2] Y. Amit, U. Grenander, and M. Piccioni, *Structural image restoration thought deformable templates*, J. Amer. Statist. Assoc., 86 (1991), pp. 376–387.

[3] G. Aubert and L. Vese, *A variational method in image recovery*, SIAM J. Numer. Anal., 34 (1997), pp. 1948–1979.

[4] A. Brandt, *Multi-level adaptive solutions to boundary value problems*, Math. Comp., 31 (1977), pp. 333–390.

[5] R. Bajcsy and S. Kovacic, *Multiresolution elastic matching*, Comput. Vision Graphics Image Process., 46 (1989), pp. 1–21.

[6] A. Barry, *Seeking Signs of Intelligence in the Theory of Control*, SIAM News, vol. 30/no. 3, 1997.

[7] T.F. Chan, G.H. Golub, and P. Mulet, *A nonlinear primal-dual method for total variation-based image restoration*, SIAM J. Sci. Comput., 20 (1999), pp. 1964–1977.

[8] G.E. Christensen, M.I. Miller, M. Vannier, and U. Grenander, *Individualizing neuroanatomical atlases using a massively parallel computer*, Computer, 29 (1996), pp. 32–38.

[9] P.C. Hansen, *Analysis of discrete ill-posed problems by means of the L-curve*, SIAM Rev., 34 (1992), pp. 561–580.

[10] P.C. Hansen and D.P. O'Leary, *The use of the L-curve in the regularization of discrete ill-posed problems*, SIAM J. Sci. Comput., 14 (1993), pp. 1487–1503.

[11] S. Henn and K. Witsch, *A multigrid-approach for minimizing a nonlinear functional for digital image matching*, Computing, 64 (2000), pp. 339–348.

[12] M. Piccioni, S. Scarlatti, and A. Trouvé, *A variational problem arising from speech recognition*, SIAM J. Appl. Math., 58 (1998), pp. 753–771.

[13] L.I. Rudin, S. Osher, and E. Fatemi, *Nonlinear total variation-based noise removal algorithms*, Phys. D, 60 (1992), pp. 259–268.

[14] K. Stüben and U. Trottenberg, *Multigrid Methods: Fundamental Algorithms, Model Problem Analysis and Applications*, Lecture Notes in Math. 960, Springer-Verlag, Berlin, 1982.

[15] D. Terzopoulos, *Image analysis using multigrid relaxation methods*, IEEE TPAMI, 2 (1986), pp. 129–139.

SIAM J. SCI. COMPUT.
Vol. 23, No. 4, pp. 1094–1112

A LAGRANGE SCHEME FOR A MATHEMATICAL MODEL OF POWDER COMPRESSION*

G. GODINAUD[†], M. N. LE ROUX[†], AND A. Y. LE ROUX[†]

Abstract. A mathematical model to simulate the compression of pharmaceutical powder under fixed deformation is proposed. This model is an hydrodynamic one, completed with the description of the porosity in the medium. This study leads to the elaboration of a general monodimensional model, made of three nonlinear partial derivative equations and a state law for the pressure. The mathematical resolution of this system is made using a splitting technique and an efficient numerical method.

Key words. nonconservative hyperbolic systems, conservative hyperbolic systems, splitting techniques, Lagrange method, pressure terms, diffusion terms

AMS subject classifications. 76M20, 35K55, 35L65, 65M06

PII. S1064827597233743

1. Introduction.

Studies on powder compression lead to descriptions of the compaction processes and also to predictions of the compactibility of powders. The consolidation mechanisms of a product subject to compression are many:

- elastic deformation,
- fragmentation or plastic deformation,
- mechanical binding,
- thermal effect.

In practice, the products undergo time-dependent deformation imposed by the punch of a press. At first contact with a powder bed, punch speed varies between 100 and 120 mm/s and decreases to 0 at the point of maximal penetration.

In this work, we propose a mathematical model to simulate the change from pulverulent state to compact state. With this aim in view, we have studied the dynamic behavior of powder undergoing deformation.

The experimental device is modelized in one dimension. The die containing the powder is described in space by [0,1], and the time by $[0, T]$ (T between 20 and 100 ms). We have decided to modelize the transitional part of the phenomenon and to follow the vanishing of the vacuum in the powder (evolution of the porosity) during the compression. This model is based on experimental results in pharmaceutical science, together with the specialists (see [7], [8], [9], [10]) for physical background.

To implement this model, we have considered that the evolution of the porosity was bound with the deformation speed tensor, and we have coupled this equation with the mass and momemtum conservations. Thus, the state law for the pressure is a function of the relative porosity and the density. Using the quantities

- σ : porosity parameter in [0,1],
- m : mass of powder,
- ρ : real density (without air),
- q : apparent density (with air), $q = (1 - \sigma)\rho$,
- v : volume,

[*] Received by the editors December 1, 1997; accepted for publication (in revised form) March 22, 1999; published electronically October 11, 2001.

http://www.siam.org/journals/sisc/23-4/23374.html

[†] GRAMM, UFR Math-Info, Université Bordeaux I, Talence, France (godinaud@math.u-bordeaux.fr, lmn@math.u-bordeaux.fr, AY.leroux@u-bordeaux.fr).

- u : particle speed,
- p : pressure,

we get the following nonconservative system (S):

$$\left\{\begin{array}{l} q_t + (qu)_x = 0, \\ \sigma_t + (\sigma u)_x = k(\sigma)\sigma u_x, \\ (qu)_t + (qu^2 + p)_x = \dfrac{\partial}{\partial x}(\varepsilon(q,\sigma)u_x), \\ + \ \text{state law} \quad : \quad p = a(\sigma)b(q), \end{array}\right. \tag{1}$$

where the function $k(\sigma)$ describes the features of the products and has the properties [1]

$$\left\{\begin{array}{l} 0 \leq k(\sigma) \leq 1, \\ k(1) = 1, \end{array}\right.$$

and the diffusion term $\frac{\partial}{\partial x}(\varepsilon(q,\sigma)u_x)$ allows a display of the phenomenon.

We are going to present the method of resolution for the system. It is a splitting technique which allows us to treat separately the different physical terms. We will group the terms in order to reproduce physical phenomena for which one can adapt a specific numerical method. In, particular, we will perform a Lagrangian treatment of the convection terms, well adapted to the following of the powder during the compression. This treatment must be conservative. The system (1) is nonconservative, and we have found a solution to obtain a conservative system equivalent to (1).

This splitting technique leads to degenerated problems, whose solutions are obtained by perturbation methods involving well-posed problems [1], [2], [3].

2. The conservative form. We consider system (1):

$$\left\{\begin{array}{l} q_t + (qu)_x = 0, \\ \sigma_t + (\sigma u)_x = k(\sigma)\sigma u_x, \\ (qu)_t + (qu^2 + p)_x = \dfrac{\partial}{\partial x}(\varepsilon(q,\sigma)u_x), \\ + \ \text{state law} \quad : \quad p = a(\sigma)b(q). \end{array}\right.$$

We denote the vector

$$W = \begin{pmatrix} q \\ \sigma \\ u \end{pmatrix},$$

and system (1) is equivalent to the system

$$W_t \ + \ A(W)\, W_x \ = \ V, \tag{2}$$

where the matrix A and the right-hand-side member V are given by

$$A(W) = \begin{pmatrix} u & 0 & q \\ 0 & u & \sigma(1 - k(\sigma)) \\ \dfrac{a(\sigma)b'(q)}{q} & \dfrac{a'(\sigma)b(q)}{q} & u \end{pmatrix},$$

$$V \ = \ \text{diffusion term}.$$

Under the condition

$$a(\sigma)b'(q) \; + \; \sigma(1-k(\sigma))\frac{a'(\sigma)b(q)}{q} \; > \; 0 \tag{3}$$

the matrix $A(W)$ has three real eigenvalues:

$$\begin{cases} \lambda_1 \; = \; u \; + \; \sqrt{a(\sigma)b'(q) \; + \; \sigma(1-k(\sigma))\dfrac{a'(\sigma)b(q)}{q}}, \\ \lambda_2 \; = \; u, \\ \lambda_3 \; = \; u \; - \; \sqrt{a(\sigma)b'(q) \; + \; \sigma(1-k(\sigma))\dfrac{a'(\sigma)b(q)}{q}}. \end{cases}$$

W is the solution of

$$\begin{cases} W_t \; + \; A(W)W_x \; = \; V, \\ W(0,x) \; = \; \begin{cases} W^l \text{ if } x < 0, \\ W^r \text{ if } x > 0. \end{cases} \end{cases} \tag{4}$$

We denote by (H_q, H_σ, H_u) the microscopic profile of shock associated with a solution of (1):

$$q \; = \; q_l \; + \; \Delta q H_q(x-ct), \tag{5}$$

$$\sigma \; = \; \sigma_l \; + \; \Delta\sigma H_\sigma(x-ct), \tag{6}$$

$$u \; = \; u_l \; + \; \Delta u H_u(x-ct), \tag{7}$$

where c is constant; H_q, H_σ, and H_u are Heaviside generalized functions [5], [6]; and $\Delta q = q_r - q_l$, $\Delta\sigma = \sigma_r - \sigma_l$, and $\Delta u = u_r - u_l$.

Replacing q and u with (5) and (7) in the first equation of (1), we get

$$(c - u_l - \Delta u H_u)\Delta q H_q' \; = \; (q_l + \Delta q H_q)\Delta u H_u', \tag{8}$$

so, by integration in $G_s(\mathbb{R})$,

$$q_l \; + \; \Delta q H_q \; = \; \frac{A}{(c - u_l - \Delta u H_u)}. \tag{9}$$

We get c, the sound speed, knowing that

$$\begin{cases} H_u = H_q = 0 \;\; \text{if} \;\; x < 0, \\ H_u = H_q = 1 \;\; \text{if} \;\; x > 0, \end{cases}$$

and so

$$c \; = \; \frac{\Delta(qu)}{\Delta q}. \tag{10}$$

Now we replace σ and u with (6) and (7) in the second equation of (1):

$$(c - u_l - \Delta u H_u)\Delta q H_\sigma' \; = \; (\sigma_l + \Delta\sigma H_\sigma)(1 - k(\sigma_l + \Delta\sigma H_\sigma))\Delta u H_u'. \tag{11}$$

Using the same method, we take the function ϕ defined by

$$\phi(\sigma) \; = \; \int \frac{d\sigma}{\sigma(1-k(\sigma))},$$

and we get

$$\phi(\sigma_l + \Delta\sigma H_\sigma) \;=\; -\ln(c - u_l - \Delta u H_u) + A, \tag{12}$$

where A is constant. The speed c is given by

$$c \;=\; \overline{u} \;+\; \frac{\Delta u}{2}\coth\frac{\Delta\phi}{2}, \tag{13}$$

where $\overline{u} \;=\; \frac{1}{2}(u_l+u_r)$. Thus, the Rankine–Hugoniot relations for the nonconservative system (1) are given by

$$c \;=\; \overline{u} \;+\; \Delta u\frac{\overline{q}}{\Delta q}, \tag{14}$$

$$c \;=\; \overline{u} \;+\; \Delta u\frac{1}{2}\coth\frac{\Delta\phi}{2}. \tag{15}$$

To get a conservative form for system (1), we must find with the jump conditions (13), (14) a variable w satisfying

$$\frac{\overline{w}}{\Delta w} \;=\; \frac{1}{2th\frac{\Delta\phi}{2}}. \tag{16}$$

Let us consider $w \;=\; \exp\phi(\sigma)$. We obtain

$$\begin{cases} \dfrac{\partial w}{\partial t} \;=\; w\;\dfrac{1}{\sigma(1-k(\sigma))}\sigma_t, \\ \dfrac{\partial w}{\partial x} \;=\; w\;\dfrac{1}{\sigma(1-k(\sigma))}\sigma_x. \end{cases}$$

This gives

$$\frac{\partial w}{\partial t} \;+\; \frac{\partial(wu)}{\partial x} \;=\; \frac{w}{\sigma(1-k(\sigma))}(\sigma_t + u\sigma_x + \sigma(1-k(\sigma))u_x), \tag{17}$$

but $\sigma_t + u\sigma_x + \sigma(1-k(\sigma))u_x \;=\; 0$, so we get the next theorem.

THEOREM 1. *Using the variables* $q, w \;=\; \exp\phi(\sigma)$ *and* u, *system* (1) *is equivalent to the following conservative system:*

$$\begin{cases} q_t + (qu)_x = 0, \\ w_t + (wu)_x = 0, \\ (qu)_t + (qu^2+p)_x = \frac{\partial}{\partial x}(\varepsilon(q,\sigma)u_x), \\ +\;\; state\;\; law\;\; :\;\; p = g(w)b(q). \end{cases} \tag{18}$$

System (18) is equivalent to the system

$$W_t \;+\; B(W)\,W_x \;=\; V, \tag{19}$$

where

$$B(W) = \begin{pmatrix} u & 0 & q \\ 0 & u & w \\ \dfrac{g(w)b'(q)}{q} & \dfrac{g'(w)b(q)}{q} & u \end{pmatrix}.$$

The matrix $B(W)$ has three real distinct eigenvalues, under the condition

$$g(w)b'(q) \;+\; w\,\frac{g'(w)b(q)}{q} \;>\; 0. \tag{20}$$

This condition is always true, because $g(w), b(q), g'(w), b'(q)$, and w are always positive.

3. Resolution. Let us consider the conservative system (18). We try to build a strong numerical method to resolve it. This leads us to use a splitting method, dividing (18) into three subsystems which will be solved separately. The usual numerical technique consists of taking an approximation of the initial problem [4]. Here, system (18) will be solved in three steps:

- convection terms,
- pressure terms,
- diffusion terms,

and the global scheme is made of the combination of the schemes adapted to each step.

3.1. The convection terms. We consider the system of convection terms (C):

$$\begin{cases} q_t + (qu)_x = 0, \\ w_t + (wu)_x = 0, \\ (qu)_t + (qu^2)_x = 0, \\ + \;\text{ state law } \;:\; p = g(w)b(q). \end{cases}$$

This system is degenerated: the speed u is a triple eigenvalue and Riemann solvers are very difficult to realize. The solution of this problem is obtained through perturbation method involving a well-posed problem. Let us consider the perturbed Riemann problem (C$_\varepsilon$):

$$\begin{cases} q_t + (qu)_x = 0, \\ w_t + (wu)_x = 0, \\ (qu)_t + (qu^2)_x + \varepsilon^2 p_x = 0 \end{cases} \tag{21}$$

with the piecewise constant initial data

$$(q, w, u) \;=\; \begin{cases} q_l, w_l, u_l & \text{for } \; x < 0, \\ q_r, w_r, u_r & \text{for } \; x > 0. \end{cases} \tag{22}$$

For a fixed $\varepsilon > 0$, system (21) is hyperbolic and has three distinct real eigenvalues:

$$u \;-\; c_\varepsilon, \quad u, \quad u \;+\; c_\varepsilon,$$

where

$$c_\varepsilon \;=\; \varepsilon\,\sqrt{g(w)b'(q) \;+\; w\frac{g'(w)b(q)}{q}}.$$

For $\varepsilon > 0$, the solution of the Riemann problem (21), (22) is made up of three waves: shock wave, rarefaction wave, or contact discontinuity (see Figure 1).

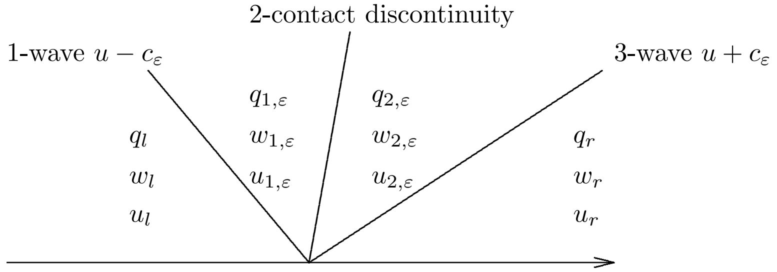


Fig. 1.

3.1.1. Eventuality of shock waves. We consider the case where the solutions of (C_ε) are discontinuous. These discontinuities correspond to shock waves propagating along the trajectories of shock. The velocity of propagation of the shock wave is given by the Rankine–Hugoniot relations:

$$\dot{x} \;=\; \frac{\Delta(qu)}{\Delta q} \;=\; \frac{\Delta(wu)}{\Delta w} \;=\; \frac{\Delta(qu^2+p)}{\Delta(qu)}.$$

Thus we get two equations of compatibility:

$$\Delta u(\overline{q}\Delta w - \overline{w}\Delta q) = 0, \tag{23}$$

$$q_+ q_- \Delta u^2 = \varepsilon^2 \Delta p \Delta q, \tag{24}$$

where q_+, q_- are the values of q here and of one of the three states 1-shock, 2-contact discontinuity, and 3-shock. Using (23) and (24), we deduce the following:

1. On the contact discontinuity,

$$\Delta u = 0 \quad \text{and} \quad \Delta p = 0. \tag{25}$$

2. On the 1-shock or the 3-shock,

$$\Delta u^2 = \frac{\varepsilon^2}{q_+ q_-} \Delta p \Delta q \quad \text{and} \quad \Delta\left(\frac{w}{q}\right) = 0. \tag{26}$$

Knowing that $p = g(w)b(q)$ and that g' and p' are positive, applying the finite increments theorem, we obtain

$$\frac{\Delta p}{\Delta q} \;\geq\; 0.$$

With (26), we deduce

$$\left\{\begin{array}{rcl} \Delta u & = & s\varepsilon \dfrac{\Delta q}{\sqrt{q_+ q_-}} \sqrt{\dfrac{\Delta p}{\Delta q}}, \\ s & = & +1 \;:\; \text{shock with the velocity } u + c_\varepsilon,\ q \text{ increasing } (q_2 > q_r), \\ s & = & -1 \;:\; \text{shock with the velocity } u - c_\varepsilon,\ q \text{ decreasing } (q_1 > q_l). \end{array}\right.$$

3.1.2. Eventuality of rarefaction waves. The rarefaction waves are the smooth solutions of the Riemann problem (21), (22). We express w, u, and p as functions of q:

$$\begin{cases} w = w(q), \\ u = u(q), \\ p = p(q). \end{cases}$$

Replacing the expressions of w, u, and p in (21), we get

$$w(q) = w'(q)q, \tag{27}$$

$$u'^2(q)q^2 = \varepsilon^2 p'(q). \tag{28}$$

Integrating (28) we obtain

$$\begin{cases} u = u_0 + s\varepsilon(\psi(q) - \psi(q_0)), \\ \text{where} \quad \psi(q) = \int_0^q \frac{\sqrt{p'}}{q} dq. \end{cases} \tag{29}$$

Taking $u_0 = u_r$ and $u_0 = u_l$,

$$u = u_r + \varepsilon(\psi(q_{2,\varepsilon}) - \psi(q_r)), \tag{30}$$

$$u = u_l - \varepsilon(\psi(q_{1,\varepsilon}) - \psi(q_l)). \tag{31}$$

3.1.3. Limit on the perturbed problem ($\mathbf{C}_\varepsilon$). We have the following situations.

- Wave $u + c$:
  1. shock if q decreases: $q_2 > q_r$,
  2. rarefaction if q increases: $0 < q_2 < q_r$.
- Wave $u - c$:
  1. shock if q increases: $q_1 > q_l$,
  2. rarefaction if q decreases: $0 < q_1 < q_l$.

In order to characterize the physically allowable solutions, we have the following theorem.

THEOREM 2. *When we study the limit for $\varepsilon \longrightarrow 0$ of the Riemann problem*

$$\begin{cases} q_t + (qu)_x = 0, \\ w_t + (wu)_x = 0, \\ (qu)_t + (qu^2)_x + \varepsilon^2 p_x = 0, \\ (q, w, u) = \begin{cases} q_l, w_l, u_l & for \quad x < 0, \\ q_r, w_r, u_r & for \quad x > 0, \end{cases} \end{cases} \tag{32}$$

supposing that q_l and q_r are not equal to zero, we get the following:

1. *For $u_l > u_r$: The solution of* (39) *converges almost everywhere to the limit:*

$$(q, w, u) = \begin{cases} q_l, w_l, u_l & for \quad x < u^*t, \\ q_r, w_r, u_r & for \quad x > u^*t, \end{cases}$$

where $u^* = \frac{\sqrt{q_l}u_l + \sqrt{q_r}u_r}{\sqrt{q_l} + \sqrt{q_r}}$.

2. *For $u_l < u_r$: We obtain*

$$(q, w, u) = \begin{cases} q_l, w_l, u_l & for \quad x < u_l t, \\ q_r, w_r, u_r & for \quad x > u_r t. \end{cases}$$

The proof is detailed in [1], [2].

The study of this convection step permits us to implement the Lagrange method for the resolution, using the shock velocity u^*.

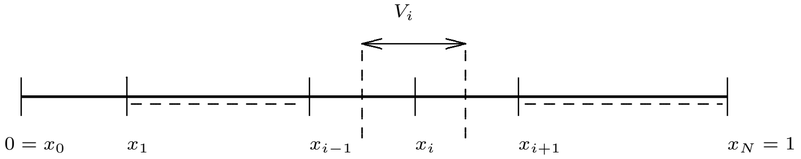


FIG. 2.

3.1.4. Numerical resolution: Finite volume Lagrange method. We denote by Ω the space $\Omega = [0,1]\times[0,T]$. We take a regular initial mesh of [0,1]:

$$0 = x_0 < x_1 < \cdots < x_i < \cdots < x_N = 1, \qquad \text{where } \forall\ i \in [0,N]\ ,\ x_i \ = \ ih.$$

We take a time increment Δt and we define

$$0 = t_0,\ t_1 = \Delta t\ ,\ldots,\ t_n = t_{n-1} + \Delta t\ ,\ldots,\ t_N = T.$$

We have the following definitions:

- $\forall\ i \in [0,N]$, $]x_i,x_{i+1}[$ is *a mesh.*
- $\forall\ i \in [0,N]$, x_i is *a node* of the mesh.
- $\forall\ i \in [0,N]$, V_i is *the volume* corresponding to the space taken by the material assigned to the node x_i.

For $t = 0$, we have the representation, where the V_i are centered on the x_i.

- In each V_i, we define

$$m_i \ = \ \int_{V_i} q\ dx,$$

$$Q_i \ = \ \int_{V_i} qu\ dx,$$

$$W_i \ = \ \int_{V_i} w\ dx.$$

We denote by x_i^n the position of the node x_i at the time t_n. For a quantity α, we shall denote by α_i^n the corresponding value of α. Now that we have the data necessary for the computation, let us describe a cycle of the method. Suppose that we know the solution of the problem

$$\left\{\begin{array}{l} U_t(x,t) + AU(x,t) = 0, \\ U(x,0) = U_0(x). \end{array}\right. \tag{33}$$

With $U_h(.,t_{n-1})$ at t_{n-1}, we want to build the solution at $t_n \ = \ t_{n-1} + \Delta t$. We get the following.

1. *Construction of the mesh at t_n (see Figure* 2*).* For $i \in [0,N]$ we define a new series of nodes x_i^n from the x_i^{n-1}:

$$x_i^n \ = \ x_i^{n-1} \ + \ \Delta t u_i^{n-1}. \tag{34}$$

The moving of the mesh must stay compatible with the computation, which means that the series of $(x_i^n)_{i=0,N}$ must be increasing:

$$x_{i-1}^n < x_i^n. \tag{35}$$

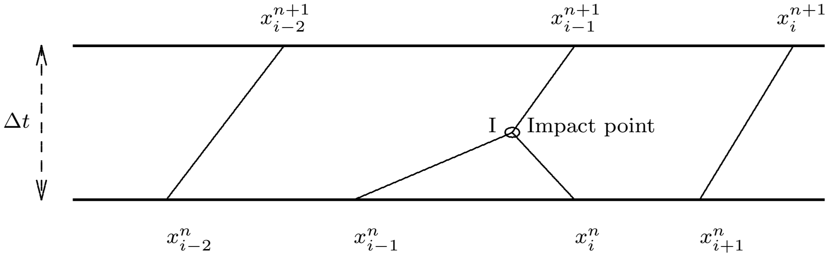


FIG. 3.

That means

$$(x_i^{n-1} - x_{i-1}^{n-1}) \;+\; \Delta t(u_i^{n-1} - u_{i-1}^{n-1}) > 0, \tag{36}$$

which is the condition of stability warranting $x_{i-1}^n < x_i^n$. We must also treat the case where a cell becomes too long and avoid that its length goes beyond a fixed value L:

$$x_i^n \;-\; x_{i-1}^n < L. \tag{37}$$

These conditions (36)–(37) involve a restriction on the time step Δt, so to treat these problems of stability without restriction of time step, we are going to present a method of creation/suppression of cells.

- (a) Suppression of cell (see Figure 3): when $x_i^n \;<\; x_{i-1}^n$.
  - (α) Position of impact point I: I=(x_I,$dt1$) is the point of intersection of the straight lines stem from x_i^{n-1} and x_{i-1}^{n-1}.
  - (β) Determination of the new quantities assigned to I:
    - * $m_I \;=\; m_{i-1}^{n-1} + m_i^{n-1}$.
    - * $Q_I \;=\; Q_{i-1}^{n-1} + Q_i^{n-1}$.
    - * $W_I \;=\; W_{i-1}^{n-1} + W_i^{n-1}$.
    - * $u_I \;=\; \dfrac{\sqrt{q_{i-1}^{n-1}}u_{i-1}^{n-1} \;+\; \sqrt{q_i^{n-1}}u_i^{n-1}}{\sqrt{q_{i-1}^{n-1}} \;+\; \sqrt{q_i^{n-1}}}$ using Theorem 2.
  - (γ) Determination of x_i^n: The new node x_i^n is given by

$$x_i^n \;=\; x_I + (\Delta t - dt1)u_I.$$

- (b) Creation of cell (see Figure 4): When $x_i^n \;-\; x_{i-1}^n > L$, we cut the cell into two parts and we assign new quantities to the new node.
  - (α) Position of the additional point S: $S =$ middle ofx_i^n and x_{i-1}^n.
  - (β) Determination of the new quantities assigned to S:
    - * $V_S \;=\; \frac{x_i^n - x_{i-1}^n}{2}$.
    - * $m_S \;=\; Am_{i-1}^n + Bm_i^n$.
    - * $W_S \;=\; AW_{i-1}^n + BW_i^n$.
    - * $u_S \;=\; \frac{u_{i-1}^n + u_i^n}{2}$.
    - * $q_S \;=\; \frac{m_S}{V_S}$.
    - * $Q_S \;=\; m_S u_S$.
  - (γ) Determination of A and B: In case of a regular mesh it is logical to take A and B constant. In our case two successive cells can

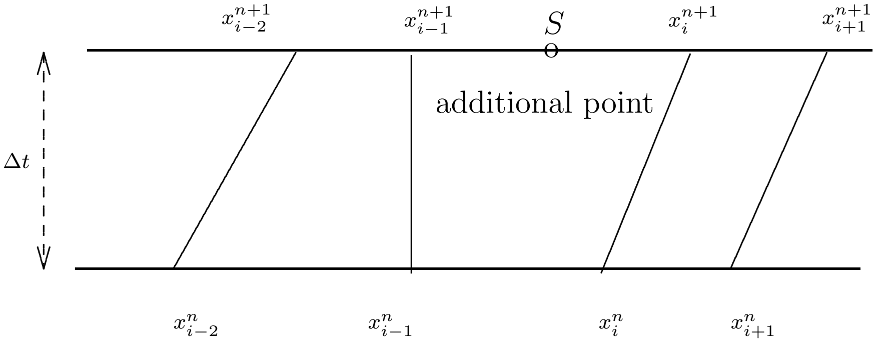


FIG. 4.

have very different volume. Therefore, it seems more advisable to consider A and B as functions of the volume of each cell.

- (δ) Computation of the new quantities assigned to the new mesh:
  * A and B are constant: We compute the three new mass in function of the two old, using the conservation of mass. Thus, we have

$$\begin{cases} m_i^n = \frac{2}{3} m_i^{n-1}, \\ m_{i-1}^n = \frac{2}{3} m_{i-1}^{n-1}, \\ m_S^n = \frac{1}{2} m_{i-1}^n + \frac{1}{2} m_i^n. \end{cases}$$

  * A and B are functions of the volume: We use the point S_{n-1} which is the middle of $[x_{i-1}n-1, x_in-1]$ and we get

$$\begin{cases} m_i^n = \frac{v_i^{n-1}}{V_i^{n-1}} m_i^{n-1}, \\ m_{i-1}^n = \frac{v_{i-1}^{n-1}}{V_{i-1}^{n-1}} m_{i-1}^{n-1}, \\ m_S^n = \frac{v_S^{n-1}}{2V_i^{n-1}} m_i^n + \frac{v_S^{n-1}}{2V_{i-1}^{n-1}} m_{i-1}^n, \end{cases}$$

where

$$\begin{cases} v_S^{n-1} = \frac{x_i^{n-1} - x_{i-1}^{n-1}}{2}, \\ v_{i-1}^{n-1} = V_{i-1}^{n-1} - \frac{v_S^{n-1}}{2}, \\ v_i^{n-1} = V_i^{n-1} - \frac{v_S^{n-1}}{2}. \end{cases}$$

We will compare the results of the two methods in section 4, and we will see that the good one is the second one, where A and B are functions of the volume of the cells.

2. Computation of the new V_i^n.
3. Computation of m_i^n, Q_i^n, W_i^n.
4. Distribution of the density and the speed.

Thus, we obtain the solution at t_n and we close the treatment for the convection.

3.2. The pressure terms. We recall the system (P):

$$\begin{cases} q_t = 0, \\ w_t = 0, \\ (qu)_t + p_x = 0. \end{cases} \tag{38}$$

Just like the system (C), (P) is degenerate: 0 is a triple eigenvalue. Let us consider the perturbed system (P$_\varepsilon$):

$$\left\{\begin{array}{l} q_t + \varepsilon^2 (qu)_x = 0, \\ w_t + \varepsilon^2 (wu)_x = 0, \\ (qu)_t + \varepsilon^2 (qu^2)_x + p_x = 0 \end{array}\right. \tag{39}$$

with the piecewise constant initial data:

$$(q, w, u) \;=\; \left\{\begin{array}{ll} q_l, w_l, u_l & \text{for } \; x < 0, \\ q_r, w_r, u_r & \text{for } \; x > 0. \end{array}\right. \tag{40}$$

For a fixed $\varepsilon > 0$, the system (46) is hyperbolic and has three distinct real eigenvalues:

$$\varepsilon^2 u \;-\; \varepsilon c_\varepsilon, \;\; \varepsilon^2 u, \;\; \varepsilon^2 u \;+\; \varepsilon c_\varepsilon,$$

where

$$c_\varepsilon \;=\; \sqrt{g(w) b'(q) \;+\; w \frac{g'(w) b(q)}{q}}.$$

We proceed as for the system (C$_\varepsilon$) and we consider the discontinuous and the regular solutions of the Riemann problem (39), (40).

3.2.1. Eventuality of shock waves. We have the next three Rankine–Hugoniot relations:

$$\dot{x} \;=\; \frac{\varepsilon^2 \Delta(qu)}{\Delta q} \;=\; \frac{\varepsilon^2 \Delta(wu)}{\Delta w} \;=\; \frac{\varepsilon^2 \Delta(Qu)}{\Delta(Q)} \;+\; \frac{\Delta(p)}{\Delta(Q)},$$

where

$$Q = qu.$$

This gives two equations of compatibility:

$$\Delta u (\overline{q} \Delta w - \overline{w} \Delta q) = 0, \tag{41}$$

$$q_+ q_- \varepsilon^2 \Delta u^2 = \Delta p \Delta q. \tag{42}$$

From (41) and (42), we deduce the following.

1. On the contact discontinuity:

$$\Delta u = 0 \;\text{ and }\; \Delta p = 0. \tag{43}$$

2. On the 1-shock or the 3-shock:

$$\varepsilon^2 \Delta u^2 = \frac{1}{q_+ q_-} \Delta p \Delta q \;\text{ and }\; \Delta\left(\frac{w}{q}\right) = 0. \tag{44}$$

With (44), we get

$$\left\{\begin{array}{rcl} \Delta u & = & s \dfrac{\Delta q}{\varepsilon \sqrt{q_+ q_-}} \sqrt{\dfrac{\Delta p}{\Delta q}}, \\ s & = & +1 \; : \text{ shock with the velocity } u + c_\varepsilon, q \text{ decreasing}, \\ s & = & -1 \; : \text{ shock with the velocity } u - c_\varepsilon, \; q \text{ increasing}. \end{array}\right.$$

Now we are going to study the regular solutions of the Riemann problem (39), (40).

3.2.2. Eventuality of rarefaction waves. We express w, u, and p as functions of q. Replacing w, u, and p in (P_ε), we get

$$w(q) = w'(q)q, \tag{45}$$

$$\varepsilon^2 u'^2(q)q^2 = p'(q). \tag{46}$$

Integrating (46) we obtain

$$\left\{\begin{array}{l} u = u_0 + \frac{s}{\varepsilon}(\psi(q) - \psi(q_0)), \\ \text{where} \quad \psi(q) = \int_0^q \frac{\sqrt{p'}}{q} dq. \end{array}\right. \tag{47}$$

Taking $u_0 = u_r$ and $u_0 = u_l$,

$$u = u_r + \frac{1}{\varepsilon}(\psi(q_{2,\varepsilon}) - \psi(q_r)), \tag{48}$$

$$u = u_l - \frac{1}{\varepsilon}(\psi(q_{1,\varepsilon}) - \psi(q_l)). \tag{49}$$

3.2.3. Limit on the perturbed problem $(\mathbf{P}_\varepsilon)$. We have the following possibilities.

- Wave $u + c$:
  1. shock if q decreases: $q_2 > q_r$,
  2. rarefaction if q increases: $0 < q_2 < q_r$.
- Wave $u - c$:
  1. shock if q increases: $q_1 > q_l$,
  2. rarefaction if q decreases: $0 < q_1 < q_l$.

Therefore, to characterize the solutions, we have the following theorem.

THEOREM 3. *When we study the limit for $\varepsilon \longrightarrow 0$ of the Riemann problem*

$$\left\{\begin{array}{l} q_t + \varepsilon^2(qu)_x = 0, \\ w_t + \varepsilon^2(wu)_x = 0, \\ (qu)_t + \varepsilon^2(qu^2)_x + p_x = 0, \\ (q, w, u) = \left\{\begin{array}{ll} q_l, w_l, u_l & if \quad x < 0, \\ q_r, w_r, u_r & if \quad x > 0, \end{array}\right. \end{array}\right. \tag{50}$$

1. *For $q_l < q_r$: The solution of* (57) *is made up of three waves: shock wave (velocity $u - c$), contact discontinuity, rarefaction wave (velocity $u + c$), and the density is given by*

$$\psi(q_2) - \psi(q_r) + \frac{1}{\sqrt{q_1 q_l}}(q_1 - q_l)\sqrt{\frac{(p - p_l)}{(q_1 - q_l)}};$$

2. *For $q_l > q_r$: The solution of* (57) *is made of three waves: rarefaction wave (velocity $u - c$), contact discontinuity, shock wave (velocity $u + c$), and the density is given by*

$$\psi(q_l) - \psi(q_1) - \frac{1}{\sqrt{q_2 q_r}}(q_2 - q_r)\sqrt{\frac{(p - p_r)}{(q_2 - q_r)}}.$$

The proof is detailed in [1].

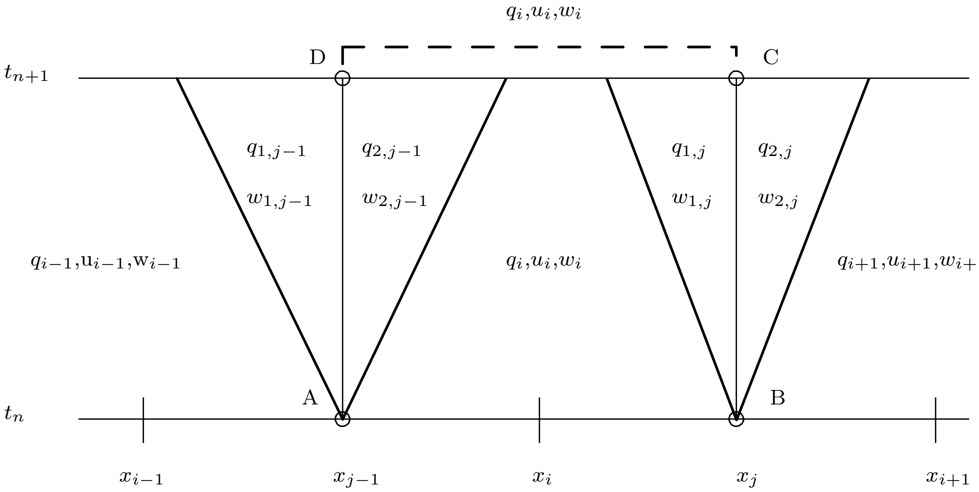


FIG. 5.

3.2.4. Numerical resolution. We denote by Δt and Δx the time and space increments. The resolution of the Riemann problem will give us the values of q and w through the interfaces of the volumes V_i^n, building an approximate Riemann solver. To compute q_i^{n+1}, w_i^{n+1}, and u_i^{n+1}, we integrate on the rectangle (ABCD) the three equations of the system

$$\begin{cases} q_t = 0, \\ w_t = 0, \\ (qu)_t + p_x = 0, \end{cases} \tag{51}$$

and so we get

$$q_i^{n+1} = q_i^n, \tag{52}$$

$$w_i^{n+1} = w_i^n, \tag{53}$$

$$u_i^{n+1} = u_i^n - \frac{\Delta t}{\Delta x}\frac{1}{q_i^n}(p_{j+1}^n - p_j^n). \tag{54}$$

(52), (53), and (54) give the new values of q, w, and u after the treatment of the pressure terms. See Figure 5 for a cycle in time.

3.3. The diffusion term. It remains to solve the following problem:

$$\begin{cases} qu_t = \frac{\partial}{\partial x}(\varepsilon(q,\sigma)u_x), \\ u(x,0) = u_0(x) \quad \text{for } x \text{ in } [0,1], \\ u(0,t) = \alpha \quad \text{and} \quad u(1,t) = 0 \quad \text{for } t \text{ in } [0,T]. \end{cases} \tag{55}$$

3.3.1. The approximate problem. For the space discretization, we use a finite element method P_1. Let us introduce the variational form of the problem.

For any functions ϕ and $\psi \in \mathrm{H}_0^1(\Omega)$ we define the bilinear form $a(.,.)$ by

$$a(\phi,\psi) \;=\; \int_\Omega \varepsilon \phi_x \psi_x \, dx,$$

and the problem to solve is as follows: Determine $u \in \mathrm{H}^1(\Omega)$ with $u(0) = \alpha$ and $u(1) = 0$, such that $\forall v \in \mathrm{H}_0^1(\Omega)$

$$(qu_t, v) \;+\; a(u,v) \;=\; 0. \tag{56}$$

We set

$$\begin{cases} x_i \;=\; ih \text{ for } 0 \le i \le N, \\ K_i \;=\; [x_i, x_{i+1}] \text{ for } 0 \le i \le N, \\ \Delta t \text{ time increment.} \end{cases}$$

We consider the following set:

$$V_h \;=\; \{\; v_h \;\in\; C^0(\Omega)/v_h|_{K_i} \in\; P_1(K_i),\; v_h(0) = 0 \;\; v_h(1) = 0 \;\}.$$

On V_h we have the approximate scalar product:

$$(\phi_h, \psi_h)_h \;=\; h \sum_{i=0}^{N} \phi_i \psi_i \quad \text{for} \;\; \phi_h,\; \psi_h \;\in\; V_h.$$

For ϕ_h, ψ_h $\in \mathrm{V}_h$ we define $\pi_h v$ $\in \mathrm{V}_h$ by

$$\pi_h v(x_i) \;=\; v(x_i), \quad 0 \le i \le N,$$

and we define

$$a_h(\phi_h, \psi_h) \;=\; \frac{1}{h} \sum_{i=0}^{N} \frac{\varepsilon_i + \varepsilon_{i+1}}{2} (\phi_{i+1} - \phi_i)(\psi_{i+1} - \psi_i) \;\; \text{for} \;\; \phi_h,\; \psi_h \;\in\; V_h.$$

For the time discretization we use an implicit Euler's method and we obtain the next approximate problem:

$$\begin{cases} \pi_h(q(u_h^{n+1} - u_h^n), v_h)_h \;+\; \Delta t a_h(u_h^{n+1}, v_h) = 0 \;\; \forall v_h \in V_h, \\ u_h^0 \;=\; \pi_h u_0, \\ u_h^n(0) = \alpha \;\; \text{and} \;\; u_h^n(1) = 0. \end{cases} \tag{57}$$

Thus, we get a linear system with a tridiagonal matrix, which is very easy to solve.

4. Applications and results. This section is divided into two parts. In the first subsection, we apply our splitting method to the hydrodynamic model, especially the Lagrangian treatment of the convection part, with the method of creation/suppression of cells. The second subsection is devoted to the application of the method to the whole system described in section 2.

4.1. Application to the hydrodynamic model. Here, we test the Lagrangian treatment on the convection terms of the hydrodynamic model. There are two tests, one for the suppression of cells, the other one for the creation of cells. In each case, we have the representation of the mesh and the density.

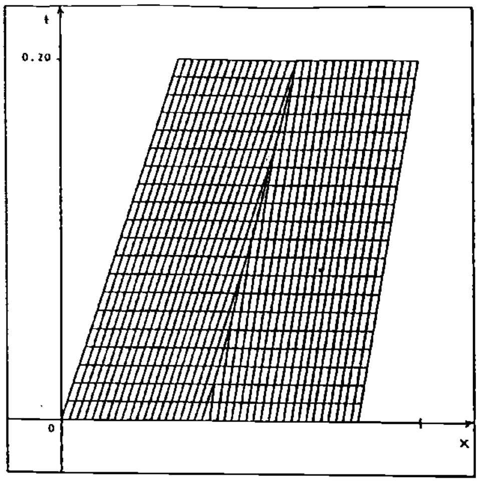


FIG. 6. *Suppression of cells: evolution of the mesh.*

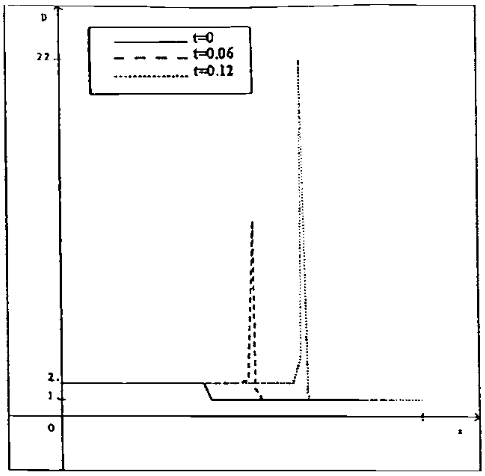


FIG. 7. *Representation of the density.*

4.1.1. Suppression of cells. The test (Figures 6 and 7) is achieved with the data

$$\text{for } x < 0.5 \begin{cases} U_l = 2, \\ \rho_l = 2 \end{cases}$$

and

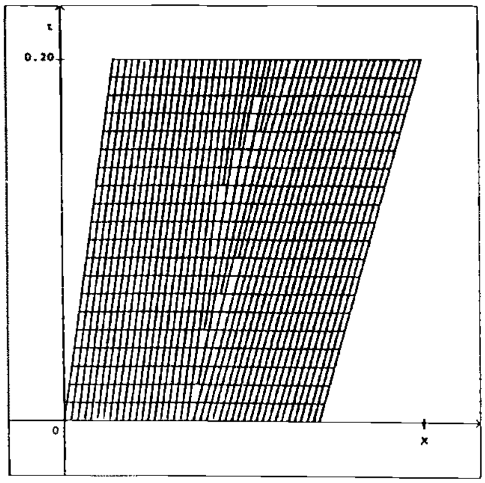


FIG. 8. *Creation of cells: evolution of the mesh.*

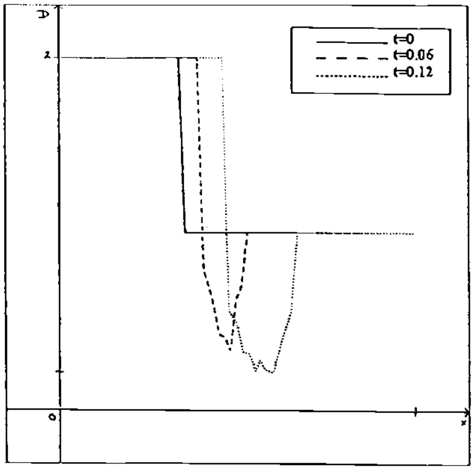


FIG. 9. *Representation of the density with A and B constant.*

$$\text{for } x > 0.5 \begin{cases} U_r = 1, \\ \rho_r = 1. \end{cases}$$

4.1.2. Creation of cells. The test (Figures 8 and 9) is realized with the data

$$\text{for } x < 0.5 \begin{cases} U_l = 1, \\ \rho_l = 2 \end{cases} \quad \text{and for } x > 0.5 \begin{cases} U_l = 2, \\ \rho_l = 1. \end{cases}$$

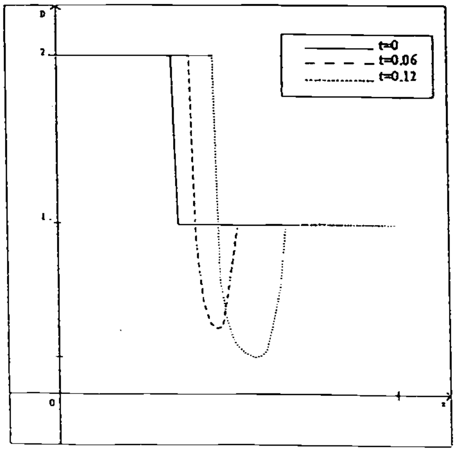


FIG. 10. *Representation of the density with A and B function of volume.*

We can see with these tests that is better to take into account the volume of the cells to compute the quantities assigned to the additional point.

4.2. Application to the compression of pharmaceutical powders. The tests we present here are computed with the whole system (see section 2). The different functions $k(\sigma)$, $p = a(\sigma)b(q)$, and $\varepsilon(\sigma, q)$ for the behavior of powder, the state law, and the diffusion term can be chosen according to the powder. Here we have used the functions

- $k(\sigma) = \sigma$,
- $p = a(\sigma)b(q) = \sigma.q$,
- $\varepsilon(\sigma, q) = \sigma^2.q$.

Figures 11 and 12 show, respectively, the evolution of the porosity in the powder during the compression and the computed pressure in the powder.

We can see that when the porosity approaches to zero (compact powder), it stays with this value, so when the powder is compact, it stays compact, and our goal is achieved.

5. Conclusion. This study leads to implementation of a mathematical model to simulate the compression of pharmaceutical powders. We have obtained a very general model which can adapt to a lot of products. The simulations allow one to infer the behavior of various materials; in particular it permits one to adjust the different parameters used by the model:

- the function $k(\sigma)$ which describes the behavior of powder,
- the state law $p = a(\sigma)b(q)$.

These simulations also permit an analysis of the behavior of the powder during the compression phase [8], especially the equilibrium between the compaction term (pressure term) and the diffusion term, which represents some phenomena occurring during the compression. Further, with these simulation of real experiences, it is possible to

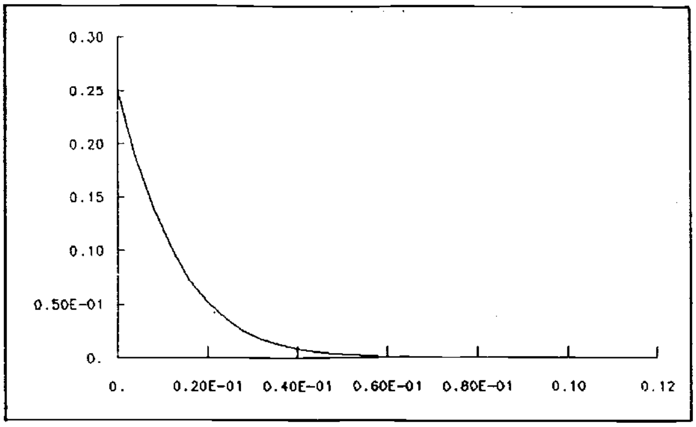


FIG. 11. *Evolution of the porosity.*

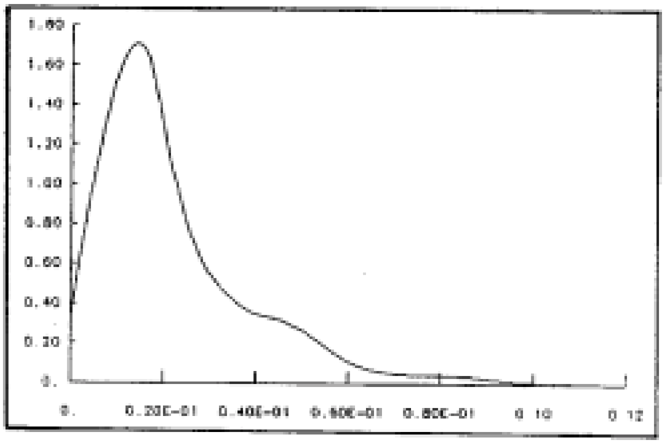


FIG. 12. *Evolution of the pressure (computed).*

characterize the properties of some new products in order to predict their tableting behavior.

REFERENCES

[1] G. BOURDIN, *Application à la production industrielle de comprimés pharmaceutiques*, Thèse, Université Bordeaux I, 1991.

[2] R. BARAILLE, *Développement de schémas numériques adaptés à l'hydrodynamique*, Thèse, Université Bordeaux I, 1991.

[3] R. BARAILLE, G. BOURDIN, F. DUBOIS, AND A.Y. LE ROUX, *Une version à pas fractionnaires du schéma de Godunov pour l'hydrodynamique*, C. R. Acad. Sci. Paris Ser. I Math., 314 (1992), pp. 147–152.

[4] A.F. TROTTER, *On the product of semi-group of operators*, Pacific J. Math., 8 (1958), pp. 887–919.

[5] J.F. Colombeau, *New Generalized Functions and Multiplication of Distributions*, North-Holland, Amsterdam, 1984.

[6] J.F. Colombeau, *Elementary Introduction to New Generalized Functions*, North-Holland, Amsterdam, 1984.

[7] L. Casahoursat, G. Lemagnen, D. Larrouture, and A. Etienne, *Quantification of pharmaceutical powder tabletability*, in III Congreso Internacional de Ciencias Farmaceuticas, Barcelona, 1987.

[8] L. Casahoursat, G. Lemagnen, D. Larrouture, and A. Etienne, *Study of the visco-elastic behaviour of some pharmaceutical powders*, in III Congreso Internacional de Ciencias Farmaceuticas, Barcelona, 1987.

[9] N.A. Armstrong, *Time-dependent factors involved in powder compression and tablet manufacture*, Int. J. Pharma., 49 (1989), pp. 1–13.

[10] G. Bourdin, L. Casahoursat, and M.N. LeRoux, *Mathematical modellization of the compression of pharmaceutical powders*, in Acts of Sixth International Conference on Pharmaceutical Technology, Vol. 3, 1992, pp. 255–261.

SIAM J. Sci. Comput.
Vol. 23, No. 4, pp. 1113–1134

COMPUTATION OF LYAPUNOV-TYPE NUMBERS FOR INVARIANT CURVES OF PLANAR MAPS[*]

K. D. EDOH[†] AND J. LORENZ[‡]

Abstract. Invariant manifolds play an important role in the study of dynamical systems, and it is often crucial to know when they persist under small perturbations of the system. Sufficient conditions for persistence have been formulated in the literature in terms of so-called Lyapunov-type numbers, which measure and compare attractivity rates. The numerical evaluations of these numbers in a simple setting is the subject of this paper. We consider planar diffeomorphisms depending on a parameter; the diffeomorphisms have invariant curves which deform as the parameter changes. The Lyapunov-type numbers are then monitored as functions of parameter and are related to the dynamics on the invariant curves. As an example, we consider the delayed logistic map. We also describe how the invariant curves have been computed.

Key words. invariant curves, invariant manifolds under perturbations, Lyapunov exponents, Lyapunov-type numbers

AMS subject classifications. 65N35, 34C35, 58F27

PII. S1064827500366707

1. Introduction.

Assume that M is an invariant manifold of a diffeomorphism f. If f is perturbed to f_ε, will M persist, i.e., will f_ε have a nearby invariant manifold M_ε diffeomorphic to M? This fundamental question has been studied in various works. Particularly strong results have been obtained by Fenichel [8]. His main assumption is formulated in terms of two Lyapunov-type numbers, $\nu(p)$ and $\sigma(p)$, which are defined for every point $p \in M$ by using the linearized dynamics of f. Roughly speaking, if $\nu(p) < 1$ and $\sigma(p) < 1$ for all $p \in M$ and f_ε is C^1 close to f, then M will persist. This result makes it important to obtain insight into the Lyapunov-type numbers $\nu(p)$ and $\sigma(p)$.

If p is a fixed point or a periodic point of f, then $\nu(p)$ and $\sigma(p)$ can be expressed by the eigenvalues of a suitable matrix. In other cases the determination of $\nu(p)$ and $\sigma(p)$ involves a limit process, which makes the evaluation nontrivial. If p is a periodic point of high period, the numerical evaluation of $\nu(p)$ and $\sigma(p)$ is also far from trivial.

In this paper we consider the simple case where $f : \mathbb{R}^2 \to \mathbb{R}^2$ is a diffeomorphism (or f is a diffeomorphism defined on an open subset of $\mathbb{R}^2$) and $M = \Gamma$ is a smooth, simply closed curve invariant under f. For this case we will define the numbers $\nu(p)$ and $\sigma(p)$, describe some of their properties, and study their numerical approximation; see section 2.

[*] Received by the editors February 3, 2000; accepted for publication (in revised form) March 23, 2001; published electronically October 11, 2001. This work was partially supported by NASA Lewis Research Center, Cleveland, OH (grant NAG32150), by NSF grant DMS–9404124, and by DOE grant DE–FG03–95ER25235.

http://www.siam.org/journals/sisc/23-4/36670.html

[†] Department of Computer Science, Montclair State University, Upper Montclair, NJ 07043 (edoh@pegasus.montclair.edu).

[‡] Department of Mathematics and Statistics, University of New Mexico, Albuquerque, NM 87131 (lorenz@math.unm.edu).

As an interesting example, in section 3 we will consider the delayed logistic map $f_\lambda(x,y) = (y, \lambda y(1-x))$, where λ is a parameter. As λ increases from $\lambda < 2$ to $\lambda > 2$, the fixed point

$$\left(\frac{\lambda-1}{\lambda}, \frac{\lambda-1}{\lambda}\right)$$

of f_λ loses stability, and an invariant curve Γ_λ is born in a Neimark–Sacker bifurcation; see, for example, [9, 12]. The breakdown of Γ_λ, with f_λ embedded in a two-parameter family of maps, has been studied extensively in [2]. In our paper, the main emphasis is the behavior of the Lyapunov-type numbers for Γ_λ when λ crosses an interval of phase locking. In this λ-interval, the map f_λ has two seven-periodic orbits on the curve Γ_λ. This allows us to compare the numerical approximations that we obtain for the Lyapunov-type numbers with values that can be computed more directly from linearizing about the seven-periodic orbits.

The phase-locking λ-interval has a subinterval where the attracting seven-periodic orbit consists of spiral points. For the corresponding λ-values, the invariant curve Γ_λ is not C^1 but merely continuous, as it winds infinitely often about each point of the orbit. In agreement with the general perturbation theory of invariant manifolds, this breakdown of smoothness of Γ_λ is related to a Lyapunov-type number approaching the value 1. As we will show, our numerical computations are in agreement with this observation. We will also show how the Lyapunov-type numbers may be used to distinguish phase-locked dynamics on Γ_λ from ergodic dynamics with irrational rotation number.

In section 4 we describe, for completeness, the algorithm that we used for approximating the invariant curves. The algorithm may be viewed as a discrete version of the Hadamard graph transform [10]. An application, confirming earlier results of [2], is given. For related work on computing invariant curves and more general invariant manifolds we refer to [3, 5, 7, 11, 14, 16, 17, 20, 21], for example.

In principle, the approach described in our paper can be applied to compute Lyapunov-type numbers for Poincaré maps of continuous-time dynamical systems such as the periodically forced oscillator of van der Pol. This work, which is in progress, will extend the results of [6].

The case of an invariant curve of a planar diffeomorphism is clearly a comparatively simple case of an invariant manifold. Nevertheless, as our study for the delayed logistic map shows, the behavior of the Lyapunov-type numbers for a parameter-dependent problem may be quite rich. They can be related to the geometry of the curves (e.g., to the appearance of spiral points) as well as to the dynamics on the curves, which may be phase locked or ergodic. We believe that the numerical approximation of the numbers, $\nu(p)$ and $\sigma(p)$, which we describe here only in a special situation, deserves further attention by researchers working on the numerical analysis of dynamical systems. The numbers are central in the perturbation theory of invariant manifolds and may give considerable insight into the dynamics.

2. Lyapunov-type numbers.

Let $f : \mathbb{R}^2 \longrightarrow \mathbb{R}^2$ denote an orientation preserving diffeomorphism and let $\Gamma \subset \mathbb{R}^2$ denote a simply closed C^1 curve which is invariant under f, i.e., $f(\Gamma) = \Gamma$. In this section we will define four Lyapunov-type numbers, $\nu(p), \bar{\nu}(p), \sigma(p)$, and $\bar{\sigma}(p)$, for every $p \in \Gamma$ and prove some useful properties. Most of the results are not new and could be inferred from more general theorems in the literature. However, since our setting is very special, the proofs given here are simple and, we believe, instructive.

We refer to [6] for similar discussions of fixed points and periodic orbits in continuous-time systems. Constancy of the numbers along orbits is well known [8]. As our setting is very special, our results on limits (see Theorems 2.3 and 2.4) are sharper than more general results on semicontinuous behavior at limit sets [8].

We summarize the results at the end of the section and obtain a rather complete description of the theoretical behavior of the functions $\nu(\cdot), \bar{\nu}(\cdot), \sigma(\cdot)$, and $\bar{\sigma}(\cdot)$.

Notation. For every $p \in \Gamma$, let T_p and N_p denote a unit tangent and a unit normal to Γ at p, respectively (see Figure 2.1). We will assume that T_p and N_p change continuously with p. If we have a smooth parameterization $z(s) = (x(s), y(s)), 0 \leq s \leq L$, of Γ with $\dot{z}(s) \neq 0$ for all s, then T_p and N_p can be obtained as follows: For $p \in \Gamma$ determine the parameter s with $p = z(s)$ and then compute

$$\vec{T}_p = \begin{pmatrix} \dot{x}(s) \\ \dot{y}(s) \end{pmatrix}, \quad T_p = \frac{1}{|\vec{T}_p|}\vec{T}_p, \quad N_p = \frac{1}{|\vec{T}_p|}\begin{pmatrix} \dot{y}(s) \\ -\dot{x}(s) \end{pmatrix}.$$

Here $|\vec{T}_p|^2 = (\dot{x}(s))^2 + (\dot{y}(s))^2$. As a further notation, let A_p denote the Jacobian of f at p. Since f is an orientation-preserving diffeomorphism, we have $\det(A_p) > 0$.

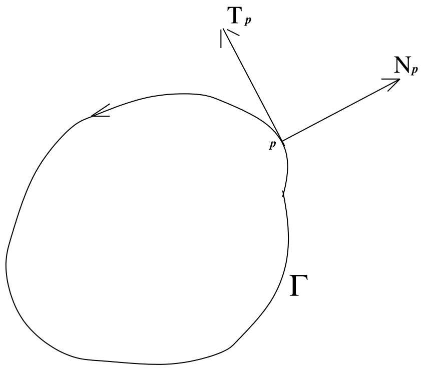


FIG. 2.1. *An invariant circle with the normal N_p and the tangent T_p at the point p.*

Definition of a_p and b_p. Invariance of Γ under f implies that A_pT_p is a multiple of T_{fp},

$$A_pT_p = a_pT_{fp}. \tag{2.1}$$

(Here $fp := f(p)$.) Since f preserves orientation we have[1] $a_p > 0$. The number a_p measures locally contraction (for $0 < a_p < 1$) or stretching (for $a_p > 1$) of the dynamics *within* Γ. One can compute a_p by taking a scalar product, $a_p = \langle T_{fp}, A_pT_p\rangle$. Here $\langle u, v\rangle = u_1v_1 + u_2v_2$.

To measure local contraction or stretching of the dynamics *toward* Γ, we compute A_pN_p and decompose

$$A_pN_p = c_pT_{fp} + b_pN_{fp}. \tag{2.2}$$

[1] Since f is a diffeomorphism, we have $a_p \neq 0$ for all $p \in \Gamma$. Then, by continuity, either $a_p > 0$ for all $p \in \Gamma$ or $a_p < 0$ for all $p \in \Gamma$. Suppose that $a_p < 0$ for all $p \in \Gamma$; then, if p moves around Γ clockwise, the image point fp would move counterclockwise, contradicting our assumption that f preserves orientation.

Then $b_p = \langle N_{fp}, A_p N_p \rangle$ measures this contraction (for $0 < b_p < 1$) or expansion (for $b_p > 1$).

Remark 2.1. Equations (2.1) and (2.2) can be written in matrix form as

$$A_p \Big(T_p \; N_p \Big) = \Big(T_{fp} \; N_{fp} \Big) \begin{pmatrix} a_p & c_p \\ 0 & b_p \end{pmatrix}, \tag{2.3}$$

where T_p, etc., denote column vectors with two components. Here

$$\det \Big(T_p \; N_p \Big) = \det \Big(T_{fp} \; N_{fp} \Big) = \pm 1,$$

and, therefore, $\det(A_p) = a_p b_p$. Since $\det(A_p) > 0$ and $a_p > 0$ we also have $b_p > 0$.

Definition of $\boldsymbol{\nu(p), \bar{\nu}(p), \sigma(p)}$, and $\boldsymbol{\bar{\sigma}(p)}$. Let $p \in \Gamma$. Invariance of Γ under f implies that the complete orbit

$$f^n p, \quad n \in \mathbb{Z}, \tag{2.4}$$

lies on Γ; therefore, the numbers $a_{f^n p}$ and $b_{f^n p}$ are well-defined for all integers n. We will define $\nu(p)$ as a measure of attractivity of the dynamics toward Γ along the orbit (2.4); correspondingly, $\sigma(p)$ will measure the rate of the ratio[2] of attractivity within and toward Γ along the orbit (2.4).

To define $\nu(p)$, one has two possibilities; (1) go forward along the orbit (2.4); or (2) first go backward from p to $f^{-n}p$, then go forward by n steps, and let $n \to \infty$. Both processes lead to useful numbers that are generally not equal. In [8], Fenichel considered *overflowing* invariant manifolds for which forward orbits are not always meaningful; therefore, he used the process (2) to define $\nu(p)$ and $\sigma(p)$. We use similar notation and set

$$\nu(p,n) = \Big(b_{f^{-1}p} \, b_{f^{-2}p} \cdots b_{f^{-n}p} \Big)^{1/n} \tag{2.5}$$

and $\nu(p) = \limsup_{n\to\infty} \nu(p,n)$. If $\nu(p) < 1$, we also define

$$\sigma(p,n) = \frac{\log \Big(a_{f^{-1}p} \cdots a_{f^{-n}p} \Big)}{\log \Big(b_{f^{-1}p} \cdots b_{f^{-n}p} \Big)} \tag{2.6}$$

and $\sigma(p) = \limsup_{n\to\infty} \sigma(p,n)$. Going forward along the orbit (2.4) leads to the following definitions:

$$\bar{\nu}(p,n) = \Big(b_{fp} \cdots b_{f^n p} \Big)^{1/n} \tag{2.7}$$

and $\bar{\nu}(p) = \limsup_{n\to\infty} \bar{\nu}(p,n)$. Furthermore, if $\bar{\nu}(p) < 1$, we set

$$\bar{\sigma}(p,n) = \frac{\log \Big(a_{fp} \cdots a_{f^n p} \Big)}{\log \Big(b_{fp} \cdots b_{f^n p} \Big)} \tag{2.8}$$

and $\bar{\sigma}(p) = \limsup_{n\to\infty} \bar{\sigma}(p,n)$.

[2] In general, this is finer than measuring the ratio of the rates.

Bounds for the Lyapunov-type numbers. Let us show that $\nu(p), p \in \Gamma$, is always a well-defined number and, if $\nu(p) < 1$, then $\sigma(p)$ is also a well-defined number, i.e., the lim sup's are finite. Completely similar arguments apply to $\bar{\nu}(p)$ and $\bar{\sigma}(p)$.

First, by continuity, there are constants $C_j > 1$ with

$$\frac{1}{C_1} \le a_r \le C_1 \quad \text{and} \quad \frac{1}{C_2} \le b_r \le C_2 \qquad \text{for all} \quad r \in \Gamma.$$

This implies $1/C_2 \le \nu(p,n) \le C_2$ for all n and, therefore, $\nu(p)$ is well-defined with $1/C_2 \le \nu(p) \le C_2$. Now assume $\nu(p) < 1$ and choose $\varepsilon > 0$ with $\nu(p) + \varepsilon < 1$. Then we have, for all large n,

$$0 < b_{f^{-1}p} \cdots b_{f^{-n}p} \le \Big(\nu(p) + \varepsilon\Big)^n < 1;$$

thus

$$\log(b_{f^{-1}p} \cdots b_{f^{-n}p}) \le n \log\Big(\nu(p) + \varepsilon\Big) < 0,$$

and thus

$$\frac{1}{|\log(b_{f^{-1}p} \cdots b_{f^{-n}p})|} \le \frac{1}{n \log\Big(\frac{1}{\nu(p)+\varepsilon}\Big)}.$$

Also, from $1/C_1 \le a_r \le C_1$ for all $r \in \Gamma$, it follows that $|\log(a_{f^{-1}p} \cdots a_{f^{-n}p})| \le n \log C_1$. Since $\varepsilon > 0$ was arbitrary, the previous bounds yield $|\sigma(p,n)| \le \log C_1 / \log(1/\nu(p))$. Therefore, $\sigma(p)$ is a well-defined number satisfying the same bound. The number $\sigma(p)$ may be positive, negative, or zero. For an illustration, assume

$$1 > a_p = \exp(-\alpha) > 0 \quad \text{and} \quad 1 > b_p = \exp(-\beta) > 0$$

with $\alpha > 0, \beta > 0$ independently of p. Then we have

$$0 < \nu(p,n) = \exp(-\beta) < 1 \quad \text{and} \quad 0 < \sigma(p,n) = \frac{\alpha}{\beta}.$$

One obtains that $\sigma(p) < 1$ iff $0 < \alpha < \beta$, i.e., if the attraction in the normal direction is stronger than in the tangential direction.

Fixed points. A simple case occurs if $p \in \Gamma$ is a fixed point of f, i.e., $fp = p$. Since $f^{-j}p = p$ for $j = 1, \dots, n$ we have (see (2.5) and (2.6))

$$\nu(p,n) = b_p, \quad \sigma(p,n) = \frac{\log a_p}{\log b_p};$$

thus $\nu(p) = b_p$ and $\sigma(p) = \frac{\log a_p}{\log b_p}$. Clearly, the same expressions are obtained for $\bar{\nu}(p)$ and $\bar{\sigma}(p)$. Furthermore, by (2.3), the numbers a_p and b_p are the eigenvalues of A_p. Here a_p is the eigenvalue to the eigenvector T_p, and b_p is the other eigenvalue. We summarize the result.

Theorem 2.1. *Let $p = fp$ and let $A_p = f'(p)$ denote the Jacobian of f. Furthermore, let a_p and b_p denote the eigenvalues of A_p, where $A_pT_p = a_pT_p$ with T_p the tangent vector to Γ at p. Then we have*

$$\nu(p) = \bar{\nu}(p) = b_p \quad \textit{and} \quad \sigma(p) = \bar{\sigma}(p) = \frac{\log a_p}{\log b_p}.$$

Periodic points. Let $p \in \Gamma$ denote a point of period q of f, i.e.,

$$fp \neq p, \quad f^2p \neq p, \ldots, f^{q-1}p \neq p, \quad f^qp = p.$$

Introduce the orthogonal matrix $Q_p = (T_p \ N_p)$ and the upper triangular matrix

$$\mathcal{T}_p = \begin{pmatrix} a_p & c_p \\ 0 & b_p \end{pmatrix}.$$

According to (2.3) we have $A_pQ_p = Q_{fp}\mathcal{T}_p$. Replacing p by fp we also have $A_{fp}Q_{fp} = Q_{f^2p}\mathcal{T}_{fp}$. Therefore,

$$A_{fp}A_pQ_p = Q_{f^2p}\mathcal{T}_{fp}\mathcal{T}_p.$$

The argument can be repeated. If Π_p denotes the product matrix

$$\Pi_p = A_{f^{q-1}p} \cdots A_{fp}A_p, \tag{2.9}$$

then

$$\Pi_p\mathcal{O}_p = \mathcal{O}_{f^qp}\mathcal{T}_{f^{q-1}p} \cdots \mathcal{T}_{fp}\mathcal{T}_p. \tag{2.10}$$

By assumption, $f^qp = p$. Therefore, (2.10) says that the matrix product (2.9) is similar to the upper triangular matrix $\mathcal{T}_{f^{q-1}p} \cdots \mathcal{T}_{fp}\mathcal{T}_p$. Consequently, the eigenvalues of Π_p are

$$\alpha_p = a_p\, a_{fp} \, \cdots \, a_{f^{q-1}p} \quad \text{and} \quad \beta_p = b_p\, b_{fp} \, \cdots \, b_{f^{q-1}p}.$$

(Here $\Pi_pT_p = \alpha_pT_p$.) It follows that $\bar{\nu}(p,q) = \beta_p^{1/q}$ and $\bar{\sigma}(p,q) = \frac{\log \alpha_p}{\log \beta_p}$.

Using the q-periodicity of f^np, it is not difficult to show that we also have

$$\nu(p) = \bar{\nu}(p) = \beta_p^{1/q} \tag{2.11}$$

and

$$\sigma(p) = \bar{\sigma}(p) = \frac{\log \alpha_p}{\log \beta_p}. \tag{2.12}$$

Therefore, we have obtained the following result.

THEOREM 2.2. *Let*

$$\mathcal{O}(p) = \Big\{p, fp, f^2p, \ldots, f^{q-1}p\Big\}$$

denote an orbit of period q of f. Form the matrix product Π_p in (2.9)*, where $A_{f^jp} = f'(f^jp)$. Let α_p, β_p denote the eigenvalues of Π_p with α_p corresponding to the eigenvector T_p which is tangent to Γ at p. Then we have the formulas* (2.11) *and* (2.12) *for the Lyapunov-type numbers.*

Constancy along orbits. Each of the functions $\nu(\cdot), \bar{\nu}(\cdot), \sigma(\cdot)$, and $\bar{\sigma}(\cdot)$ is constant along each orbit $f^np, \ p \in \Gamma$. To show this for $\bar{\nu}$, for example, note that

$$\Big(\bar{\nu}(p,n)\Big)^n = b_{fp} \ldots b_{f^np} \quad \text{and} \quad \Big(\bar{\nu}(fp,n)\Big)^n = b_{f^2p} \ldots b_{f^{n+1}p}.$$

Therefore,

$$\left(\frac{\bar{\nu}(p,n)}{\bar{\nu}(fp,n)}\right)^n = \frac{b_{fp}}{b_{f^{n+1}p}},$$

and the bound $C_2^{-1} \leq b_r \leq C_2$ for all r implies

$$C_2^{-2/n} \leq \frac{\bar{\nu}(p,n)}{\bar{\nu}(fp,n)} \leq C_2^{2/n}.$$

For $n \to \infty$ it follows that $\bar{\nu}(p) = \bar{\nu}(fp)$, which yields constancy of $\bar{\nu}(\cdot)$ along the orbit through p. The arguments for the functions ν, σ, and $\bar{\sigma}$ are similar.

Behavior at limits. Assume that an orbit $f^n p$, $p \in \Gamma$, approaches a fixed point p_+ of f as $n \to \infty$. We claim that, in this case, $\bar{\nu}(p) = \bar{\nu}(p_+)$ and $\bar{\sigma}(p) = \bar{\sigma}(p_+)$. A similar result holds for ν and σ if $f^n p \to p_-$ as $n \to -\infty$.

THEOREM 2.3. *If $f^n p \to p_+$ as $n \to \infty$, then*

$$\bar{\nu}(p) = \bar{\nu}(p_+) = \nu(p_+) \quad \textit{and} \quad \bar{\sigma}(p) = \bar{\sigma}(p_+) = \sigma(p_+).$$

Similarly, if $f^n p \to p_-$ as $n \to -\infty$, then

$$\nu(p) = \bar{\nu}(p_-) = \nu(p_-) \quad \textit{and} \quad \sigma(p) = \bar{\sigma}(p_-) = \sigma(p_-).$$

Proof. Note that the equations $\bar{\nu}(p_+) = \nu(p_+)$, etc., at the fixed points have already been shown in Theorem 2.1. We will prove only that $\bar{\nu}(p) = \bar{\nu}(p_+)$ if $f^n p \to p_+$ as $n \to \infty$. The other claims follow similarly. To show the theorem, we first note that, for any $N \in \mathbf{N}$,

$$\log \bar{\nu}(f^N p, n) = \frac{1}{n} \sum_{j=1}^{n} \log b_{f^{N+j}p}. \tag{2.13}$$

Let $\varepsilon > 0$ be given. Continuity of the function $r \to \log b_r$ implies that $|\log b_{f^{N+j}p} - \log b_{p_+}| \leq \varepsilon$ for all $j \geq 1$ if $N = N(\varepsilon)$ is sufficiently large. Therefore, (2.13) yields $|\log \bar{\nu}(f^N p, n) - \log \bar{\nu}(p_+)| \leq \varepsilon$ for all $n \geq 1$. As $n \to \infty$ one obtains $|\log \bar{\nu}(f^N p) - \log \bar{\nu}(p_+)| \leq \varepsilon$. However, since $\bar{\nu}(\cdot)$ is constant along orbits, $\bar{\nu}(f^N p) = \bar{\nu}(p)$, thus $|\log \bar{\nu}(p) - \log \bar{\nu}(p_+)| \leq \varepsilon$. Since $\varepsilon > 0$ was arbitrary, the equality $\bar{\nu}(p) = \bar{\nu}(p_+)$ follows. □

Orbits that approach periodic orbits. Assume that f has an orbit of period q,

$$\mathcal{O}(p_+) = \left\{p_+, fp_+, f^2 p_+, \ldots, f^{q-1} p_+\right\}, \tag{2.14}$$

which is approached by $f^n p$ as $n \to \infty$,

$$dist\left(f^n p, \mathcal{O}(p_+)\right) \to 0 \quad \text{as} \quad n \to \infty.$$

(Here $dist(f^n p, \mathcal{O}(p_+)) = \min_j |f^n p - f^j p_+|$.) We claim that $\bar{\nu}(p) = \bar{\nu}(p_+)$ and $\bar{\sigma}(p) = \bar{\sigma}(p_+)$, and that a similar result holds for ν and σ if $f^n p$ approaches a periodic orbit $\mathcal{O}(p_-)$ as $n \to -\infty$. To prove this, we first note that the q points of the orbit (2.14) are all fixed points of f^q,

$$f^q(f^j p_+) = f^j p_+, \quad j = 0, \ldots, q-1. \tag{2.15}$$

As $n \to \infty$, the sequence

$$f^{qn}p, \quad n = 1, 2, \ldots, \tag{2.16}$$

approaches precisely *one* of the fixed points (2.15). To conclude this, we use that the dynamics of f restricted to Γ is one-dimensional. This implies that the iteration (2.16) always progresses *monotonically* between any two fixed points of f^q. An approach of the sequence (2.16) to the orbit (2.14) which oscillates between two or more fixed points of f^q is, therefore, impossible.

Without loss of generality (by renaming p_+, if necessary) we may assume that $f^{qn}p \to p_+$ as $n \to \infty$. This yields

$$f^{qn+j}p \to f^j p_+ \quad \text{as} \quad n \to \infty \quad \text{for} \quad j = 0, \ldots, q-1.$$

We have $\bar{\nu}(p_+) = \bar{\nu}(fp_+) = \cdots = \bar{\nu}(f^{q-1}p_+)$, and a limit argument as in the proof of Theorem 2.3 then yields the equality $\bar{\nu}(p) = \bar{\nu}(p_+)$.

THEOREM 2.4. *Let $p \in \Gamma$. Assume that $f^n p$ approaches a periodic orbit*

$$\{p_+, fp_+, \ldots, f^{q-1}p_+\}$$

of f as $n \to \infty$. Then we have

$$\bar{\nu}(p) = \bar{\nu}(p_+) = \nu(p_+) \quad \textit{and} \quad \bar{\sigma}(p) = \bar{\sigma}(p_+) = \sigma(p+).$$

Similarly, if $f^n p$ approaches a periodic orbit $\{p_-, fp_-, \ldots, f^{q-1}p_-\}$ as $n \to -\infty$, then

$$\nu(p) = \bar{\nu}(p_-) = \nu(p_-) \quad \textit{and} \quad \sigma(p) = \bar{\sigma}(p_-) = \sigma(p_-).$$

Dense orbits. The restriction of f to the invariant curve Γ can be identified with an orientation-preserving circle diffeomorphism, which we denote by F. To be precise, assume that $z(s) = (x(s), y(s))$, $0 \le s \le L$, parameterizes Γ by arclength s. Then, for any $0 \le s \le L$, we can write $f(z(s)) = z(\tilde{s})$ with $\tilde{s} = \tilde{s}(s)$. The normalization $F(s/L) = \tilde{s}(s)/L, 0 \le s/L \le 1$, determines the circle diffeomorphism F, where the circle is identified with $\mathbb{R} \bmod \mathbb{Z}$.

Let $\rho(F)$ denote the rotation number of F (see, e.g., [1, 13]). A well-known result of Poincaré states that $\rho(F)$ is irrational iff the orbits of F are dense on the circle. (See, for example, [1, 13].) Consequently, if $\rho(F)$ is irrational, the orbits $f^n p$ with $p \in \Gamma$ are dense on Γ. We will then show, under a mild smoothness assumption, that the Lyapunov-type numbers $\nu(p)$, etc., are constant functions on Γ.

The result is closely related to Birkhoff's ergodic theorem; see, for example, [18]. In general, however, one obtains only an *almost-everywhere* result under the assumptions of Birkhoff's theorem. Since one computes only on a set of (Lebesgue) measure zero, it is of some interest to know that the *almost-everywhere* restriction may be dropped under our special assumptions.

THEOREM 2.5. *Assume that $\mu := \rho(F)$ is irrational and that $F \in C^2$. Then the functions $\nu(p) \equiv \bar{\nu}(p)$ and $\sigma(p) \equiv \bar{\sigma}(p)$ are constant on Γ.*

Remark 2.2. The assumption $F \in C^2$ can be weakened. It suffices to assume that $dF/d\xi$ is of bounded variation. The assumption $F \in C^1$, however, is not sufficient for the application of Denjoy's theorem below.

Our proof of Theorem 2.5 is based on the following lemma.

LEMMA 2.6. *Let $\mu \in \mathbb{R}$ be irrational and let $\phi : \mathbb{R} \to \mathbf{C}$ denote a continuous, 1-periodic function. Then we have, for all $x \in \mathbb{R}$,*

$$\frac{1}{n}\sum_{j=1}^{n} \phi(x+\mu j) \to \int_0^1 \phi(\xi)\,d\xi \quad as \quad n \to \infty. \tag{2.17}$$

The convergence in (2.17) *is uniform in x.*

Proof. (a) First let $\phi(x) = e^{2\pi ikx}$ with integer k. For $k \neq 0$ we have

$$\sum_{j=1}^{n} \phi(x+\mu j) = e^{2\pi ikx}\,\frac{q^{n+1}-q}{q-1} \quad \text{with} \quad q = e^{2\pi ik\mu}.$$

(The irrationality of μ ensures $q \neq 1$, of course.) It follows that

$$\Big|\sum_{j=1}^{n} \phi(x+\mu j)\Big| \le \frac{2}{|q-1|}$$

is bounded uniformly in n and x. Therefore, convergence in (2.17) to $0 = \int_0^1 \phi\,d\xi$ is uniform in x.

If $k = 0$, then $\phi \equiv 1$, and uniform convergence in (2.17) is immediate.

(b) Consider any finite sum

$$\phi_K(x) = \sum_{k=-K}^{K} c_k\, e^{2\pi ikx}. \tag{2.18}$$

Applying (a), we obtain uniform convergence in (2.17) for $\phi = \phi_K$.

(c) Let ϕ be continuous and 1-periodic. Given any $\varepsilon > 0$, there is a function ϕ_K of the form (2.18) with[3]

$$|\phi - \phi_K|_\infty \le \varepsilon. \tag{2.19}$$

Then, by (b), there is an integer N with

$$\Big|\ \frac{1}{n}\sum_{j=1}^{n} \phi_K(x+\mu j)\ -\ \int_0^1 \phi_K\,d\xi\ \Big| \le \varepsilon$$

for all $n \ge N$ and all x. Together with (2.19) we obtain

$$\Big|\ \frac{1}{n}\sum_{j=1}^{n} \phi(x+\mu j)\ -\ \int_0^1 \phi\,d\xi\ \Big|\ \le\ 3\varepsilon,$$

and the lemma is proved. □

To prove Theorem 2.5, let $\phi : \mathbb{R} \to \mathbb{R}$ denote any continuous, 1-periodic function and consider the averages

$$\frac{1}{n}\sum_{j=1}^{n} \phi(F^j y). \tag{2.20}$$

[3] Here $|\psi|_\infty = \max_\xi |\psi(\xi)|$ denotes the maximum norm.

Let $R_\mu = R_\mu(x)$ denote the rigid rotation of the circle by angle $2\pi\mu$, i.e., $R_\mu(x) = (x + \mu) \bmod 1$ since the circle is identified with $\mathbb{R} \bmod \mathbb{Z}$. By Denjoy's theorem (see [4] or, for example, [1, 13]) the circle map F is topologically conjugate to R_μ, i.e., there is a circle homeomorphism H with $F = H^{-1} \circ R_\mu \circ H$. We then obtain $F^j = H^{-1} \circ R_\mu^j \circ H$, thus $F^j y = H^{-1}(Hy + \mu j)$. Substitution into (2.20) yields

$$\frac{1}{n} \sum_{j=1}^{n} \phi(F^j y) = \frac{1}{n} \sum_{j=1}^{n} \phi \circ H^{-1}(Hy + \mu j),$$

and, therefore, by Lemma 2.6 the averages (2.20) converge,

$$\frac{1}{n} \sum_{j=1}^{n} \phi(F^j y) \to \int_0^1 \phi(H^{-1}(\xi))\, d\xi. \tag{2.21}$$

The convergence is uniform in y.

Now recall that $r \to b_r$ is a continuous, positive function on Γ and

$$\bar{\nu}(p,n) = \Big(b_{fp} \dots b_{f^n p}\Big)^{1/n}.$$

Therefore,

$$\log \bar{\nu}(p,n) = \frac{1}{n} \sum_{j=1}^{n} \log b_{f^j p}.$$

When we identify the invariant curve Γ with the circle $\mathbb{R} \bmod \mathbb{Z}$, the function $r \to b_r$ corresponds to a positive, continuous, 1-periodic function $\beta(\xi)$, and $\log b_{f^j p}$ corresponds to $\log \beta(F^j y)$. With these identifications we have

$$\log \bar{\nu}(p,n) = \frac{1}{n} \sum_{j=1}^{n} \log \beta(F^j y).$$

Therefore, by (2.21), the sequence $\log \bar{\nu}(p,n)$ converges to the integral

$$\text{Int} := \int_0^1 \phi(H^{-1}(\xi))\, d\xi \quad \text{with} \quad \phi(y) = \log \beta(y).$$

In particular,

$$\bar{\nu}(p) = \limsup_{n \to \infty} \bar{\nu}(p,n) = e^{Int}$$

is independent of p. We claim that we also have $\nu(p) = e^{Int}$ for all $p \in \Gamma$. To prove this, first note that

$$\frac{1}{n} \sum_{j=1}^{n} \phi(x - \mu j) \to \int_0^1 \phi(\xi)\, d\xi \quad \text{as} \quad n \to \infty$$

under the assumptions of Lemma 2.6. The arguments, then, are the same as before,

$$\begin{aligned} \log \nu(p,n) &= \frac{1}{n} \sum_{j=1}^{n} \log b_{f^{-j} p} \\ &= \frac{1}{n} \sum_{j=1}^{n} \phi(F^{-j} y) \to \int_0^1 \phi(H^{-1}(\xi))\, d\xi \quad \text{as} \quad n \to \infty. \end{aligned}$$

Here the functions ϕ, F, and H are exactly the same as above.

To prove $\sigma(p) = \bar{\sigma}(p) = \text{const}$ for $p \in \Gamma$ we note that

$$\bar{\sigma}(p,n) \;=\; \frac{\frac{1}{n}\sum_{j=1}^{n} \log a_{f^j p}}{\frac{1}{n}\sum_{j=1}^{n} \log b_{f^j p}}.$$

The denominator and the numerator can be treated separately by the same arguments as above. This completes the proof of Theorem 2.5. □

Summary and discussion. As above, let F denote the circle map corresponding to the restriction f_Γ of f to Γ, and let $\rho(F)$ denote the rotation number of F.

If $\rho(F)$ is irrational, then all orbits $f^n p, p \in \Gamma$, are dense on Γ, and we have

$$\nu(p) = \bar{\nu}(p) = C_1, \quad \sigma(p) = \bar{\sigma}(p) = C_2$$

with constants C_1, C_2 independent of p.

For the further discussion assume that $\rho(F) = \frac{m}{q}$ is rational, where the integers m and q have no common divisor. Then f_Γ has at least one orbit of period q. (A fixed point is considered as an orbit of period 1.) Furthermore, *every* periodic orbit of f_Γ has period q. On such an orbit, $\mathcal{O}(p) = \{p, fp, \dots, f^{q-1}p\}$, the Lyapunov-type numbers are constant, and the two numbers $\nu(p) = \bar{\nu}(p)$ and $\sigma(p) = \bar{\sigma}(p)$ can be computed in terms of the eigenvalues of a 2×2 matrix. This matrix is obtained by linearizing f about $\mathcal{O}(p)$.

If the orbit $f^n p, p \in \Gamma$, is not periodic, then it approaches a q-periodic orbit $\mathcal{O}(p_+)$ as $n \to \infty$ and a q-periodic orbit $\mathcal{O}(p_-)$ as $n \to -\infty$. Typically, the two limit orbits, $\mathcal{O}(p_+)$ and $\mathcal{O}(p_-)$, are not the same. Then one can expect that

$$\nu(p_+) = \bar{\nu}(p_+) \neq \bar{\nu}(p_-) = \nu(p_-)$$

and

$$\sigma(p_+) = \bar{\sigma}(p_+) \neq \bar{\sigma}(p_-) = \sigma(p_-)$$

since the linearizations of f about $\mathcal{O}(p_+)$ and $\mathcal{O}(p_-)$ are not directly related.

The Lyapunov-type numbers are constant along the orbit $\{f^n p\}_{n\in\mathbb{Z}}$, and we have shown that

$$\begin{aligned} \nu(f^n p) &= \nu(p_-), \quad & \bar{\nu}(f^n p) &= \bar{\nu}(p_+), \\ \sigma(f^n p) &= \sigma(p_-), \quad & \bar{\sigma}(f^n p) &= \bar{\sigma}(p_+). \end{aligned}$$

For an illustration, assume that $\nu(p_+) > \nu(p_-)$. (In the next section we will give a simple example where this occurs for two seven-periodic orbits $\mathcal{O}(p_+)$ and $\mathcal{O}(p_-)$.) Since $f^n p$ approaches $\mathcal{O}(p_+)$ for $n \to \infty$, but

$$\nu(f^n p) = \nu(p_-) < \nu(p_+) \quad \text{for all} \quad n,$$

the function $\nu(\cdot)$ cannot be continuous at the points of the periodic orbit $\mathcal{O}(p_+)$. (For the example, the resulting jump discontinuities of $\nu(\cdot)$ at the seven points of $\mathcal{O}(p_+)$ are shown in Figure 2.3.) See Figures 2.2–2.5.

This piecewise constant and discontinuous behavior is typical for the functions

$$\nu(\cdot), \quad \bar{\nu}(\cdot), \quad \sigma(\cdot), \quad \bar{\sigma}(\cdot) \tag{2.22}$$

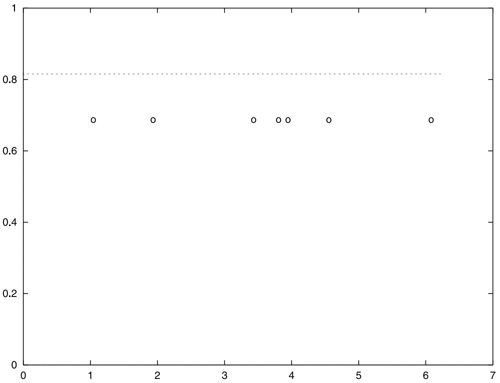


FIG. 2.2. *The values of* $\bar{\nu}(\theta)$ *for* $\lambda =$ 2.18 *and* $0 \leq \theta \leq 2\pi$.

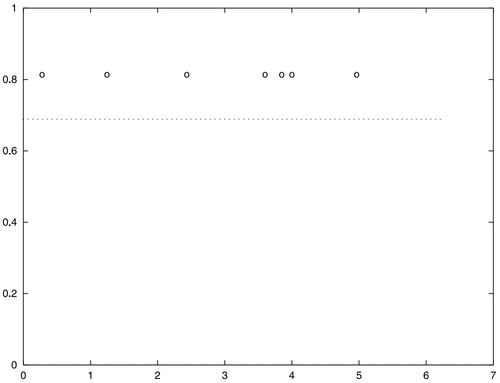


FIG. 2.3. *The values of* $\nu(\theta)$ *for* $\lambda =$ 2.18 *and* $0 \leq \theta \leq 2\pi$.

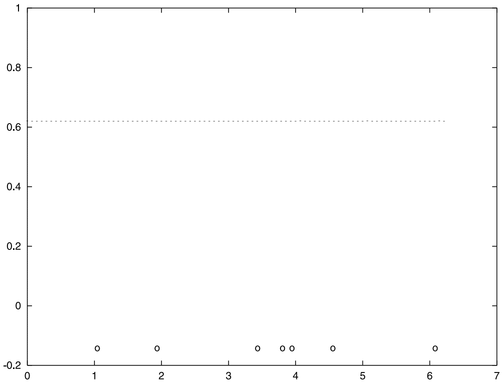


FIG. 2.4. *The values of* $\bar{\sigma}(\theta)$ *for* $\lambda =$ 2.18 *and* $0 \leq \theta \leq 2\pi$.

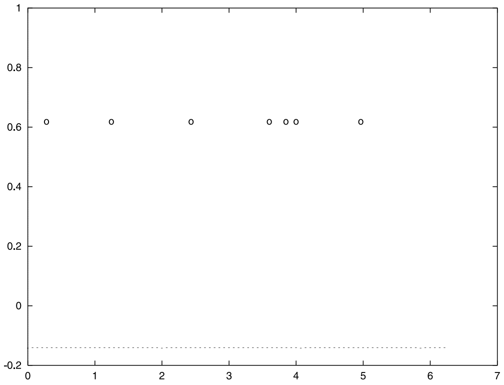


FIG. 2.5. *The values of* $\sigma(\theta)$ *for* $\lambda =$ 2.18 *and* $0 \leq \theta \leq 2\pi$.

in the phase-locked situation under consideration. To see this, assume that p_1 and p_2 are two fixed points of f^q, i.e., $p_{1,2}$ belong to q-periodic orbits of f on Γ. Denote by $\mathcal{C} = \mathcal{C}(p_1, p_2)$ the segment of Γ which lies strictly between p_1 and p_2 if one moves from p_1 to p_2 along Γ counterclockwise, say. Assume that $\mathcal{C}$ does not contain any fixed point of f^q. Then we have the following alternative: either (a) for all $p \in \mathcal{C}$,

$$\lim_{n\to\infty} f^{nq} p = p_2 \quad \text{and} \quad \lim_{n\to-\infty} f^{nq} p = p_1$$

or (b) for all $p \in \mathcal{C}$,

$$\lim_{n\to\infty} f^{nq} p = p_1 \quad \text{and} \quad \lim_{n\to-\infty} f^{nq} p = p_2.$$

In case (a) we have, for all $p \in \mathcal{C}$,

$$\begin{aligned} \nu(p) &= \nu(p_1), \quad \bar{\nu}(p) = \bar{\nu}(p_2), \\ \sigma(p) &= \sigma(p_1), \quad \bar{\sigma}(p) = \bar{\sigma}(p_2). \end{aligned}$$

Similarly, in case (b),

$$\begin{aligned} \nu(p) &= \nu(p_2), \quad \bar{\nu}(p) = \bar{\nu}(p_1), \\ \sigma(p) &= \sigma(p_2), \quad \bar{\sigma}(p) = \bar{\sigma}(p_1). \end{aligned}$$

This shows that the functions (2.22) are constant between consecutive fixed points of f^q on Γ.

Let us summarize the results: If $\rho(F)$ is irrational, then $\nu(p) = \bar{\nu}(p)$ and $\sigma(p) = \bar{\sigma}(p)$ are constant on Γ. If $\rho(F) = m/q$ is rational, then the functions (2.22) are piecewise constant, and jumps may occur only at fixed points of f^q. If $\rho(F)$ is rational, then, in general,

$$\nu(p) - \bar{\nu}(p) \quad \text{and} \quad \sigma(p) - \bar{\sigma}(p) \tag{2.23}$$

differ from zero. This last observation is numerically important, because we can compute good approximations of the numbers (2.23). If the differences (2.23) are significantly different from zero, we may conclude that we have phase-locking of f on Γ. Conversely, if the differences (2.23) are close to zero with high accuracy, then either $\rho(F)$ is irrational or we may expect that $\rho(F) = m/q$ with large q. (This last conclusion is not rigorous and needs further investigation.) In the latter case, where $\rho(F) = m/q$ with large q, the dynamics of f on Γ is close to the ergodic case where $\rho(F)$ is irrational. In any case, the numbers (2.23), and the amount by which they differ from zero, are numerically computable indicators of the degree of phase-locking of f on Γ. We will demonstrate this by an example in the next section.

3. Numerical results for the delayed logistic map. A simple model for the discrete-time evolution of the size of a population is given by $N_{n+1} = \lambda N_n(1 - N_{n-1})$. Here N_n is the scaled size of the population in the nth generation and $\lambda > 0$ is a parameter. Define $f_\lambda : \mathbb{R}^2 \to \mathbb{R}^2$ by $f_\lambda(x, y) = (y, \lambda y(1 - x))$. If one sets $(x_n, y_n) = (N_{n-1}, N_n)$, then the evolution for N_n corresponds to the planar map

$$(x_n, y_n) \to f_\lambda(x_n, y_n) = (x_{n+1}, y_{n+1}).$$

The map $f_\lambda, \lambda > 0$, has the fixed points, $P_1 = (0, 0)$ and $P_2 = (1 - \frac{1}{\lambda})(1, 1)$. Of interest is the fixed point $P_2 = P_2(\lambda)$, which is asymptotically stable for $0 < \lambda < 2$ and unstable for $\lambda > 2$. At $\lambda = 2$, where $P_2(\lambda)$ loses its stability, a Neimark–Sacker bifurcation (see, e.g., [12]) occurs, leading to an invariant curve $\Gamma = \Gamma_\lambda$ of f_λ. A detailed computer-assisted study of the curves Γ_λ and their breakdown has been made in [2].

In this paper we are mostly interested in the Lyapunov-type numbers, but we also briefly describe in the next section how we computed the approximations to Γ_λ. Figure 3.1 shows these curves for some λ-values between $\lambda = 2.001$ (near the Neimark–Sacker bifurcation) and $\lambda = 2.27$ (near breakdown). Our algorithm yields a cubic periodic C^2 spline through a discrete set of points representing Γ_λ. The spline is then used to obtain approximations to tangents and normals of Γ_λ. These are needed in (2.1) and (2.2) to obtain a_p and b_p. Approximations for the Lyapunov-type numbers, $\nu(p), \sigma(p), \bar{\nu}(p)$, and $\bar{\sigma}(p)$, can then be obtained from (2.5), (2.6), (2.7), and (2.8). See Figures 3.1–3.3.

Study of $\nu(p,n)$, etc., for $\lambda = 2.18$. We first fix $\lambda = 2.18$ and evaluate $\nu(p, n)$ and $\bar{\nu}(p, n)$ using (2.5) and (2.7) for increasing n. The point p is chosen as the point corresponding to $\theta = 0$ in our parameterization of Γ_λ, but any other point $p \in \Gamma_\lambda$ would serve the same purpose. The results for $\nu(p, n)$ and $\bar{\nu}(p, n)$ are shown in Figure 3.4. For $n \geq 300$ (about) the results are practically independent of n, giving approximations to $\nu(p)$ and $\bar{\nu}(p)$. It is clearly seen that $\nu(p) \neq \bar{\nu}(p)$, which implies phase-locking of the dynamics on Γ_λ.

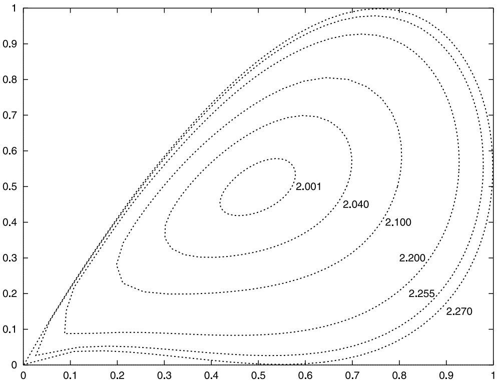


Fig. 3.1. *Invariant circles for the delayed logistic map for various values of* λ.

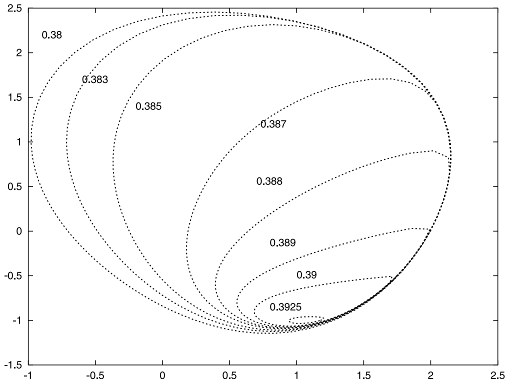


Fig. 3.2. *Invariant circles for the forced oscillator of van der Pol for various values of* λ.

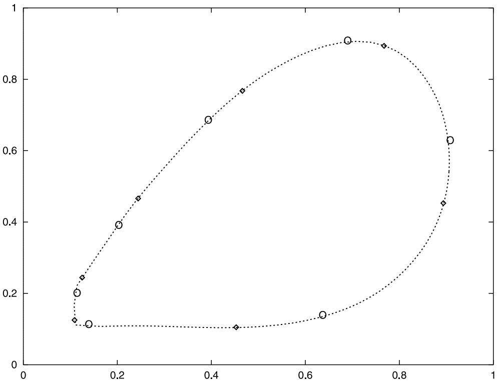


Fig. 3.3. *The invariant circle for* $\lambda = 2.18$ *with seven saddles denoted by* ($\diamond$) *and seven sinks denoted by* (o).

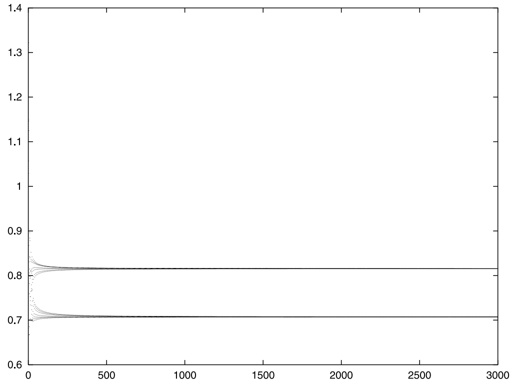


Fig. 3.4. *The values of* $\bar{\nu}(p,n)$ *and* $\nu(p,n)$ *for* $\lambda = 2.18$ *and* $0 \le n \le 3000$.

Study of $\boldsymbol{\nu(p,n)}$, etc., for $\boldsymbol{\lambda = 2.10}$. If $\lambda = 2.10$ then, according to Figure 3.4 of [2], the rotation number ρ of f_λ on Γ_λ is not a rational number with small denominator; i.e., either ρ is irrational or $\rho = m_1/m_2$ is rational with large denominator m_2. We have again computed $\nu(p,n)$ and $\bar{\nu}(p,n)$ for increasing n.

Here we see that there is no significant difference between the two numbers $\nu(p,n)$ and $\bar{\nu}(p,n)$ for large n.

The results for $\nu(p,n)$ and $\bar{\nu}(p,n)$ for $\lambda = 2.10$ are shown in Figures 3.5 and 3.6, respectively.

Study of the $\boldsymbol{\lambda}$-dependence of the Lyapunov-type numbers. If n is large enough, the numbers $\nu(p,n)$, etc., practically agree with their limits $\nu(p)$, etc. We found that the choice $n = 1{,}000$ was always sufficient for the map f_λ under consideration. Furthermore, the numbers $\nu(p)$, etc., are independent of p if we ignore the discontinuities at the sink orbit (for $\nu(p)$ and $\sigma(p)$) and the saddle orbit (for $\bar{\nu}(p)$ and $\bar{\sigma}(p)$) in a phase-locked situation. Thus, for every parameter value λ, we can compute

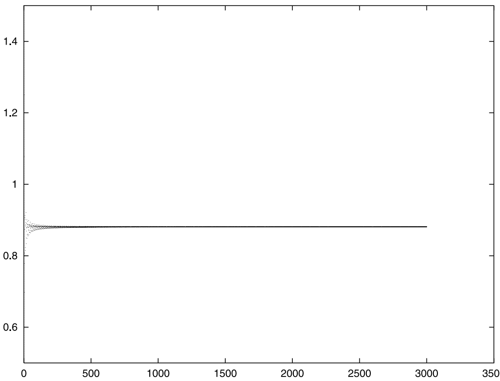


FIG. 3.5. *The values of $\nu(p,n)$ for $\lambda =$ 2.10 and $0 \leq n \leq 3000$.*

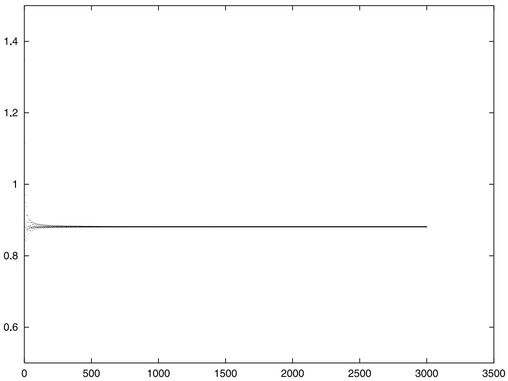


FIG. 3.6. *The values of $\bar{\nu}(p,n)$ for $\lambda =$ 2.10 and $0 \leq n \leq 3000$.*

four numbers,

$$\nu_\lambda, \quad \sigma_\lambda, \quad \bar{\nu}_\lambda, \quad \bar{\sigma}_\lambda. \tag{3.1}$$

Numerically, we have obtained these numbers by choosing $n = 1{,}000$ in the formulae for $\nu(p,n)$, etc., and by choosing p as the point corresponding to $\theta = 0$ in our parameterization of Γ_λ. (Any other point p on Γ_λ would serve the same purpose.)

Our numerical approximations of the functions (3.1) for $2 \leq \lambda \leq 2.27$ are shown in Figures 3.7, 3.9, 3.11, and 3.13.

If the dynamics of f_λ on Γ_λ is phase-locked, then we can compute the numbers (3.1) also by linearizing f_λ about the corresponding periodic orbits and using the results of section 2. This gives us a possibility of checking the accuracy of the computations. We have carried this out for the λ-interval with rotation number $\rho = 1/7$, i.e., for $\lambda_1 \leq \lambda \leq \lambda_4$, where

$$\lambda_1 \approx 2.1763, \quad \lambda_4 \approx 2.2006.$$

The results of the comparison are shown in Figures 3.8, 3.10, 3.12, and 3.14. Note that the solid lines correspond to the computations of the Lyapunov-type numbers based on eigenvalues; the dotted lines are based on the approximations $\nu(p,n)$, etc., for large n. The chosen λ-interval in the figures is somewhat larger than $[\lambda_1, \lambda_4]$, but the solid curves exist only for $\lambda_1 \leq \lambda \leq \lambda_4$. We will explain below that the numbers $\bar{\nu}_\lambda$ and $\bar{\sigma}_\lambda$ are, strictly speaking, not defined for λ in a certain subinterval of $[\lambda_1, \lambda_4]$.

The Lyapunov-type numbers in the phase-locked interval with $\rho = 1/7$. As shown in Figures 3.8, 3.10, 3.12, and 3.14, the behavior of the Lyapunov-type numbers as functions of λ is rather complicated. This is in agreement with the rather complex behavior of the eigenvalues of the linearizations about the periodic orbits, which we want to discuss in some detail.

For $\lambda_1 < \lambda < \lambda_4$ there are two seven-periodic orbits of f_λ on Γ_λ, which we denote by

$$\mathcal{O}^{(sa)} = \left\{q, f_\lambda q, f_\lambda^2 q, \ldots, f_\lambda^6 q\right\} \quad \text{and} \quad \mathcal{O}^{(si)} = \left\{r, f_\lambda r, f_\lambda^2 r, \ldots, f_\lambda^6 r\right\}$$

with $q = q_\lambda$ and $r = r_\lambda$. For $\lambda \to \lambda_1+$ and $\lambda \to \lambda_4-$ the two orbits collide, leading to a single nonhyperbolic orbit. The orbit $\mathcal{O}^{(sa)}$ consists of saddle fixed points of f_λ^7 and

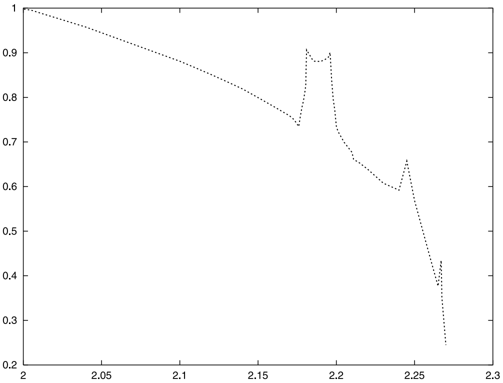


Fig. 3.7. *The values of $\bar{\nu}_\lambda$ for $2 \le \lambda \le 2.27$.*

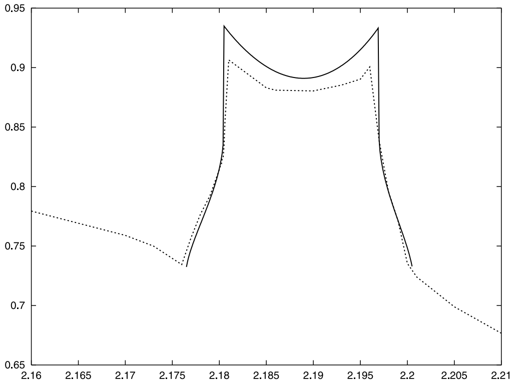


Fig. 3.8. *The values of $\bar{\nu}_\lambda$ obtained from eigenvalues for $\lambda_1 \le \lambda \le \lambda_4$ (solid line).*

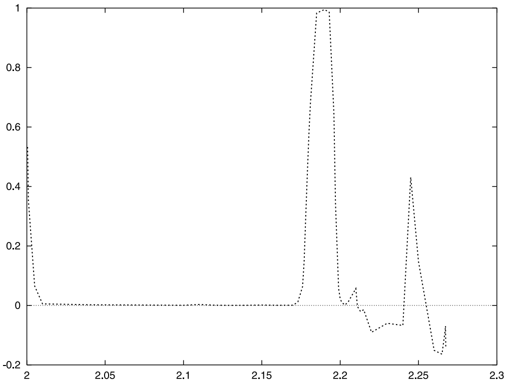


Fig. 3.9. *The values of $\bar{\sigma}_\lambda$ for $2 \le \lambda \le 2.27$.*

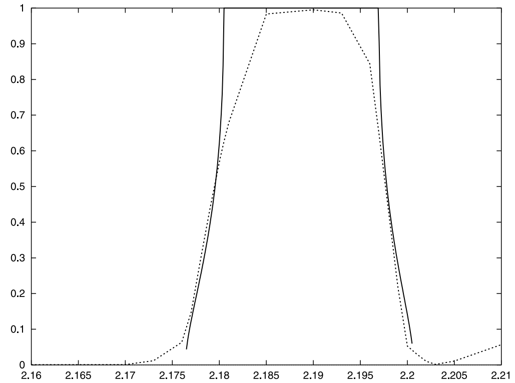


Fig. 3.10. *The values of $\bar{\sigma}_\lambda$ obtained from eigenvalues for $\lambda_1 \le \lambda \le \lambda_4$ (solid line).*

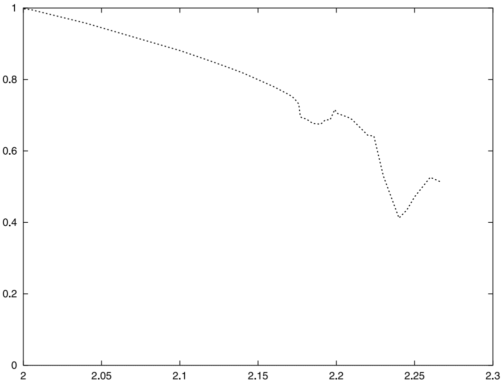


Fig. 3.11. *The values of ν_λ for $2 \le \lambda \le 2.27$.*

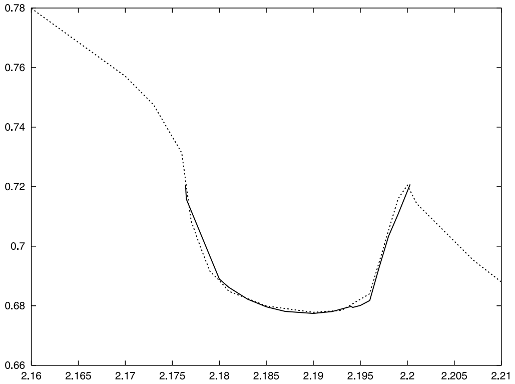


Fig. 3.12. *The values of ν_λ obtained from eigenvalues for $\lambda_1 \le \lambda \le \lambda_4$ (solid line).*

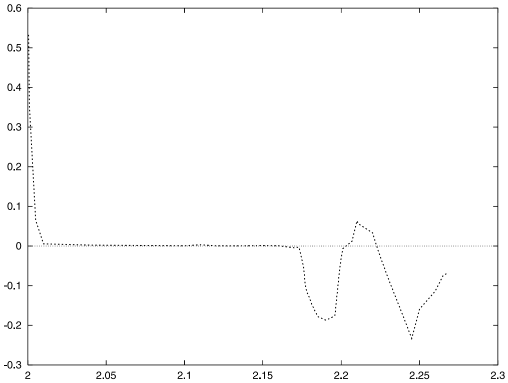


FIG. 3.13. *The values of σ_λ for $2 \leq \lambda \leq 2.27$.*

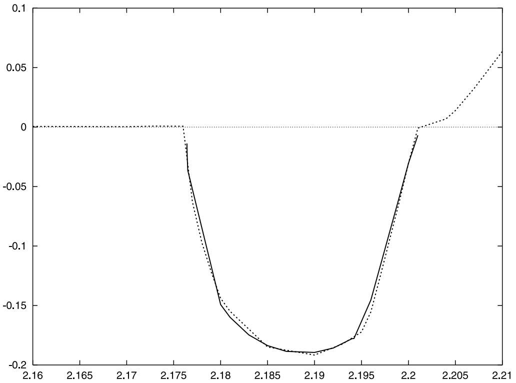


FIG. 3.14. *The values of σ_λ obtained from eigenvalues for $\lambda_1 \leq \lambda \leq \lambda_4$ (solid line).*

$\mathcal{O}^{(si)}$ consists of sinks. All points can be computed accurately by Newton's method applied to $f_\lambda^7(p) - p = 0$, together with continuation in λ. For $\lambda = 2.18$ the orbits are shown in Figure 3.3. Let

$$\Pi_\lambda^{(sa)} = \left(f_\lambda^7\right)'(q_\lambda) \quad \text{and} \quad \Pi_\lambda^{(si)} = \left(f_\lambda^7\right)'(r_\lambda)$$

denote the linearizations of f_λ about the orbits, and let $\mu_{1,2}^{(sa)}(\lambda)$ and $\mu_{1,2}^{(si)}(\lambda)$ denote the eigenvalues of these two matrices.

Behavior of $\bar{\nu}_\lambda$ and $\bar{\sigma}_\lambda$ for $\lambda_1 \leq \lambda \leq \lambda_4$. As stated in Theorems 2.2 and 2.4, the behavior of $\bar{\nu}_\lambda$ and $\bar{\sigma}_\lambda$ is determined by the eigenvalues at the sinks, $\mu_{1,2}^{(si)} = \mu_{1,2}^{(si)}(\lambda)$. However, this is only true as long as the sinks are *not* of spiral type. In the present example, the phase-locking interval $\lambda_1 \leq \lambda \leq \lambda_4$ contains a subinterval, $\lambda_2 < \lambda < \lambda_3$,

$$\lambda_2 \approx 2.181, \quad \lambda_3 \approx 2.196,$$

where the points of $\mathcal{O}^{(si)}$ are spiral points of f_λ^7, however. For the eigenvalues $\mu_{1,2}^{(si)}$ the following hold:

(1) For $\lambda_1 < \lambda < \lambda_2$ and for $\lambda_3 < \lambda < \lambda_4$ we have

$$0 < \mu_1^{(si)} < \mu_2^{(si)} < 1. \tag{3.2}$$

(2) At $\lambda = \lambda_2$ and at $\lambda = \lambda_3$ we have

$$0 < \mu_1^{(si)} = \mu_2^{(si)} < 1. \tag{3.3}$$

(3) For $\lambda_2 < \lambda < \lambda_3$ the eigenvalues $\mu_{1,2}^{(si)}$ form a complex conjugate pair with

$$0 < |\mu_1^{(si)}| = |\mu_2^{(si)}| < 1. \tag{3.4}$$

The inequalities (3.2) imply that

$$0 < \bar{\sigma}_\lambda < 1 \quad \text{and} \quad 0 < \bar{\nu}_\lambda < 1 \tag{3.5}$$

for $\lambda_1 < \lambda < \lambda_2$ and for $\lambda_3 < \lambda < \lambda_4$. Furthermore, we obtain from (3.3) that

$$\bar{\sigma}_\lambda \to 1 \quad \text{as} \quad \lambda \to \lambda_2 - \quad \text{or} \quad \lambda \to \lambda_3 + . \tag{3.6}$$

As $\lambda \to \lambda_1+$ or $\lambda \to \lambda_4-$, the orbits $\mathcal{O}^{(sa)}$ and $\mathcal{O}^{(si)}$ collide. At the saddle orbit $\mathcal{O}^{(sa)}$ we have

$$0 < \mu_1^{(sa)} < 1 < \mu_2^{(sa)} \quad \text{for} \quad \lambda_1 < \lambda < \lambda_4.$$

Together with (3.2) this implies that we have at collision

$$\mu_2^{(si)} = \mu_2^{(sa)} = 1 \quad \text{for} \quad \lambda = \lambda_1 \quad \text{and} \quad \lambda = \lambda_4. \tag{3.7}$$

Then $0 < \mu_1^{(si)} < 1 = \mu_2^{(si)}$ yields

$$\bar{\sigma}_\lambda = 0 \quad \text{for} \quad \lambda = \lambda_1 \quad \text{and} \quad \lambda = \lambda_4. \tag{3.8}$$

The behavior (3.6) and (3.8) for $\bar{\sigma}_\lambda$, as well as $0 < \bar{\sigma}_\lambda < 1$ for $\lambda_1 < \lambda < \lambda_2$ and for $\lambda_3 < \lambda < \lambda_4$, is clearly seen in Figure 3.10.

For $\lambda_2 < \lambda < \lambda_3$ the seven points of the sink orbit $\mathcal{O}^{(si)}$ are spiral fixed points of f_λ^7, which implies that Γ_λ winds infinitely often about each of the seven sinks. In particular, Γ_λ is not smooth at the sinks, but merely continuous. This breakdown of smoothness of Γ_λ is consistent with the general perturbation theory of invariant manifolds (see, e.g., [8]) in view of the limit behavior

$$\bar{\sigma}_\lambda \to 1 \quad \text{as} \quad \lambda \to \lambda_2 - \quad \text{or} \quad \lambda \to \lambda_3 + .$$

Since Γ_λ is not smooth for $\lambda_2 < \lambda < \lambda_3$, the Lyapunov-type numbers are, strictly speaking, not defined in this λ-interval. For $\nu_\lambda(p)$ and $\sigma_\lambda(p)$ the breakdown of smoothness at the *sinks* is irrelevant, however, because $\nu_\lambda(p)$ and $\sigma_\lambda(p)$ are determined by going along *backward* orbits. (This holds true unless $p \in \mathcal{O}^{(si)}$.) For $\bar{\nu}_\lambda$ and $\bar{\sigma}_\lambda$ it is natural to use the following definitions, in terms of eigenvalues,

$$\bar{\sigma}_\lambda = 1 \quad \text{and} \quad \bar{\nu}_\lambda = |\mu_{1,2}^{(si)}|. \tag{3.9}$$

(In case of an attracting spiral, the rates of attractivity toward and within Γ_λ are the same; thus $\bar{\sigma}_\lambda = 1$. The attractivity toward Γ_λ is governed by the absolute values of the eigenvalues; thus $\bar{\nu}_\lambda = |\mu_{1,2}^{(si)}|$.) The numerical curves (dotted lines) in Figures 3.8 and 3.10, which are based on (2.7) and (2.8), agree reasonably well with the solid curves, which are based on the eigenvalues.

Remark 3.1. The algorithm that we used to approximate Γ_λ is robust enough to ignore the difficulties caused by the presence of the small spirals. They are simply ignored. According to the computations in [19], the spiral diameters are less than 10^{-4}.

Behavior of $\boldsymbol{\nu_\lambda}$ and $\boldsymbol{\sigma_\lambda}$ for $\boldsymbol{\lambda_1 \le \lambda \le \lambda_4}$. According to Theorems 2.2 and 2.4, the numbers ν_λ and σ_λ are determined by the eigenvalues at the saddles, $\mu_{1,2}^{(sa)} = \mu_{1,2}^{(sa)}(\lambda)$. From the inequalities $0 < \mu_1^{(sa)} < 1 < \mu_2^{(sa)}$ it follows that

$$0 < \nu_\lambda = \left(\mu_1^{(sa)}\right)^{1/7} < 1 \quad \text{and} \quad \sigma_\lambda = \frac{\log\left(\mu_2^{(sa)}\right)}{\log\left(\mu_1^{(sa)}\right)} < 0.$$

As remarked above, we have

$$\mu_2^{(sa)} = 1 \quad \text{for} \quad \lambda = \lambda_1 \quad \text{and} \quad \lambda = \lambda_4,$$

which yields

$$\sigma_\lambda = 0 \quad \text{for} \quad \lambda = \lambda_1 \quad \text{and} \quad \lambda = \lambda_4.$$

This predicted behavior of ν_λ and σ_λ is clearly seen in Figures 3.12 and 3.14. We also see reasonable agreement between the results based on (2.5) and (2.6) (for large n) and the results based on the eigenvalues $\mu_{1,2}^{(sa)}(\lambda)$.

Remark on the accuracy of the computations. In Figures 3.8, 3.10, 3.12, and 3.14 we show two approximations to the functions $\bar{\nu}_\lambda, \bar{\sigma}_\lambda, \nu_\lambda$, and σ_λ. One approximation is obtained from eigenvalues, the other by evaluating the formulas for $\bar{\nu}(p,n)$, etc., for $n = 1{,}000$. The results of the two computations are in reasonable agreement, making it rather certain that we do not have major bugs in our codes. However, the agreement is far from perfect. We are rather sure that the discrepancies can be traced to inaccurate approximations of the invariant curves Γ_λ. (See the algorithm in the next section.) One can obtain better approximations of Γ_λ than we have used here by increasing the number of mesh points. However, the following difficulty arises: For $\lambda_2 < \lambda < \lambda_3$ the true invariant curves Γ_λ contain seven (infinite) spirals of small diameter ($< 10^{-4}$). If one chooses too many mesh points, these spirals are no longer ignored by the algorithm and the computations become extremely slow.

4. Invariant curves of planar maps. Our method of computing an invariant curve of a planar map is a discrete form of the Hadamard graph transformation. As an analytical tool, the Hadamard graph transform has been used in many works to establish perturbation results for invariant manifolds. We refer to Fenichel [8], for example. The constructive nature of the transformation suggests to apply it also in a numerical context.

For the convenience of the reader, we will describe the transformation—and a discrete version—here for the simple case of an invariant curve of a planar map. It should be noted, however, that the approach has much wider applicability.

We first describe the transformation in its analytical form. To this end, let $(\theta, r), 0 \le \theta < 2\pi, r \ge 0$, denote polar coordinates in the plane. Let g denote a map from the plane into itself whose representation in polar coordinates is given by

$$(\theta, r) \to \Big(g_{an}(\theta, r), g_{ra}(\theta, r)\Big),$$

that is, g_{an} and g_{ra} denote the angular and the radial part of the image under g. We assume that g has an invariant closed curve $\Gamma = g(\Gamma)$, which surrounds the origin and can be represented in the form

$$\Gamma : \quad (\theta, R^*(\theta)), \quad 0 \le \theta \le 2\pi.$$

Here R^* is a positive, 2π-periodic C^1 function.

We assume that we know an approximation $R^{old}(\theta)$ to the unknown function $R^*(\theta)$. Then, starting with the graph

$$\Gamma^{old} : \quad (\theta, R^{old}(\theta)), \quad 0 \le \theta \le 2\pi, \tag{4.1}$$

the Hadamard graph transform determines a new graph

$$\Gamma^{new}: \quad (\theta, R^{new}(\theta)), \quad 0 \le \theta \le 2\pi, \tag{4.2}$$

as follows.

Graph transform: For any $0 \le \theta < 2\pi$ *determine* $0 \le \alpha < 2\pi$ *with*

$$g_{an}(\alpha, R^{old}(\alpha)) = \theta \tag{4.3}$$

and then set

$$g_{ra}(\alpha, R^{old}(\alpha)) = R^{new}(\theta). \tag{4.4}$$

Remark. If one assumes that the Lyapunov-type numbers $\nu(p)$ and $\sigma(p)$ of the invariant curve Γ satisfy

$$\max_p \nu(p) < 1 \quad \text{and} \quad \max_p \sigma(p) < 1 \tag{4.5}$$

and if Γ^{old} is sufficiently close to Γ (in C^1), then the above equation (4.3) for α is uniquely solvable.

The transformation which transforms the graph (4.1) into (4.2) via (4.3) and (4.4) is known as Hadamard's graph transform. In our context, where the graphs are simply determined by functions $R(\theta)$, we may think of the transform as a map between functions,

$$R^{old} \to R^{new} = \mathcal{H}(R^{old}).$$

In an iterative fashion, the process can be repeated, leading to a sequence of functions,

$$\mathcal{H}^k(R^{old}), \quad k = 1, 2, 3 \dots .$$

Then, assuming (4.5) and closeness of Γ^{old} to Γ, a contraction argument can be applied to prove convergence,

$$\lim_{k\to\infty} \left| \mathcal{H}^k(R^{old}) - R^* \right|_\infty .$$

Here $|\cdot|_\infty$ denotes the maximum norm.

Numerical aspects. Let $0 = \theta_1 < \theta_2 < \cdots < \theta_N < 2\pi$ denote a finite grid,[4] and let

$$R_i^{old} = R_i^{old}(\theta_i), \quad i = 1, \dots, N,$$

denote a known grid function. We use these grid values to determine a 2π-periodic interpolant, denoted again by $R^{old}(\theta)$. In our code we have used periodic cubic splines for interpolation.

Discrete graph transform: For $i = 1, \dots, N$, *determine* $0 \le \alpha < 2\pi$ *with*

$$g_{an}(\alpha, R^{old}(\alpha)) = \theta_i \tag{4.6}$$

[4] In our applications to the delayed logistic map it was always sufficient to work with a uniform grid.

and then set

$$g_{ra}(\alpha, R^{old}(\alpha)) = R^{new}(\theta_i). \tag{4.7}$$

In our code, the scalar equation (4.6) for α is solved by bisection (to obtain a starting value) and Newton's method. Assuming convergence of the solutions for (4.6), the above process defines a transformation between grid functions,

$$(R_i^{old})_{1\leq i\leq N} \to (R_i^{new})_{1\leq i\leq N} = \mathcal{H}^{dis}((R_i^{old})).$$

Repeated application of the discrete transform $\mathcal{H}^{dis}$ leads to a (finite) sequence of grid functions,

$$\left(\mathcal{H}^{dis}\right)^k((R_i^{old})), \quad k = 1, 2, 3, \ldots .$$

We terminate the iteration when the estimate

$$\left|\left(\mathcal{H}^{dis}\right)^{k+1}((R_i^{old})) - \left(\mathcal{H}^{dis}\right)^k((R_i^{old}))\right|_2 < \text{tolerance}$$

is satisfied for a given tolerance. Here $|\cdot|_2$ is the discrete L_2-norm, which approximates $(\int_0^{2\pi}\psi^2(\theta)d\theta)^{1/2}$.

Application near a Neimark–Sacker bifurcation. Assume that $(x, y) \to f_\lambda(x, y)$ is a family of smooth maps of the plane, depending smoothly on the real parameter λ. An example is given by the maps $f_\lambda(x, y) = (y, \lambda y(1 - x))$ of section 3. Let $P_\lambda = (x_\lambda, y_\lambda)$ denote a branch of fixed points, $f_\lambda(P_\lambda) = P_\lambda$ and let $\mu_{1,2}(\lambda)$ denote the eigenvalues of the Jacobian $f'_\lambda(P_\lambda)$. It is well known that P_λ is an asymptotically stable fixed point if $\max_j |\mu_j(\lambda)| < 1$, whereas P_λ is unstable if $\max_j |\mu_j(\lambda)| > 1$.

Thus, a loss of stability of P_λ occurs at $\lambda = \lambda_0$ if the eigenvalues $\mu_{1,2}(\lambda)$ form a complex conjugate pair which leaves the unit circle when λ increases from $\lambda < \lambda_0$ to $\lambda > \lambda_0$. Then, under some additional assumptions, the map f_λ will have an attractive invariant curve Γ_λ for λ in some interval $\lambda_0 < \lambda < \lambda_1$. For details of this so-called Neimark–Sacker (or second Hopf or Poincaré–Andronov–Hopf) bifurcation we refer to [12], for example.

To apply our algorithm for approximating Γ_λ, we introduce polar coordinates in the (x, y)-plane which are centered at the bifurcation point $P_{\lambda_0} = (x_{\lambda_0}, y_{\lambda_0})$. In other words, we let

$$x = x_{\lambda_0} + r\cos\theta, \quad y = y_{\lambda_0} + r\sin\theta,$$

and then rewrite the map f_λ in terms of (θ, r)-coordinates obtaining $g_\lambda(\theta, r)$,

$$f_\lambda(x, y) \Longleftrightarrow g_\lambda(\theta, r).$$

The algorithm can then be applied to g_λ with λ as a continuation parameter. A computed invariant curve $\Gamma_{\lambda_{old}}$ provides starting values for approximating $\Gamma_\lambda, \lambda = \lambda_{old} + \Delta\lambda$. A first approximation to Γ_λ for λ near the bifurcation value λ_0 is provided by the Neimark–Sacker bifurcation theorem. Results for the delayed logistic map $f_\lambda(x, y) = (y, \lambda y(1 - x))$ are shown in Figure 3.1.

Acknowledgment. The authors wish to thank the referees for their careful reading of the first manuscript. Their comments led to an improved presentation of the paper.

REFERENCES

[1] V. I. Arnold, *Geometrical Methods in the Theory of Ordinary Differential Equations*, Springer-Verlag, New York, 1983.

[2] D. G. Aronson, M. A. Chory, G. R. Hall, and R. P. McGehee, *Bifurcation from an invariant circle for two parameter families of maps of the plane: A computer-assisted study*, Comm. Math. Phys., 83 (1982), pp. 303–354.

[3] V. Broer, H. M. Osinga, and G. Vegter, *Algorithms for computing normally hyperbolic invariant manifolds*, Z. Angew. Math. Phys., 48 (1997), pp. 480–524.

[4] A. Denjoy, *Sur les courbes definies par les equations differentielles a la surface du tore*, J. Math. Pures Appl. (9), 11 (1932), pp. 333–375.

[5] L. Dieci and J. Lorenz, *Computation of invariant tori by the method of characteristics*, SIAM J. Numer. Anal., 32 (1995), pp. 1436–1474.

[6] L. Dieci and J. Lorenz, *Lyapunov-type numbers and torus breakdown: Numerical aspects and a case study*, Numer. Algorithms, 14 (1997), pp. 79–102.

[7] K. D. Edoh, R. D. Russell, and W. Sun, *Computation of invariant tori by orthogonal collocation*, Appl. Numer. Math., 32 (2000), pp. 273–289.

[8] N. Fenichel, *Persistence and smoothness of invariant manifolds for flows*, Indiana Univ. Math. J., 21 (1971), pp. 193–226.

[9] J. Guckenheimer and P. Holmes, *Nonlinear Oscillations, Dynamical Systems and Bifurcation of Vector Fields*, Springer-Verlag, New York, 1983.

[10] J. Hadamard, *Sur l'itération et let solutions asymptotic des équations différentialles*, Proc. Soc. Math. France, 29 (1901), pp. 224–228.

[11] I. G. Kevrekidis, R. Aris, L. D. Schmidt, and S. Pelikan, *Numerical computation of invariant circles of maps*, Phys. D, 16 (1985), pp. 243–251.

[12] Y. Kuznetsov, *Elements of Applied Bifurcation Theory*, Springer-Verlag, New York, 1995.

[13] W. de Melo and S. van Strien, *One-Dimensional Dynamics*, Springer-Verlag, New York, 1993.

[14] G. Moore, *Computation and parameterization of invariant curves and tori*, SIAM J. Numer. Anal., 33 (1996), pp. 2333–2358.

[15] T. S. Parker and L. O. Chua, *Practical Numerical Algorithms of Chaotic Systems*, Springer-Verlag, New York, 1989.

[16] V. Reichelt, *Computing invariant tori and circles in dynamical systems*, in Numerical Methods for Bifurcation Problems and Large-Scale Dynamical Systems, IMA Vol. Math. Appl. 119, E. Doedel and L. Tuckerman, eds., Springer-Verlag, New York, 2000, pp. 407–437.

[17] V. Reichelt, *Berechnung invarianter Mannigfaltigkeiten in dynamischen Systemen*, Dissertation, RWTH Aachen, Aachen, Germany, 2000.

[18] K. Petersen, *Ergodic Theory*, Cambridge University Press, Cambridge, UK, 1983.

[19] W. Qin, *Invariant Curves for the Delayed Logistic Map*, manuscript, University of New Mexico, Albuquerque, NM, 1997.

[20] M. van Veldhuizen, *A new algorithm for the numerical approximation of an invariant curve*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 951–962.

[21] M. van Veldhuizen, *Convergence results for invariant curves algorithms*, Math. Comp., 51 (1987), pp. 677–697.

SIAM J. SCI. COMPUT.
Vol. 23, No. 4, pp. 1135–1160

A NUMERICAL STUDY OF FETI ALGORITHMS FOR MORTAR FINITE ELEMENT METHODS*

DAN STEFANICA†

Abstract. The finite element tearing and interconnecting (FETI) method is an iterative substructuring method using Lagrange multipliers to enforce the continuity of the finite element solution across the subdomain interface. Mortar finite elements are nonconforming finite elements that allow for a geometrically nonconforming decomposition of the computational domain into subregions and, at the same time, for the optimal coupling of different variational approximations in different subregions.

We present a numerical study of FETI algorithms for elliptic self-adjoint equations discretized by mortar finite elements. Several preconditioners which have been successful for the case of conforming finite elements are considered. We compare the performance of our algorithms when applied to classical mortar elements and to a new family of biorthogonal mortar elements, and we discuss the differences between enforcing mortar conditions instead of continuity conditions for the case of matching nodes across the interface. Our experiments are carried out for both two and three dimensional problems, and include a study of the relative costs of applying different preconditioners for mortar elements.

Key words. FETI algorithms, mortar finite elements, Lagrange multipliers, domain decomposition

AMS subject classifications. 65F10, 65N30, 65N55

PII. S1064827500378829

1. Introduction.

The finite element tearing and interconnecting (FETI) method is an iterative substructuring method using Lagrange multipliers which is actively used in industrial-size parallel codes for solving difficult computational mechanics problems. This method was introduced by Farhat and Roux [26]; a detailed presentation is given in [27], a monograph by the same authors. Originally used to solve second order, self-adjoint elliptic equations, it has later been extended to many other problems, e.g., time-dependent problems [18], plate bending problems [19, 24, 41], heterogeneous elasticity problems with composite materials [43, 44], acoustic scattering and Helmholtz problems [22, 23, 28, 29], linear elasticity with inexact solvers [32], and Maxwell's equations [42, 50]. Another Lagrange multiplier based method, the dual-primal FETI method, has recently been introduced by Farhat et al. [20] and Farhat, Lesoinne, and Pierson [21] for two dimensional problems, and a coordinate–free formulation of the FETI method has been analyzed by Brenner [11].

The FETI method is a nonoverlapping domain decomposition method and requires the partitioning of the computational domain Ω into nonoverlapping subdomains. It has been designed for conforming finite elements and makes use of Lagrange multipliers to enforce pointwise continuity across the interface of the partition. After eliminating the subdomain variables, the dual problem, given in terms of Lagrange

*Received by the editors October 2, 2000; accepted for publication (in revised form) May 30, 2001; published electronically November 7, 2001. Part of this work was carried out while the author was at the Courant Institute of Mathematical Sciences and with support in part by the National Science Foundation under grant NSF-CCR-9732208, and in part by the U.S. Department of Energy under contract DE-FG02-92ER25127.

http://www.siam.org/journals/sisc/23-4/37882.html

†Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA 02139 and Baruch College, City University of New York, 17 Lexington Avenue, New York, NY 10017 (dstefan@math.mit.edu, http://www-math.mit.edu/∼dstefan).

multipliers, is solved by a projected conjugate gradient (PCG) method. Once an accurate approximation for the Lagrange multipliers has been obtained, the values of the primal variables are obtained by solving a local problem for each subdomain.

It was shown experimentally in [25] that a certain projection operator used in the PCG solver plays a similar role to that of a coarse problem for other domain decomposition algorithms and that certain variants of the FETI algorithm are numerically scalable with respect to both the subproblem size and the number of subdomains. Mandel and Tezaur later showed that for a FETI method which employs a Dirichlet preconditioner, the condition number grows at most in proportion to $(1+\log(H/h))^2$ if the decomposition of Ω does not have crosspoints, i.e., the points that belong to the closure of more than two subdomains, and as $C(1+\log(H/h))^3$ in the general case; cf [40, 49]. Here, H is the subdomain diameter and h is the mesh size. Using a different preconditioner, Klawonn and Widlund obtained a FETI method which converges in fewer iterations than the classical FETI method. In [33], they proved an upper bound for the condition number of their method for elliptic problems with heterogeneous coefficients which is on the order of $(1+\log(H/h))^2$; see section 4.3 for more details.

Rixen and Farhat [43, 44] considered a Dirichlet preconditioner with a maximal number of pointwise continuity conditions at crosspoints, which results in a FETI algorithm with redundant Lagrange multipliers. It was shown in [33] that this algorithm is equivalent to using the preconditioner of Klawonn and Widlund and nonredundant multipliers for the FETI method.

In this paper, we study the numerical convergence properties of a family of FETI algorithms applied to mortar finite elements. Mortar finite elements are nonconforming finite element methods that allow for a geometrically nonconforming decomposition of the computational domain into subregions and, at the same time, for the optimal coupling of different variational approximations in different subregions. Here, optimality means that the global error is bounded by the sum of the local approximation errors on each subregion.

The importance of our study is related to the inherent advantages of mortar methods over the conforming finite elements. For example, the mesh generation is more flexible and can be made quite simple on individual subregions. This also makes it possible to move different parts of the mesh relative to each other, e.g., in a study of time-dependent problems. The same feature is most valuable in optimal design studies, where the relative position of parts of the model is not fixed a priori. The mortar methods also allow for local refinement of finite element models in only certain subregions of the computational domain, and they are also well suited for parallel computing; cf. [30].

We have used geometrically nonconforming mortar finite elements. Three FETI algorithms with different preconditioners for the dual problem have been considered: the Dirichlet preconditioner of Farhat and Roux [26], the block-diagonal preconditioner of Lacour [34], and the new preconditioner of Klawonn and Widlund [33]. These algorithms have been implemented for both the classical mortar finite elements of Bernardi, Maday, and Patera [8], and for the new biorthogonal mortar elements of Wohlmuth [52, 53], in two and three dimensions. Based on the results presented in this paper, we proposed and analyzed a new FETI algorithm with inexact solvers applied to three dimensional mortar finite elements in [47]. We note that a study of a FETI preconditioner for Maxwell's equations on nonmatching grids has been completed by Rapetti and Toselli [42].

Our results show that the Dirichlet preconditioner does not perform well in the mortar case, since convergence is achieved only after hundreds or thousands of iterations. However, the new preconditioner performs satisfactory; i.e., the number of iterations required to achieve convergence and the condition number of the algorithms depend only weakly on the number of nodes in each subregion and is independent of the number of subregions. For each of the three preconditioners, using the biorthogonal mortars results in algorithms which require less computational effort and fewer iterations than those using the classical mortar finite elements.

We have also studied the extra computational effort, due to the complexity of the mortar conditions, required for the implementation of the FETI algorithm with the new preconditioner. These costs might have been significant, in particular, in the three dimensional case. We conclude that the improvement of the iteration count was enough to offset this extra cost.

In the conforming finite element case, the meshes across the interface match. Therefore, continuity conditions, as well as mortar conditions, may be enforced across the interface. We have studied the differences between the FETI algorithms using both types of constraints in terms of iteration counts and computational costs. We conclude that the new preconditioner for either continuity conditions or for biorthogonal mortars results in the best algorithms.

The rest of the paper is structured as follows. In the next section, we describe the mortar finite element method. In section 3, we present the classical FETI method and the Dirichlet preconditioner, and in section 4, we discuss the FETI algorithm for mortars with two different preconditioners. In sections 5 and 6, we present numerical comparisons of the performances of three different FETI algorithms for mortar finite elements, and for two and three dimensional problems, respectively. In the last section, we discuss the differences between enforcing mortar conditions instead of continuity conditions for the case of matching nodes across the interface.

2. Mortar finite elements.

The mortar finite element methods were first introduced by Bernardi, Maday, and Patera in [8] for low order and spectral finite elements. A three dimensional version was developed by Ben Belgacem and Maday in [7] and was further analyzed for three dimensional spectral elements in [6]. Another family of biorthogonal mortar elements has recently been introduced by Wohlmuth [52, 53]. See also [45] for mortar *hp* finite elements and [4, 12, 31] for mortar $H(\mathrm{curl})$ elements. Cai, Dryja, and Sarkis [13] have extended the mortar methods to overlapping decompositions.

Several domain decomposition methods for mortar finite elements have been shown to perform similarly to the case of conforming finite elements; cf. [3, 15] for iterative substructuring methods, [16, 37, 38] for Neumann–Neumann algorithms, and [36, 48] for the FETI method. For other studies of preconditioners for the mortar method, see [14] for a hierarchical basis preconditioner and [1, 2] for iterative substructuring preconditioners. Multigrid methods have also been used to solve mortar problems; cf. [9, 10, 51].

2.1. Two dimensional low order mortar finite elements.

For simplicity, we restrict our presentation to mortar elements used in our numerical experiments, i.e., first order mortar finite elements on polygonal domains. The computational domain Ω is decomposed using a nonoverlapping polygonal partition $\{\Omega_i\}_{i=1:N}$. Let $\partial\Omega_D$ be the part of $\partial\Omega$ where Dirichlet conditions are imposed. If an edge of a polygon intersects $\partial\Omega_D$, we require that the entire edge belongs to $\partial\Omega_D$. The partition is said to be geometrically conforming if the intersection between the closure of any two subregions is either empty, a vertex, or an entire edge, and it is geometrically

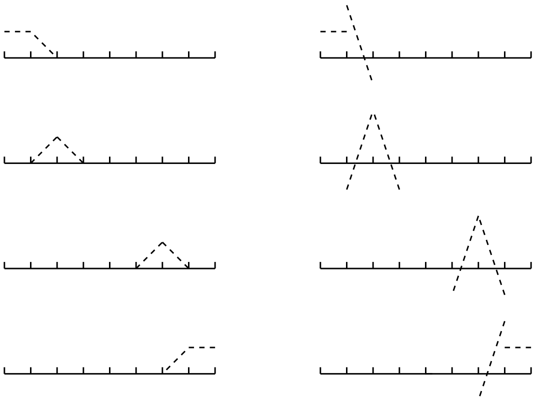


FIG. 1. *Test functions. Left: classical mortars; right: new mortars.*

nonconforming otherwise.

The interface between the subregions $\{\Omega_i\}_{i=1:N}$, denoted by Γ, is defined as the set of points that belong to the boundaries of at least two subregions.

The mortar finite element space V^h is defined as follows: Any mortar function $v \in V^h$ vanishes at all the nodes on $\partial\Omega_D$. The restriction of v to any Ω_i is a P_1 or a Q_1 finite element function. We do not require pointwise continuity across Γ. Instead, we choose a set of open edges of the subregions $\{\Omega_i\}_{i=1:N}$, called nonmortars, which form a disjoint partition of the interface. The edges of $\{\Omega_i\}_{i=1:N}$, which are part of Γ and were not chosen to be nonmortars, are called mortars. Across each nonmortar side γ, we impose weak continuity conditions for v. Let $\Gamma(\gamma)$ be the union of the parts of the mortars that coincides geometrically with $\overline{\gamma}$. Let v_γ and $v_{\Gamma(\gamma)}$ be the restriction of v to γ and $\Gamma(\gamma)$, respectively. The values of v on the nonmortar γ are then given by the mortar conditions

$$(1) \qquad \int_\gamma \left(v_\gamma - v_{\Gamma(\gamma)}\right) \ \psi \ ds \ = \ 0 \quad \forall \ \psi \in \Psi^h(\gamma),$$

where $\Psi^h(\gamma)$ is the space of test functions. Here, $\Psi^h(\gamma)$ is a subspace of codimension two of $V^h(\gamma)$, the restriction of V^h to γ. It consists of continuous, piecewise linear functions on γ that are constant in the first and last mesh intervals of $\overline{\gamma}$; cf. Figure 1.

We note that a nonmortar partition of the interface is always possible; cf. Stefanica [46]. The partition is not unique, but any choice can be treated the same from a theoretical point of view.

From (1), it results that the interior nodes of the nonmortar sides are not associated with genuine degrees of freedom in the finite element space V^h, while the values of v at the end points of $\overline{\gamma}$ are genuine degrees of freedom. To emphasize this aspect, we present here the matrix formulation of the mortar conditions, which will be further used in section 4.

Let $\bar{v}_\gamma$ be the vector of the interior nodal values of v on γ. For simplicity, we assume that the mesh is uniform on γ of mesh size h. Let $\bar{v}_{\Gamma(\gamma)}$ be the vector of the values of v at the end points of γ and at all the nodes on the edges opposite γ such that the intersection of γ and the supports of the corresponding nodal basis functions are not empty. Then $\bar{v}_\gamma$ is uniquely determined by $\bar{v}_{\Gamma(\gamma)}$; the matrix formulation of the mortar conditions (1) is

$$M_\gamma \bar{v}_\gamma - N_\gamma \bar{v}_{\Gamma(\gamma)} \;=\; 0, \tag{2}$$

or, solving for $\bar{v}_\gamma$, $\bar{v}_\gamma \;=\; P_\gamma \bar{v}_{\Gamma(\gamma)}$, with $P_\gamma = M_\gamma^{-1} N_\gamma$.

We note that N_γ is a banded matrix with a bandwidth of similar size for both the classical and the new mortars. For the classical mortar method, M_γ is a tridiagonal matrix and the mortar projection matrix P_γ is a full matrix. The projection of a nodal basis function from the mortar side results in a function with support equal to γ. The nodal values of this function decay exponentially to 0 at the end points of γ, away from the nodes on γ opposite the support of the nodal basis function from the mortar side.

Since $V^h(\Omega_i) \subset H^1(\Omega_i)$, we know that $v_\gamma \in H^{1/2}(\gamma)$. Thus, the test functions space $\Psi^h(\gamma)$ may be embedded in the dual space of $H^{1/2}(\gamma)$ with respect to the L^2 inner product, and therefore $\Psi^h(\gamma) \subset H^{-1/2}(\gamma)$. Based on this observation, a space of discontinuous piecewise linear test functions $\Psi^h_{new}(\gamma)$ for low order mortars has been developed by Wohlmuth [52]. The test function associated with the first interior node on γ is the constant 1 on the first mesh interval, decreases linearly from 2 to -1 on the second mesh interval, and vanishes everywhere else. A similar test function is introduced for the last interior node on γ. The test function for any other node on γ has the support on the two mesh intervals having the node as an end point; it increases linearly from -1 to 2 on the first interval and decreases from 2 to -1 on the second; cf. Figure 1.

The new mortar space has similar approximation properties to the classical mortar space; cf. [52]. A major advantage of the new mortar finite element space is that the mortar projection can be represented by a banded matrix, as opposed to the classical mortar finite element method, where the mortar projection matrix is, in general, a full matrix. More precisely, for the new mortar method, $M_\gamma = hI$ is a diagonal matrix and $P_\gamma = N_\gamma/h$ is banded. Therefore, the mortar projection of a nodal basis function on the mortar side vanishes outside the mesh intervals on the nonmortar which intersect its support.

2.2. The three dimensional case. For three dimensional problems, the mortars and nonmortars are open faces of the subregions which form the nonconforming decomposition of the computational domain Ω.

To introduce the mortar finite element space, we follow the outline from the previous section. Let $\{\Omega_i\}_{i=1:N}$ be a nonoverlapping polyhedral partition of Ω. If a face or an edge of a polyhedron intersects $\partial\Omega_D$ at an interior point, then the entire face or edge is assumed to belong to $\partial\Omega_D$. The partition is said to be geometrically conforming if the intersection between the closures of any two subregions is either empty, a vertex, an entire edge, or an entire face, and it is nonconforming otherwise.

The nonmortars $\{\mathcal{F}_l\}_{l=1}^{L}$ are faces of the subregions which form a disjoint partition of the interface Γ. The faces of $\{\Omega_i\}_{i=1:N}$ that are part of Γ and were not chosen to be nonmortars are called mortars.

We now describe the test functions associated with an arbitrary nonmortar face $\mathcal{F}$. Let $\Gamma(\mathcal{F})$ be the union of parts of mortar faces opposite $\mathcal{F}$. The test function space $\Psi^h(\mathcal{F})$ is a subset of $V^h(\mathcal{F})$, the restriction of V^h to $\mathcal{F}$, such that the value of a test function at a node on $\partial\mathcal{F}_l$ is a convex combination of its values at the neighboring interior nodes of $\mathcal{F}_l$. If $V^h(\mathcal{F})$ is a P_1 or a Q_1 space, then the dimension of $\Psi^h(\mathcal{F})$ is equal to the number of interior nodes of $\mathcal{F}$.

The mortar finite element space V^h consists of functions v which vanish at all the nodal points of $\partial\Omega_D$. Its restriction to any Ω_i is a P_1 or a Q_1 finite element function. The values of a mortar function $v \in V^h$ on any nonmortar face $\mathcal{F}$ are given by the mortar conditions

$$\int_{\mathcal{F}} \left(v_{\mathcal{F}} - v_{\Gamma(\mathcal{F})}\right) \ \psi \ ds \ = \ 0 \quad \forall \ \psi \in \Psi^h(\mathcal{F}).$$

A version of the new mortars for the three dimensional case, based on biorthogonal test functions such as those described in section 2.1, has been developed by Wohlmuth. For details, we refer the reader to [53].

3. The classical FETI algorithm.

In this section, we review the original FETI method of Farhat and Roux [27] for elliptic problems discretized by *conforming finite elements*. To simplify our presentation, we discuss only the Poisson equation with mixed Neumann–Dirichlet boundary conditions. The extension of the algorithm to the case of other self-adjoint elliptic equations is straightforward.

Let $f \in L^2(\Omega)$. We look for a solution $u \in H^1(\Omega)$ of the mixed boundary value problem

$$\left\{ \begin{array}{rcll} -\Delta u & = & f & \text{on} \quad \Omega, \\ u & = & 0 & \text{on} \quad \partial\Omega_D, \\ \frac{\partial u}{\partial n} & = & 0 & \text{on} \quad \partial\Omega_N, \end{array} \right. \tag{3}$$

where $\partial\Omega = \partial\Omega_N \cup \partial\Omega_D$. For unique solvability, we require that $\partial\Omega_D$ has positive Lebesgue measure.

On Ω, we consider P_1 or Q_1 finite elements with mesh size h. The finite element mesh is partitioned along mesh lines into N nonoverlapping subdomains $\Omega_i \subset \Omega$, $i = 1 : N$. Since the finite element mesh is conforming, the boundary nodes of the subdomains match across the interface. A subdomain Ω_i is said to be floating if $\partial\Omega_i \cap \partial\Omega_D = \emptyset$, and nonfloating otherwise.

As in other substructuring methods, the first step of the FETI method consists of eliminating the interior subdomain variables, which results in a Schur complement formulation of our problem. Let $S^{(i)}$ be the Schur complement matrix of Ω_i, and let f_i be the contribution of Ω_i to the load vectors. Let $S = \text{diag}_{i=1}^{N} S^{(i)}$ be a block-diagonal matrix, and let f be the vector $[f_1, \ldots, f_N]$. We denote by u_i the vector of nodal values on $\partial\Omega_i$ and by u the vector $[u_1, \ldots, u_N]$.

If Ω_i is a floating subdomain, then $S^{(i)}$ is a singular matrix, and its kernel is generated by a vector Z_i with entries corresponding to the nodes of $\partial\Omega_i$ equal to 1 and entries corresponding to all the other nodes equal to 0. Let Z consist of all the column vectors Z_i. Then $\text{Ker}S = \text{Range}Z$.

Let B be the matrix of constraints which measures the jump of a given vector u across the interface; B will also be referred to as the Lagrange multiplier matrix. Each row of the matrix B is associated with two matching nodes across the interface, and has values 1 and -1, respectively, at the two nodes, and zero entries everywhere else. A finite element function with corresponding vector values u is continuous if and only if $Bu = 0$.

For a method without redundant constraints and multipliers, the number of pointwise continuity conditions required at crosspoints, i.e., the points that belong to the closure of more than two subdomains, and, therefore, the number of corresponding rows in the matrix B, is one less then the number of the subdomains meeting at the crosspoint. There exist several different ways of choosing which conditions to enforce at a crosspoint, all of them resulting in algorithms with similar properties.

An alternative suggested in [43, 44] is to connect all the degrees of freedom at the crosspoints by Lagrange multipliers and use a special scaling, resulting in a method with redundant multipliers; see section 4.3 for further details.

Let W_i be the space of the degrees of freedom associated with $\partial\Omega_i \setminus \partial\Omega_D$, and let W be the direct sum of all spaces W_i. If $U = \text{Range}B$ is the space of the Lagrange multipliers, then $S : W \to W$ and $B : W \to U$. Let B^t be the transpose of B. By introducing Lagrange multipliers λ for the constraint $Bu = 0$, we obtain a saddle point Schur formulation of (3),

$$\left\{ \begin{array}{lclcl} Su & + & B^t\lambda & = & f, \\ Bu & & & = & 0. \end{array} \right. \tag{4}$$

3.1. Algebraic formulation. In the FETI method, the primal variable u is eliminated from (4), and the resulting equation for the dual variable λ is solved by a PCG method.

We note that S is singular if there exist at least one floating subdomain among the subdomains Ω_i, $i = 1 : N$. Let $S^\dagger$ be the pseudoinverse of S; i.e., for any $b \perp \text{Ker}S$, $S^\dagger b$ is the unique solution of $Sx = b$ such that $S^\dagger b \in \text{Range}S$. The first equation in (4) is solvable if and only if

$$f - B^t\lambda \ \perp \ \text{Ker}S. \tag{5}$$

If (5) is satisfied, then

$$u \ = \ S^\dagger(f - B^t\lambda) + Z\alpha, \tag{6}$$

where $Z\alpha$ is an element of $\text{Ker}S = \text{Range}Z$ to be determined.

Let $G = BZ$. Substituting (6) into the second equation in (4), it follows that

$$BS^\dagger B^t\lambda \ = \ BS^\dagger f + G\alpha. \tag{7}$$

An important role in the FETI algorithm is played by V, a subset of U defined by $V = \text{Ker}G^t$. In other words, $V \ = \ \text{Ker}G^t \ \perp \ \text{Range}G \ = \ B\text{Range}Z \ = \ B\text{Ker}S$. Let $P = I - G(G^tG)^{-1}G^t$ be the orthogonal projection onto V. It is easy to see that G^tG is nonsingular, by using the fact that $\text{Ker}B \cap \text{Range}Z = \text{Ker}B \cap \text{Ker}S = \emptyset$. Since $P(G\alpha) = 0$, if P is applied to (7), it results that

$$PBS^\dagger B^t\lambda \ = \ PBS^\dagger f. \tag{8}$$

We now return to the necessary condition (5). Since $\text{Ker}S \ = \ \text{Range}Z$, we obtain that (5) is equivalent to $f - B^t\lambda \perp \text{Range}Z$, which leads to $Z^t(f - B^t\lambda) \ = \ 0$ and therefore to

$$G^t\lambda \ = \ Z^tf. \tag{9}$$

Let $F = BS^\dagger B^t$, $d = BS^\dagger f$, and $e = Z^tf$. We concluded that we have to solve the dual problem (8) for λ subject to the constraint (9); with the new notations,

$$PF\lambda = Pd, \tag{10}$$

$$G^t\lambda = e. \tag{11}$$

We note that, from (7), it follows that $\alpha = (G^tG)^{-1}G^t(F\lambda - d)$. Therefore, after an approximate solution for λ is found, the primal variable u is obtained from (6) by solving a Neumann or a mixed boundary problem on each floating and nonfloating subdomain, respectively, corresponding to a vector multiplication by $S^{\dagger}$.

The main part of the FETI algorithm consists of solving (10) for the dual variable λ, which is done by a PCG method. Since λ must also satisfy the constraint (11), let $\lambda_0 = G(G^tG)^{-1}e$ be the initial approximation. Then $G^t\lambda_0 = e$ and $\lambda - \lambda_0 \in \mathrm{Ker}G^t = V$. If all the increments $\lambda_k - \lambda_{k-1}$, i.e., the search directions, are in V, then (11) will be satisfied.

One possible preconditioner for (10) is of the form PM, where $M = BSB^t$. When a vector multiplication by M is performed, N independent Dirichlet problems have to be solved in each iteration step. Therefore, M is known as the Dirichlet preconditioner. We note that the Schur complement matrix S is never computed explicitly, since only the action of S on a vector is needed.

Mandel and Tezaur [40] have shown that the condition number of this FETI method is bounded from above by $C(1 + \log(H/h))^3$, where C is a positive constant independent of h, H. If there are no crosspoints in the partition of Ω, then this polylogarithmic bound improves to $C(1 + \log(H/h))^2$.

We conclude this section with a few comments on the PCG algorithm:

Projected Preconditioned Conjugate Gradient Iteration (PCG)

```
$\lambda_0 = G(G^tG)^{-1}e$, $r_0 = Pd - PF\lambda_0$, $n = 1$
while $(Mr_{n-1}, r_{n-1}) \geq tol$
  $w_{n-1} = Pr_{n-1}$
  $z_{n-1} = Mw_{n-1}$
  $y_{n-1} = Pz_{n-1}$
  $\beta_n = (y_{n-1}, r_{n-1})/(y_{n-2}, r_{n-2})$        $(\beta_1 = 0)$
  $p_n = y_{n-1} + \beta_n p_{n-1}$                        $(p_1 = y_0)$
  $\alpha_n = (y_{n-1}, r_{n-1})/(Fp_n, p_n)$
  $\lambda_n = \lambda_{n-1} + \alpha_n p_n$
  $r_n = r_{n-1} - \alpha_n PFp_n$
  $n = n + 1$
end
```

In each iteration step of the PCG algorithm, the residual and the search directions are projected onto the space V, i.e., $w_{n-1} = Pr_{n-1}$ and $y_{n-1} = Pz_{n-1}$. This projection step plays the role of a coarse problem which is solved in each iteration and is the reason why the FETI method is numerically scalable, even though it lacks an explicit coarse space construction. We note that $r_{n-1} \in V$ at every step. Therefore, it follows that $w_{n-1} = r_{n-1}$, and thus only one projection onto V is required per iteration step. This observation is particularly important for some of the algorithms suggested in [33].

4. The FETI algorithm for mortars. As we have seen in section 3, in the classical FETI algorithm the computational domain Ω is partitioned into nonoverlapping subregions, multiple degrees of freedom are introduced for the matching nodes across the interface, and pointwise continuity across the interface is enforced by a Lagrange multiplier matrix B. This methodology is very similar to that used in [5], where a saddle point formulation for the mortar finite element method has been introduced.

In fact, the FETI method can be applied without any algorithmic changes for a mortar finite element discretization of Ω, using the nonoverlapping partition $\{\Omega_i\}_{i=1:N}$ considered in section 2. We recall that this partition may be geometrically nonconforming and the nodes across the interface do not necessarily match. To keep the presentation clear, we assume that each subregion Ω_i has a diameter of order H and that its triangulation has a mesh size of order h. The matrix S is again a block-diagonal matrix $\mathrm{diag}_{i=1}^{N} S^{(i)}$, where the local Schur complement matrices $S^{(i)}$ are obtained from the finite element discretizations on individual subregions. We have to solve the problem

$$\left\{ \begin{array}{lclcl} Su & + & B^t\lambda & = & f, \\ Bu & & & = & 0, \end{array} \right.$$

where the matrix B enforces mortar conditions across the interface. The dual problem is obtained as in section 3.1. It results in solving

$$PF\lambda \;=\; Pd, \tag{12}$$

with a PCG method, with initial approximation $\lambda_0 = G(G^tG)^{-1}e$, and with all the search directions in V.

The price we pay for the inherent flexibility of the mortar finite elements is due to the fact that the matrix B is more complicated in the mortar case, compared to that of the classical FETI method with conforming finite elements. The matrix B has one block, B_γ, for each nonmortar side γ. We adopt the matrix formulation of the mortar conditions from section 2.1. Let M_γ and N_γ be the matrices which multiply the nonmortar and mortar nodal values in the mortar conditions across γ, respectively. Then B_γ consists of the columns of M_γ and $-N_\gamma$ for the nodes of γ and those on the mortars opposite γ, and has zero columns corresponding to all the other nodes.

We note that the mortar conditions are all associated with the interior nodes on the nonmortar sides. Therefore, the problem of choosing the crosspoints constraints does not arise in the mortar case.

In our numerical experiments, we have implemented three different preconditioners suggested in the FETI literature for the dual problem (12). In sections 4.1–4.3, we present each of them briefly.

4.1. The Dirichlet preconditioner. In [26], Farhat and Roux introduced the Dirichlet preconditioner for the FETI method

$$PM = PBSB^t. \tag{13}$$

This preconditioner was shown to perform well for conforming finite elements; see, e.g., [26] for numerical results and [40] for condition number estimates.

4.2. A block-diagonal preconditioner. In [34, 35], Lacour suggested another preconditioner designed specifically for a mortar version of the FETI algorithm and without a counterpart in the conforming case. Let $\mathrm{diag} B_\gamma B_\gamma^t$ be the block-diagonal matrix which has a block $B_\gamma B_\gamma^t$ of size equal the number of interior nodes on γ for each nonmortar γ. We note that $\mathrm{diag} B_\gamma B_\gamma^t$ is the block-diagonal part of the matrix BB^t. In the three dimensional case, each block corresponds to a nonmortar face $\mathcal{F}$, and the block-diagonal matrix is $\mathrm{diag} B_\mathcal{F} B_\mathcal{F}^t$. To simplify our presentation, we will use the same notation, $\mathrm{diag} B_\gamma B_\gamma^t$, for the three dimensional block-diagonal matrix.

The preconditioner $P\overline{M}$ is defined as follows:

$$P\overline{M} \;=\; P(\mathrm{diag} B_\gamma B_\gamma^t)^{-1} BSB^t (\mathrm{diag} B_\gamma B_\gamma^t)^{-1}. \tag{14}$$

4.3. A new preconditioner. In [33], Klawonn and Widlund studied a FETI method for elliptic problems with heterogeneous coefficients, discretized by conforming finite elements, and designed a new preconditioner for these types of problems. They used this preconditioner to show the connection between FETI methods and Neumann–Neumann methods, in particular the balancing method [39].

In the case of no coefficient jump, as in our Poisson problem, the new preconditioner has the form

$$P\widehat{M} \;=\; P(BB^t)^{-1}BSB^t(BB^t)^{-1}. \tag{15}$$

Klawonn and Widlund established the following upper bound for the condition number of their FETI algorithms, which is valid for all cases, including when the partition contains crosspoints

$$\kappa(P\widehat{M}PF) \;\leq\; C\left(1+\log(H/h)\right)^2.$$

In the same paper, it is proven that the preconditioner $\widehat{M}$ with a minimal number of pointwise continuity conditions at the crosspoints, and therefore of Lagrange multipliers, results in a similar algorithm as the FETI method with redundant Lagrange multipliers of Rixen and Farhat [43, 44]. Since the Lagrange multipliers in the mortar case are not associated with the vertices of the subregions, a FETI algorithm with redundant multipliers cannot be implemented for mortars.

To use the new preconditioner of Klawonn and Widlund for the FETI method with mortars, we must show that the matrix BB^t is nonsingular in the mortar case. The number of columns of B is equal to the number of nodes from W, while the number of rows of B is equal to the number of Lagrange multipliers. Since each Lagrange multiplier is associated with an interior node on a nonmortar side, it results that B has fewer rows than columns. Therefore, if we show that the rank of B is equal to its number of rows, we may conclude that BB^t is nonsingular. We consider the minor of B consisting of the columns corresponding to the interior nodes of the nonmortars. The resulting block-diagonal square matrix diagM_γ is nonsingular, since each block M_γ is a diagonally dominant matrix for the classical mortar elements, and the identity matrix for the new mortar elements.

We conclude this section with comments on the difficulties of extending the convergence analysis of the FETI method from conforming finite elements to mortar elements.

The numerical evidence presented in sections 5.1 and 6.1 strongly suggested that, in the mortar case, the new preconditioner performed as well as it did in the conforming finite element case. Extending the analysis of Klawonn and Widlund to the mortar case would require establishing the bound

$$|B^t(BB^t)^{-1}Bw|_S^2 \;\leq\; C\left(1+\log(H/h)\right)^2|w|_S^2 \quad \forall\, w\in W, \quad Bw\in V. \tag{16}$$

In the conforming finite element case, the matrix BB^t is equal to twice the identity matrix, except for the entries corresponding to crosspoint conditions, where it has a block-diagonal structure. The inverse matrix $(BB^t)^{-1}$ has the same block-diagonal structure, and (16) can be obtained by using Sobolev-type inequalities for finite element functions (also known as cutoff estimates).

In the mortar finite element case, the matrix BB^t is not close to a multiple of the identity and is no longer block diagonal. There are two types of entries outside the diagonal blocks. Some correspond to Lagrange multipliers associated with the first and last interior points of the nonmortars. Others occur because there exist nodal basis functions associated with points on the mortar sides, the support of which intersects more than one nonmortar; see Figure 4 for the sparsity pattern of BB^t.

An important consequence of such structure of BB^t is that the matrix $(BB^t)^{-1}$ is a full matrix in the mortar case. Therefore, cutoff estimates can no longer be used to prove (16). An alternative solution would require new tools in order to reduce the inequality (16) to the two subdomain case. This case would then be solved by deriving some specific stability estimates for the mortar projection.

5. Numerical results for two dimensional problems.

In this section, we present numerical results for the FETI method for a mortar finite element discretization of a two dimensional problem. We have tested each of the three preconditioners of sections 4.1–4.3 on nonconforming discretizations of the computational domain.

Our interests were three fold:

- to compare the convergence performances of the different FETI preconditioners for mortar methods, based on iteration counts and estimates for the condition numbers;
- to apply the FETI algorithms for the new mortar finite elements and compare the iteration counts and the flop counts to those obtained for the classical mortar finite elements;
- to analyze the extra computational effort, due to the complexity of the mortar conditions, required for the implementation of the FETI algorithm with the new preconditioner.

As the model problem in two dimensions, we chose the Poisson equation on the unit square $\Omega = [0,1]^2$ with zero Dirichlet boundary conditions. The right-hand side function f in (3) was selected such that the exact solution of the problem is known.

The computational domain Ω was partitioned into 16, 32, 64, and 128 geometrically nonconforming rectangular subregions, respectively; see Figure 2. On each subregion, we considered Q_1 elements of mesh size h, and, to make the comparisons easier, all the subregions had diameters of the same order, H. For each partition, the number of nodes on each edge, H/h, was taken to be, *on average*, 4, 8, 16, and 32, respectively, for different sets of experiments. Across the partition interface Γ the meshes did not necessarily match. A saddle point formulation of the problem was used, and mortar conditions were enforced across Γ.

We report the iteration counts and the flop counts of the algorithms. We also report condition number estimates for the algorithms, obtained directly from the conjugate gradient iteration. The PCG iteration was stopped when the residual norm had decreased by a factor of 10^{-6}. All the experiments were carried out in MATLAB.

We now present some implementation details. We did not compute the Schur complements explicitly, nor their pseudoinverses, but only the stiffness matrices for each subdomain. To multiply a vector by a Schur complement matrix, we solved, in each subregion, a Poisson problem with Dirichlet boundary conditions. To multiply a vector by $S^\dagger$, we solved a Poisson problem with mixed boundary conditions in each nonfloating subregion and with Neumann boundary conditions in each floating subregion; see, e.g., [17]. We stored only the interior-boundary and boundary-boundary blocks of the local stiffness matrix and the Cholesky factor of the interior-interior block, which is symmetric and positive definite. To have a uniquely solvable problem on the floating subregions, we required that the solution of the local Neumann problem be orthogonal to Ker$S$, i.e., to the constant functions on the subregion. A simple way of enforcing this orthogonality condition was by adding a Lagrange multiplier

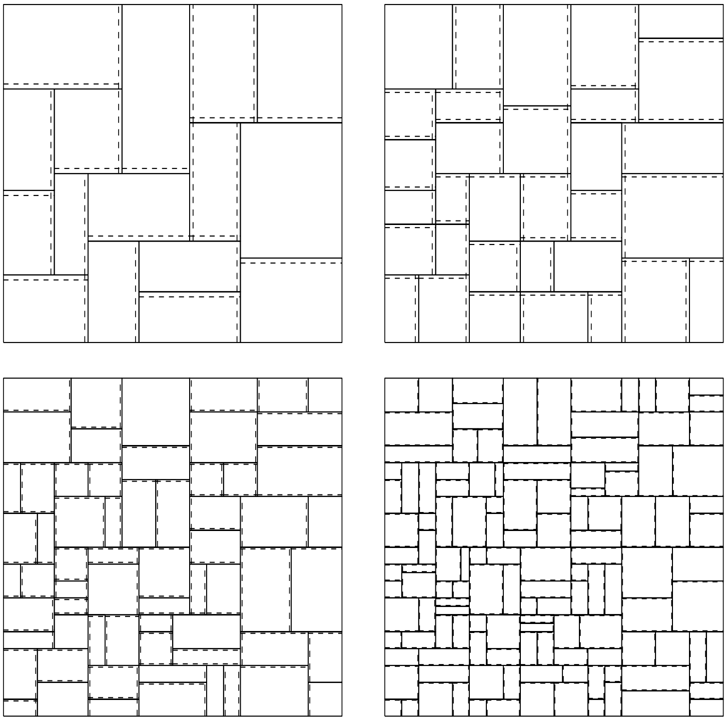


FIG. 2. *Geometrically nonconforming partitions of* Ω. *Upper left:* 16 *subdomains; upper right:* 32 *subdomains; lower left:* 64 *subdomains; lower right:* 128 *subdomains.*

and storing the LU components of the extended stiffness matrix.

5.1. Convergence properties of the FETI algorithms. We now turn to the main part of this section, a discussion of the performance of the FETI algorithms for mortar finite elements with the new preconditioner $\widehat{M}$, (15), the preconditioner $\overline{M}$, (14), and the Dirichlet preconditioner M, (13). In Table 1 we report the iteration count, the condition number approximation, and the flop count for the aforementioned preconditioners.

The FETI algorithm with the Dirichlet preconditioner M required hundreds of iterations to converge, and the computational costs were one to two orders of magnitude bigger than for the other preconditioners. The iteration count grew faster than polylogarithmically as a function of the number of nodes on each subdomain edge, H/h, and appeared to grow linearly with the number of subdomains. The Dirichlet preconditioner is therefore noncompetitive, since many domain decomposition methods have convergence rates independent of the number of subdomains. The condition numbers estimates were on the order of 10^4–10^6, unusually large for these types of algorithms. Moreover, our estimates are likely to be smaller than the actual condition numbers, since there was no convincing convergence pattern for the condition number approximation obtained in the iteration; see section 6.1 and Figure 3 therein for more details. Thus, unlike in the case of FETI algorithms with conforming finite elements, the Dirichlet preconditioner M did not yield a numerically scalable method for mortar finite element methods.

The new preconditioner $\widehat{M}$ scaled similarly to M in the conforming case. When

Table 1
Convergence results; two dimensional geometrically nonconforming partition; classical mortar elements.

		New precond.			Block-diag. precond.			Dirichlet precond.		
N	H/h	Iter	Cond	Mflops	Iter	Cond	Mflops	Iter	Cond	Mflops
16	4	10	4.14	6.9e–1	21	26.95	1.4e+0	111	7.3e+3	7.2e+0
16	8	12	5.14	8.2e+0	21	29.86	1.4e+1	240	4.2e+4	1.6e+2
16	16	13	6.44	1.5e+2	23	36.53	2.6e+2	320	6.5e+4	3.7e+3
16	32	14	7.35	3.4e+3	23	38.03	5.6e+3	348	6.8e+4	7.4e+4
32	4	11	6.53	1.9e+0	23	34.51	3.7e+0	223	1.2e+4	3.5e+1
32	8	13	7.58	1.9e+1	24	45.96	3.5e+1	455	7.1e+4	6.6e+2
32	16	14	8.86	3.5e+2	26	61.97	6.5e+2	528	1.0e+5	1.3e+4
32	32	16	9.79	9.1e+3	27	65.39	1.5e+4	569	1.2e+5	3.2e+5
64	4	14	7.23	6.1e+0	32	47.99	1.3e+1	578	9.1e+4	2.2e+2
64	8	16	8.76	5.7e+1	35	72.62	1.2e+2	1012	7.5e+5	3.5e+3
64	16	18	10.68	1.0e+3	36	91.43	2.0e+3	1266	1.2e+6	7.1e+4
64	32	20	12.40	2.5e+4	39	94.47	4.8e+4	1324	1.4e+6	1.5e+6
128	4	14	7.60	1.3e+1	36	64.53	3.0e+1	1144	9.2e+5	9.2e+2
128	8	17	9.56	1.3e+2	40	82.09	2.9e+2	1350	6.8e+5	1.0e+4
128	16	19	11.73	2.3e+3	41	96.60	4.8e+3	1436	9.9e+6	1.7e+5
128	32	21	13.03	5.7e+4	41	99.82	1.1e+5	–	–	–

the number of nodes on each subdomain edge, H/h, was fixed and the number of subdomains, N, was increased, the iteration count showed only a slight growth. When H/h was increased, while the partition was kept unchanged, the increase in the number of iterations was quite satisfactory and very similar to that of the conforming case. The condition number estimates exhibited a similar dependence on the number of subdomains and on the number of nodes on each subdomain edge.

The block-diagonal preconditioner $\overline{M}$ had good convergence properties as well. The iteration counts showed just a small increase when the number of nodes on each subdomain edge was increased, while the partition was kept unchanged. There seemed to be a stronger than desired dependence of the iteration counts on the number of subdomains, which was less than optimal. The condition number estimates followed a similar pattern but were significantly larger than the condition number estimates for the new preconditioner $\widehat{M}$.

Overall, the block-diagonal preconditioner $\overline{M}$ required about twice as many iterations to convergence and twice as much computational effort as $\widehat{M}$; cf. Table 1. Therefore, even though multiplying $\overline{M}$ by a vector required less computational effort than when $\widehat{M}$ was used, the increase in the iteration count resulted in flop counts which were twice as large. This suggests that dropping the nonzero diagonal terms of BB^t relaxed the weak continuity conditions for mortar finite elements more than is optimal.

We conclude that, among the three preconditioners for FETI algorithms for mortar finite element methods analyzed here, the new preconditioner $\widehat{M}$ is the best.

5.2. New mortars vs. classical mortars. Another objective of our study was to compare the performance of the FETI algorithms for new mortar element methods with the performance of the FETI algorithms for classical mortar element methods.

We used the same nonconforming partitions of our computational domain (see Figure 2) and considered new mortar finite elements on the subdomains. As explained in section 2.1, the only difference between the two mortars methods is due to different mortar conditions across the interface Γ. This results in different Lagrange multiplier

Table 2

Convergence results; two dimensional geometrically nonconforming partition; new mortar elements.

		New precond.			Block-diag. precond.			Dirichlet precond.		
N	H/h	Iter	Cond	Mflops	Iter	Cond	Mflops	Iter	Cond	Mflops
16	4	10	4.20	6.8e–1	18	18.40	1.2e+0	55	897	3.5e+0
16	8	12	5.15	8.1e+0	19	20.85	1.3e+1	70	908	4.7e+1
16	16	13	6.50	1.5e+2	20	25.40	2.3e+2	70	1018	8.0e+2
16	32	14	7.43	3.4e+3	21	30.11	5.1e+3	80	1090	2.0e+5
32	4	11	6.55	1.9e+0	19	23.58	3.0e+0	91	913	1.4e+1
32	8	13	7.61	1.9e+1	20	34.45	2.9e+1	101	1565	1.4e+2
32	16	14	8.87	3.5e+2	22	48.14	5.5e+2	114	1873	2.9e+3
32	32	16	9.80	9.1e+3	23	50.63	1.3e+4	117	2108	6.6e+4
64	4	14	7.29	6.0e+0	27	32.49	1.0e+1	278	1.0e+4	1.1e+2
64	8	16	8.69	5.6e+1	28	43.37	9.6e+1	292	1.6e+4	1.0e+3
64	16	18	10.83	1.0e+3	29	56.19	1.6e+3	297	1.9e+4	1.7e+4
64	32	20	12.57	2.5e+4	31	70.74	3.9e+4	312	2.2e+4	3.1e+5
128	4	14	7.60	1.3e+1	30	41.78	2.4e+1	614	1.0e+5	4.9e+2
128	8	17	9.57	1.3e+2	33	52.36	2.4e+2	677	1.3e+5	4.8e+3
128	16	19	11.75	2.2e+3	35	62.54	4.1e+3	755	1.6e+5	8.9e+4
128	32	21	13.07	5.6e+4	36	66.66	9.7e+4	–	–	–

matrices B.

We ran the same set of experiments for the new mortars discretization, for preconditioners $\widehat{M}$, $\overline{M}$, and M. The PCG iteration was stopped when the residual norm had decreased by a factor of 10^{-6}. We report the iteration count, the condition number approximation, and the flop counts in Table 2.

Due to the inherent simplification of the Lagrange multiplier matrix B for the new mortar constraints, we expected the results for the new mortar method to be similar but somewhat better than those for the classical mortar method. Indeed, this was confirmed by our numerical results.

The iteration counts for the new preconditioner $\widehat{M}$ were identical to those for $\widehat{M}$ for classical mortars. The condition number estimates and the flop counts were slightly smaller than in the classical mortar case but essentially the same. Therefore, $\widehat{M}$ scaled just as well as in the classical mortar case.

For the new mortar conditions, the matrix BB^t had fewer nonzero entries outside its block-diagonal structure and fewer terms to be dropped in order to obtain diag$B_\gamma B_\gamma^t$. Therefore, the block-diagonal preconditioner $\overline{M}$ was closer to $\widehat{M}$ than in the classical mortar case. This resulted in lower iteration counts and condition numbers for $\overline{M}$ than in the classical mortar case, and, consequently, in lower flop counts.

Once again, the iteration counts increased moderately with H/h and seemed to have a stronger than desired dependence on the number of subdomains. Overall, the block-diagonal preconditioner still performed worse than $\widehat{M}$ for the new mortars with iteration counts one and a half times higher and with significantly bigger condition numbers.

An even greater improvement over the classical mortar case was obtained for the Dirichlet preconditioner M. The number of iterations required for M in the new mortar case was less than half the number of iterations required in the classical mortar case, and a similar improvement can be observed for the flop counts. The condition number estimates were about an order of magnitude less than for the classical mortar case.

Despite these improvements, the FETI algorithm with the Dirichlet preconditioner still required hundreds of iterations to converge and the iteration count appeared to grow linearly with the number of subdomains. The condition number estimates were on the order of 10^3–10^5, much higher than desired. Therefore, as in the classical mortar case, applying the preconditioner M for the FETI method did not result in a scalable algorithm.

Once again, among the three preconditioners for FETI algorithms methods, the new preconditioner $\widehat{M}$ was the best. There was no significant improvement when using the new mortar elements instead of the classical mortar elements for the FETI algorithm with optimal preconditioner $\widehat{M}$.

5.3. Complexity study of the preconditioners. The last topic of this section is an analysis of how expensive it is to apply the preconditioners $\widehat{M}$ and $\overline{M}$, compared to applying the Dirichlet preconditioner M.

In each iteration step, we compute one vector multiplication by the preconditioner, which requires solving two systems with the matrix BB^t, and $\text{diag}B_\gamma B_\gamma^t$, respectively. In section 4.2, we mentioned that $\text{diag}B_\gamma B_\gamma^t$ is obtained from BB^t by eliminating the nonzero entries outside the diagonal blocks. There are two such types of entries; see section 4.3 for more details. While in the three dimensional case the sparsity pattern of BB^t is complex, in the two dimensional case there are relatively few nonzero entries; see Figure 4.

It is easy to see that $\text{diag}B_\gamma B_\gamma^t$ has a bandwidth of order H/h, the number of interior nodes on an arbitrary nonmortar. The matrix BB^t is also banded, but in this case the band depends on the ordering of the nodes on the interface, and it is possible to have bandwidth of order $1/h$. Therefore, multiplying a vector by $(BB^t)^{-1}$ is potentially an expensive operation. To minimize the effect of these vector multiplications, we computed the Cholesky factorizations of BB^t and $\text{diag}B_\gamma B_\gamma^t$ just once and stored the factors. Then, solving systems with BB^t or $\text{diag}\, B_\gamma B_\gamma^t$ amounted only to one back and one forward solve.

Our results showed that the costs of a vector multiplication by $(BB^t)^{-1}$ were between two and ten times larger than those associated with $(\text{diag}B_\gamma B_\gamma^t)^{-1}$. However, due to the sparsity pattern of BB^t, even the costs associated with $(BB^t)^{-1}$ were relatively small compared to those for other operations performed during one iteration, e.g., multipling a vector by the Schur complement or by its pseudoinverse; cf. Table 3. These low relative costs result in very similar flop counts per iteration step for the two preconditioners, almost identical for the case of many nodes per subdomain edge, i.e., $H/h = 16$ and $H/h = 32$.

As expected, the costs associated with $(BB^t)^{-1}$ in each iteration step decreased significantly, from six percent to less than .05 percent, when the partition was fixed and H/h increased, since the costs of multiplying S and $S^\dagger$ by a vector rose much faster than those corresponding to $(BB^t)^{-1}$.

From the flop counts reported in Table 1, it is clear that the improvement of the iteration count easily offsets the small extra costs due to the complexity of $\widehat{M}$ and $\overline{M}$.

6. Numerical results for three dimensional problems. In this section, we report numerical results for the FETI method for mortar finite element discretizations of a three dimensional problem. As before, we compare the performance of different FETI preconditioners and discuss the effects of using the new mortar finite elements instead of the classical ones. We include a study of the costs of applying the new preconditioner for the for classical mortar elements and more details on the convergence rate of the condition number approximation for some of our algorithms.

Table 3

Complexity study of one iteration step for the new preconditioner and the block-diagonal preconditioner; two dimensional geometrically nonconforming partition.

		New preconditioner			Block-diagonal preconditioner		
N	H/h	Mflops for $(BB^t)^{-1}$	Mflops per iteration	Ratio	Mflops for $(diagB_\gamma B_\gamma^t)^{-1}$	Mflops per iteration	Ratio
16	4	3.9e–3	6.9e–2	.06	1.4e–3	6.7e–2	.02
16	8	9.7e–3	6.8e–1	.02	4.6e–3	6.7e–1	.007
16	16	2.2e–2	1.1e+1	.002	1.1e–2	1.1e+1	.001
16	32	4.6e–2	2.4e+2	.0002	2.4e–2	2.4e+2	.0001
32	4	1.1e–2	1.7e–1	.07	3.2e–3	1.6e–1	.02
32	8	3.1e–2	1.5e+0	.02	9.9e–3	1.5e+0	.007
32	16	6.5e–2	2.5e+1	.003	2.4e–2	2.5e+1	.001
32	32	1.4e–1	5.7e+2	.0003	5.2e–2	5.7e+2	.00009
64	4	4.1e–2	4.3e–1	.10	6.9e–3	3.9e–1	.02
64	8	1.1e–1	3.5e+0	.03	2.2e–2	3.4e+0	.007
64	16	2.3e–1	5.6e+1	.004	5.4e–2	5.6e+1	.001
64	32	4.7e–1	1.2e+3	.0004	1.2e–1	1.2e+3	.00009
128	4	1.2e–1	9.3e–1	.13	1.4e–2	8.2e–1	.02
128	8	3.1e–1	7.4e+0	.04	4.4e–2	7.2e+0	.006
128	16	6.8e–1	1.2e+2	.006	1.1e–1	1.2e+2	.0009
128	32	1.4e+0	2.7e+3	.0005	2.3e–1	2.7e+3	.00009

As the model problem in three dimensions, we chose the Poisson equation on the unit cube $\Omega = [0,1]^3$ with zero Dirichlet boundary conditions. The right-hand side was selected such that the exact solution is known. The computational domain Ω was partitioned into 8, 16, and 32 nonconforming parallelepipeds, respectively. We chose these partitions such that in each case there exist floating subdomains, i.e., interior subdomains.

The subdomains of the partition had diameter of order H, and Q_1 elements of mesh size h were used in each subdomain. The number of nodes on each edge was, *on average*, 4, 8, and 16. Across the partition interface Γ the meshes did not match, and mortar conditions for three dimensional elements were enforced; cf. section 2.2. This results in a Lagrange multipliers matrix B, which, as explained for the two dimensional case, plays a very important role in all FETI algorithms.

We report the iteration counts, the condition number approximations, and the flop counts of the algorithms. The PCG iteration was stopped when the residual norm had decreased by a factor of 10^{-6}. All the experiments were carried out in MATLAB.

6.1. Convergence properties of the FETI algorithms. We did not compute the Schur complements explicitly but only stored those components of the stiffness matrices which were relevant for the multiplication of a vector by the Schur complement matrix and by the pseudoinverse of the Schur complement. We tested the performance of the same preconditioners as in the two dimensional case, i.e., the new preconditioner $\widehat{M}$, cf. (15); the preconditioner $\overline{M}$, cf. (14); and the Dirichlet preconditioner M, cf. (13). We report the iteration count, the condition number estimate, and the flop count of the algorithms in Table 4.

As in the two dimensional case, the FETI algorithm with the Dirichlet preconditioner M did not scale well and required thousands of iterations to converge. Since it soon became clear that M was not an optimal preconditioner, and due to significant computational costs, we performed only tests for every partition of Ω in the case of 4 nodes on each edge, and for the 8 and the 16 subdomains partitions for the case of 8

Table 4
Convergence results; three dimensional geometrically nonconforming partition; classical mortar elements.

		New precond.			Block-diag. precond.			Dirichlet precond.		
N	H/h	Iter	Cond	Mflops	Iter	Cond	Mflops	Iter	Cond	Mflops
8	4	11	4.31	1.8e+0	33	55	3.9e+0	866	2.2e+6	9.9e+1
8	8	14	6.54	4.9e+1	33	70	7.3e+1	7984	4.4e+7	1.7e+4
8	16	16	7.90	1.4e+3	38	81	2.4e+3	–	–	–
16	4	13	6.77	6.6e+0	36	75	9.7e+0	2985	1.2e+7	7.7e+2
16	8	15	8.20	1.6e+2	37	85	1.7e+2	14169	2.7e+8	6.3e+4
16	16	17	9.28	4.1e+3	50	176	6.5e+3	–	–	–
32	4	14	8.29	3.1e+1	44	141	2.7e+1	4156	3.5e+7	2.4e+3
32	8	16	9.77	1.1e+3	55	350	5.5e+2	–	–	–
32	16	17	10.69	2.7e+4	69	523	2.9e+4	–	–	–

nodes on each edge of the subdomains.

The iteration count seemed to be a linear function of the number of subdomains, and it grew by an order of magnitude when H/h was doubled while keeping the 8 and the 16 subdomain partitions fixed. The computational costs were also at least two orders of magnitude bigger than those for the other preconditioners and deteriorated as the number of nodes per subdomain edge increased.

The condition number estimates followed a similar dependence pattern on H/h and the number of subdomains. They were on the order of 10^6–10^8, much worse than even in the two dimensional case. In Figure 3, we present the convergence pattern of the condition number estimates for the eight subdomains partition with $H/h = 4$. For M, the PCG iteration was stopped when the residual norm had decreased by a tolerance factor of 10^{-6}, while for $\widehat{M}$ and $\overline{M}$ the tolerance was set at 10^{-10}.

For the Dirichlet preconditioner, there was no clear convergence pattern for the condition number estimates. This suggests that the (extremely large) condition number approximations reported in Table 4 are just lower bounds for the actual condition number. For the other two preconditioners, convergence was achieved early in the iteration count. The estimates reported in Table 4 are within one percent of the condition number corresponding to a tolerance of 10^{-10}.

The new preconditioner $\widehat{M}$ scaled similarly to the two dimensional case and to the Dirichlet preconditioner in the conforming case. The number of iterations grew very slowly when the number of nodes on each subdomain edge (i.e., H/h) was fixed and the number of subdomains was increased. When the partition was kept unchanged and H/h was increased, the iteration count increased slightly, and it seemed to have a polylogarithmic dependence on H/h.

The convergence analysis for the block-diagonal preconditioner $\overline{M}$ is particularly interesting in the three dimensional case; $\overline{M}$ was a possible alternative to $\widehat{M}$ since it required significantly less computational effort per iteration step. However, our results showed a much stronger than desired dependence of the iteration count for $\overline{M}$ on the number of nodes on each subdomain edge. This dependence grew stronger as the number of subdomains in the partition increased. The number of iterations increased with the number of subdomains, another undesirable property. The condition number estimates followed a similar pattern and were significantly larger than those corresponding to $\widehat{M}$. Their relatively large values, on the order of 10^2, and their dependence on the number of subdomains were unsatisfactory.

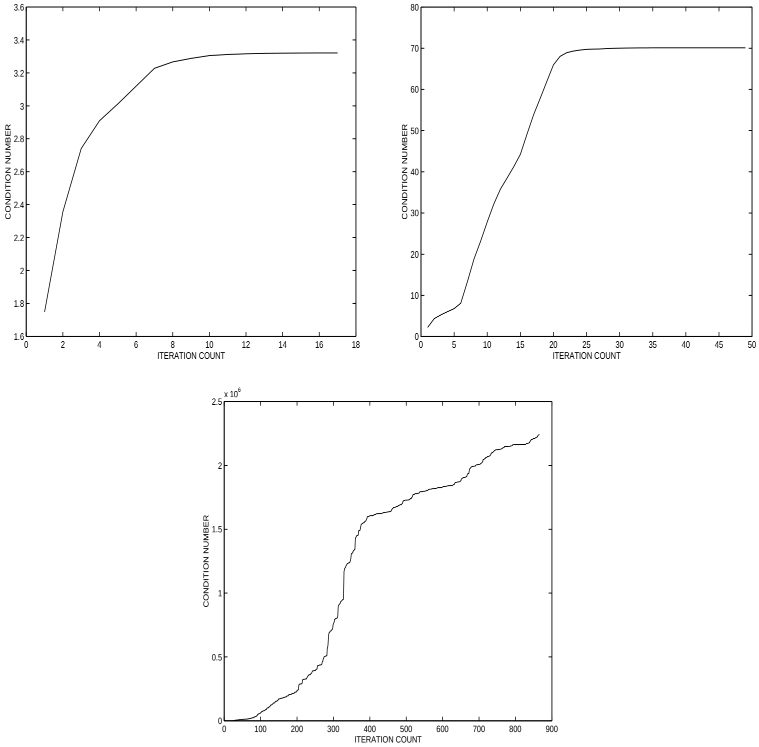


FIG. 3. *Convergence pattern for the condition number; three dimensional geometrically nonconforming partition, $N = 8$, $H/h = 4$. Top left: new preconditioner, tol $= 10^{-10}$; top right: block-diagonal preconditioner, tol $= 10^{-10}$; bottom: Dirichlet preconditioner, tol $= 10^{-6}$.*

TABLE 5
Convergence results; three dimensional geometrically nonconforming partition; new mortar elements.

		New precond.			Block-diag. precond.			Dirichlet precond.		
N	H/h	Iter	Cond	Mflops	Iter	Cond	Mflops	Iter	Cond	Mflops
8	4	11	4.46	1.7e+0	29	42	3.3e+0	272	2.8e+4	3.0e+1
8	8	14	6.52	4.8e+1	30	55	6.5e+1	648	4.1e+4	1.3e+3
8	16	16	7.80	1.4e+3	37	66	2.4e+3	720	4.7e+4	4.5e+4
16	4	13	6.69	6.4e+0	33	59	8.8e+0	817	2.2e+5	2.0e+2
16	8	15	8.17	1.6e+2	34	74	1.5e+2	1306	3.0e+5	5.7e+3
16	16	17	9.19	4.1e+3	43	115	5.6e+3	1870	3.5e+5	2.4e+5
32	4	14	8.02	3.1e+1	41	113	2.3e+1	989	4.2e+5	5.3e+2
32	8	16	9.28	1.1e+3	46	219	4.5e+2	1503	6.2e+5	1.4e+4
32	16	17	10.26	2.3e+4	53	331	3.6e+4	–	–	–

6.2. New mortars vs. classical mortars. In this section, we compare the performance of the FETI algorithms for new mortar element methods with that of the FETI algorithms for classical mortar element methods.

Using the same nonconforming partitions of Ω as before, we introduced new mortar finite elements on the subdomains and run the same set of experiments as before. We report the results in Table 5.

The iteration counts for the new preconditioner were identical to those for classical mortars. The condition number estimates were slightly smaller than in the classical mortar case but essentially the same. The flop counts were between one percent and five percent smaller than in the classical mortar case. In other words, $\widehat{M}$ scaled as well as in the classical mortar case.

As explained in the two dimensional case, the new mortar conditions resulted in simpler mortar conditions. An important consequence was that the off-diagonal entries of BB^t were fewer and smaller in absolute value than in the classical mortar case. Therefore, the block-diagonal preconditioner $\overline{M}$ was closer to $\widehat{M}$ than in the classical mortar case.

This generated a clear improvement for the iteration count and for the condition number estimate of the block-diagonal preconditioner $\overline{M}$. The number of iterations decreased from the classical mortar case, in particular when the iteration count for $\overline{M}$ was higher than desired, e.g., for the partition of Ω into 32 subdomains. It also resulted in a decrease in the flop counts. However, the iteration count appeared to depend on the number of subdomains when the number of nodes on each edges was fixed. The dependence of the iteration count on H/h seemed to be stronger than polylogarithmic.

The improvement generated by the new mortar method was even more significant for the Dirichlet preconditioner. The iteration count decreased to hundreds of iterations, instead of thousands, as was the case for the classical mortar method. The condition number estimates were lower by two orders of magnitude, in a range of order 10^4–10^5, and the flop counts were one order of magnitude bigger than for the other preconditioners. However, the FETI algorithm with the Dirichlet preconditioner did not scale as a good domain decomposition method.

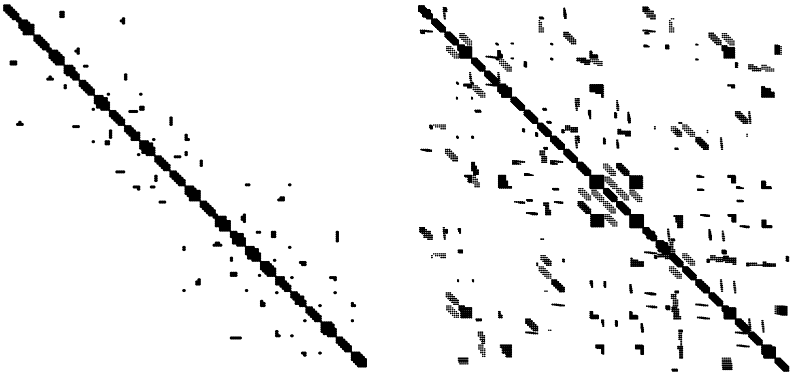


FIG. 4. *Sparsity pattern of BB^t. Left: two dimensional partition, $N = 16$, $H/h = 8$; right: three dimensional partition, $N = 16$, $H/h = 8$.*

6.3. Complexity study of the preconditioners. A comparison between the block-diagonal and the new preconditioner showed that using the preconditioner $\overline{M}$ resulted in a method which converged in about three times as many iterations than when $\widehat{M}$ was used. We recall that, in the two dimensional case, the number of iterations for the method with $\overline{M}$ was only about twice as large as that for $\widehat{M}$. This appears to be due to the fact that, in the three dimensional case, there are many nodes, e.g., the nodes on the wire baskets of the subdomains, which influence several nonmortar conditions. Therefore, the block-diagonal structure of BB^t is no longer as dominant, and many nonzero entries of BB^t need to be dropped; see Figure 4 for the differences in the sparsity pattern of BB^t for the two dimensional and three dimensional cases.

The flop counts for $\overline{M}$ were less than twice as large as those required for the convergence of the new preconditioner, even if the FETI method with $\overline{M}$ required about three times as many iterations as that with $\widehat{M}$. Moreover, for partitions with many subdomains and a small number of nodes on each edge, e.g., for $N = 32$ and $H/h = 4$ or $H/h = 8$, the complexities of the mortar conditions and of the Lagrange multiplier matrix B are higher relative to those of the Schur complement and its pseudoinverse. In these cases, the flop count for the block-diagonal preconditioner was less than that for the new preconditioner, despite the difference in iteration count.

This suggested that the costs of applying $(BB^t)^{-1}$ were significant in the three dimensional case, and this was confirmed by our results. In Table 6, we present the costs of applying $(BB^t)^{-1}$ and $\mathrm{diag}B_\gamma B_\gamma^t$ twice during an iteration step, relative to the total flop count for one iteration step.

Table 6

Complexity study of one iteration step for the new preconditioner and the block-diagonal preconditioner; three dimensional geometrically nonconforming partition.

		New preconditioner			Block-diagonal preconditioner		
N	H/h	Mflops for $(BB^t)^{-1}$	Mflops per iteration	Ratio	Mflops for $(diagB_\gamma B_\gamma^t)^{-1}$	Mflops per iteration	Ratio
8	4	3.2e–2	1.5e–1	.22	4.8e–3	1.2e–1	.04
8	8	5.8e–1	2.7e+0	.21	9.2e–2	2.2e+0	.04
8	16	6.8e+0	6.9e+1	.10	1.2e+0	6.4e+1	.02
16	4	1.3e–1	3.9e–1	.34	1.1e–2	2.7e–1	.04
16	8	2.1e+0	6.6e+0	.32	1.8e–1	4.6e+0	.04
16	16	2.2e+1	1.5e+2	.15	1.9e+0	1.3e+2	.02
32	4	6.9e–1	1.3e+0	.55	3.0e–2	6.0e–1	.05
32	8	1.1e+1	2.1e+1	.54	5.0e–1	9.9e+0	.05
32	16	1.2e+2	3.8e+2	.32	5.4e+0	2.7e+2	.02

The costs associated with $(BB^t)^{-1}$ were between 10 and 55 percent of those for one iteration step. This was much higher than for the two dimensional case, when the relative costs were at most 13 percent, and as low as .02 percent; cf. Table 3. The costs associated with $(\mathrm{diagB}_\gamma B_\gamma^t)^{-1}$ were much smaller, at most 5 percent of those for one iteration step. This was the reason why the flop counts per iteration were significantly lower for the block-diagonal preconditioner than for the new preconditioner.

The dependence of the relative cost of applying $(BB^t)^{-1}$ on the number of subdomains N and the number of nodes on each edge H/h was similar to that for the two dimensional case. It increased when H/h was kept fixed while the partition became more complex, and decreased when the partition was kept unchanged and H/h was increased. These results are consistent with the increased costs of multiplying a vector by the Schur complement and the pseudoinverse of the Schur complement when H/h increases, and the increased complexity of the Lagrange multiplier matrix B when the partition had more subdomains.

7. Continuity and mortar conditions for matching meshes. In the classical FETI algorithm, the underlying partition of Ω is geometrically conforming, the meshes across the interface match, and continuity conditions are enforced across the interface; cf. section 3. However, it is also possible to require mortar matching across the interface. It is important to note that using either mortar elements or conforming finite elements results in approximation errors of the exact solution that are of the

Table 7

Convergence results; two dimensional geometrically conforming partition; matching grids; and new mortar constraints.

		New precond.			Block-diag. precond.			Dirichlet precond.		
N	H/h	Iter	Cond	Mflops	Iter	Cond	Mflops	Iter	Cond	Mflops
16	4	5	2.18	3.4e–1	5	2.41	3.4e–1	5	3.93	3.3e–1
16	8	5	2.42	2.9e+0	6	2.49	3.5e+0	6	3.92	3.4e+0
16	16	6	3.18	7.2e+1	7	3.27	8.5e+1	7	5.11	8.5e+1
16	32	6	3.59	1.3e+3	7	4.31	1.6e+3	8	6.73	1.8e+3
36	4	7	2.34	1.5e+0	8	3.71	1.6e+0	8	3.76	1.6e+0
36	8	9	2.90	1.4e+1	10	4.70	1.6e+1	11	4.81	1.7e+1
36	16	9	3.70	2.5e+2	10	4.62	2.8e+2	12	6.24	3.3e+2
36	32	10	4.22	6.9e+3	11	5.14	7.6e+3	13	8.11	9.0e+3
64	4	8	2.42	3.5e+0	9	3.93	3.8e+0	9	4.00	3.8e+0
64	8	9	3.09	2.7e+1	11	5.02	3.4e+1	12	5.20	3.5e+1
64	16	11	3.98	7.3e+2	11	5.37	7.2e+2	13	6.77	8.6e+2
64	32	12	4.53	1.4e+4	13	5.55	1.5e+4	15	8.79	1.8e+4
121	4	9	2.45	8.6e+0	10	4.08	9.0e+0	10	4.18	8.9e+0
121	8	10	3.14	6.5e+1	12	5.20	7.7e+1	13	7.10	8.3e+1
121	16	11	4.04	1.2e+3	13	5.73	1.4e+3	15	8.29	1.6e+3
121	32	13	4.66	3.7e+4	14	5.84	4.0e+4	17	9.19	4.8e+4

same order. In particular, for our example, the difference in the discretization errors corresponding to the two types of elements is negligible.

In this section, we compare the performance of the resulting FETI algorithms for the two different types of matchings. We considered both two and three dimensional problems. For mortar finite elements, we tested FETI algorithms with all three preconditioners, while for conforming finite elements we used only the new preconditioner and the Dirichlet preconditioner. The block-diagonal preconditioner is identical to the new preconditioner for continuity matchings, since BB^t is a block-diagonal matrix.

The convergence results for the classical mortar methods and the new mortar methods were once again very similar, except for the case of the Dirichlet preconditioner, where the new mortars provided a significant improvement. However, our main goal was to compare the performance of continuity matchings versus mortar matchings. Therefore, we present only here the results for the new mortar methods, which always resulted in better algorithms than the classical mortar methods.

7.1. The two dimensional case. For the two dimensional experiments, the computational domain Ω, the unit square, was partitioned into 4×4, 6×6, 8×8, and 11×11 congruent squares, and Q_1 elements were used in each square. The meshes match across Γ, and nonredundant pointwise continuity conditions, or mortar conditions, were used across Γ for comparison purposes. Except for the different partitions, the experiments have the same parameters as in section 5. We report the iteration count, the condition number estimate, and the flop count for the FETI algorithms with new mortar finite elements in Table 7.

When new mortar conditions were used across the interface, computing the Lagrange multiplier matrix B was very simple for matching nodes. In particular, no computations of integrals resulting from the mortar conditions (1) were necessary. The new mortar conditions are equivalent to continuity conditions for all matchings except for those corresponding to the first and last interior nodes on the nonmortar sides, where the end point nodes are involved as well. Therefore, B was very similar for the two types of matchings, and BB^t was very close to twice the identity matrix.

Table 8

Convergence results; two dimensional geometrically conforming partition; matching grids; and continuity constraints.

		New precond.			Dirichlet precond.		
N	H/h	Iter	Cond	Mflops	Iter	Cond	Mflops
16	4	7	2.17	4.7e–1	18	23.02	1.2e+0
16	8	8	2.91	4.6e+0	19	28.11	1.1e+1
16	16	10	3.90	1.2e+2	20	33.75	2.4e+2
16	32	11	4.47	2.4e+3	21	39.01	4.7e+3
36	4	8	2.18	1.6e+0	24	23.23	4.8e+0
36	8	10	2.93	1.6e+1	26	28.15	4.1e+1
36	16	12	3.91	3.3e+2	26	34.03	7.2e+2
36	32	13	4.50	9.0e+3	28	39.13	1.9e+4
64	4	8	2.19	3.4e+0	25	23.34	1.0e+1
64	8	10	2.95	2.9e+1	28	28.19	8.2e+1
64	16	12	3.90	7.9e+2	28	34.03	1.8e+3
64	32	13	4.51	1.5e+4	29	39.33	3.4e+4
121	4	9	2.21	8.1e+0	27	23.35	2.4e+1
121	8	10	2.99	6.4e+1	29	28.23	1.8e+2
121	16	12	3.92	1.3e+3	29	34.10	3.1e+3
121	32	13	4.54	3.7e+4	30	39.44	8.5e+4

Almost no extra work was required when a system with the matrix BB^t had to be solved.

Another consequence of matching meshes was that BB^t had very few nodes outside its block-diagonal structure diag$B_\gamma B_\gamma^t$. Therefore, we expected the convergence results for the new preconditioner and for the block-diagonal preconditioner to be similar. Indeed, the results from Table 7 show that $\widehat{M}$ and $\overline{M}$ behaved similarly in terms of iteration counts and condition number estimates, which were just slightly higher for the block-diagonal preconditioner $\overline{M}$. Both preconditioners scaled very well with the number of subdomains and the number of nodes on each edge. The computational costs for one iteration step were almost identical, which resulted in better flop counts for $\widehat{M}$, even when the iteration counts differed by only one iteration.

In contrast with the other algorithms, the Dirichlet preconditioner M performed very well for two dimensional problems with matching nodes. The iteration counts were very small, comparable to those corresponding to $\widehat{M}$ and $\overline{M}$. However, since BB^t is close to twice the identity matrix, the computational costs per iteration do not show a relevant improvement for M. The Dirichlet preconditioner yielded higher flop counts than the other two preconditioners.

We now turn our attention to the case when pointwise continuity was enforced across the interface; cf. Table 8. As expected, both the Dirichlet preconditioner and the new preconditioner had very good scaling properties. In particular, the condition number estimates were almost constant when H/h was kept fixed and the number of subdomains was changed. However, $\widehat{M}$ converged in less than half the number of iterations necessary for M, and the same was true for the computational costs.

For continuity matchings, the vector matrix multiplication by $B^t(BB^t)^{-1}B$ is very easy to compute, since it is close to an operator from the balancing algorithm; see [33]. It is possible to write the PCG algorithm with the new preconditioner such that only the product of a vector by $B^t(BB^t)^{-1}B$, and not by $(BB^t)^{-1}$, needs to be computed.

Using the data from Table 7 and Table 8, we can address the main topic of this section, finding the best FETI algorithm for conforming partitions of Ω. We compared

Table 9

Convergence results; three dimensional geometrically conforming partition; matching grids; and new mortar constraints.

		New precond.			Block-diag. precond.			Dirichlet precond.		
N	H/h	Iter	Cond	Mflops	Iter	Cond	Mflops	Iter	Cond	Mflops
8	4	8	3.13	7.8e–1	12	4.03	9.8e–1	127	2.9e+4	1.1e+1
8	8	9	3.92	1.9e+1	12	4.78	2.2e+1	212	3.6e+4	3.8e+2
8	16	11	4.41	6.8e+2	13	5.99	7.7e+2	301	4.2e+4	1.8e+4
16	4	8	5.31	2.4e+0	12	7.34	2.3e+0	141	6.5e+4	2.3e+1
16	8	10	7.00	5.5e+1	12	9.39	4.7e+1	233	9.0e+4	8.7e+2
16	16	11	8.32	1.4e+3	13	11.89	1.6e+3	341	1.2e+5	4.1e+4
32	4	9	5.70	3.9e+0	12	11.55	4.0e+0	400	1.6e+5	1.3e+2
32	8	11	8.24	8.8e+1	14	14.99	8.7e+1	630	2.0e+5	3.8e+3
32	16	12	10.31	2.2e+3	15	18.62	2.8e+3	–	–	–

the new preconditioner for new mortar matchings and for continuity conditions. The iteration counts and the condition number estimates were slightly lower for the new mortar case. The flop counts were also better for mortar matchings, since the Lagrange multiplier matrices had similar structure for the two types of matchings and the costs per iteration step were almost identical.

We conclude that the new mortar matchings represent an improvement over the continuity matchings. This might be due in part to the fact that the mortar matching conditions corresponding to the first and last interior points on the nonmortars replace the continuity constraints at the crosspoints and their neighboring nodes.

7.2. The three dimensional case. For the three dimensional experiments, the unit cube was partitioned into $2 \times 2 \times 2$, $2 \times 2 \times 4$, and $2 \times 4 \times 4$ geometrically conforming, noncongruent parallelepipeds; Q_1 meshes were considered in each subdomain such that the meshes across the interface matched. Across Γ, nonredundant pointwise continuity conditions, or biorthogonal mortar conditions, were enforced by using Lagrange multipliers.

For the mortar matchings, we present convergence results only for the new mortar method. We run the same set of experiments as in section 6 for all three preconditioners and for 4, 8, and 16 nodes on each subdomain edge. We report the results in Table 9.

For the three dimensional case, the Lagrange multiplier matrix B was no longer very close to a multiple of the identity. The mortar matching conditions were different than the continuity matchings for all the interior nodes on the nonmortar faces with neighbors on the boundary of the face.

Once again, the new preconditioner $\widehat{M}$ scaled well. The iteration count and the condition number estimate depended only weakly on H/h and on N, the number of subdomains in the partition of Ω. A similar behavior was observed for the block-diagonal preconditioner, at somewhat higher iteration counts. However, due to the relative complexity of BB^t, which was no longer very close to its block-diagonal structure $\text{diag}B_\gamma B_\gamma^t$, the preconditioner $\overline{M}$ required less computational effort per iteration than $\widehat{M}$. The difference in the iteration counts has thus been compensated, the two preconditioners resulting in algorithms with very close flop counts.

Unlike in the two dimensional case with matching nodes, the Dirichlet preconditioner M required hundreds of iterations to converge and did not have good scalability properties. The condition number estimates were on the order of 10^4–10^5, and the flop counts were at least one order of magnitude greater than for the other preconditioners.

Table 10

Convergence results; three dimensional geometrically conforming partition; matching grids; and continuity constraints.

		New precond.			Dirichlet precond.		
N	H/h	Iter	Cond	Mflops	Iter	Cond	Mflops
8	4	6	1.77	4.6e–1	21	75.54	1.5e+0
8	8	8	2.50	1.4e+1	25	80.30	4.4e+1
8	16	9	3.17	5.3e+2	27	84.25	1.6e+3
16	4	8	3.34	1.4e+0	31	84.12	5.4e+0
16	8	9	4.91	3.4e+1	35	90.73	1.3e+2
16	16	11	6.30	1.3e+3	36	94.35	4.3e+3
32	4	8	4.13	2.6e+0	38	95.51	1.2e+1
32	8	10	6.76	6.0e+1	41	98.12	2.4e+2
32	16	12	8.73	2.2e+3	43	99.82	7.9e+3

The convergence results for pointwise continuity matchings across the interface are reported in Table 10. The new preconditioner yielded a scalable algorithm with very low iteration counts and condition numbers. The Dirichlet preconditioner also resulted in a scalable algorithm but required at least three times as many iterations as $\widehat{M}$ for convergence. The condition number estimates were much larger as well but depended weakly on changes of parameters H/h and N. The complexity of the matrix B was compensated by the improvement in the iteration counts. The flop counts for $\widehat{M}$ were at least half of those for M.

We conclude this section by discussing the differences between the two types of matchings for three dimensional problems. The new preconditioner resulted in the best algorithms for both new mortar and continuity matchings. The iteration counts and the condition number estimates were slightly lower for the continuity case. The computational effort per iteration required in the mortar case is greater than in the continuity case, since BB^t is no longer very close to a multiple of the identity. Coupled with lower iteration counts, this results in better flop counts for the continuity matching algorithms.

REFERENCES

[1] Y. Achdou and Y. A. Kuznetsov, *Substructuring preconditioners for finite element methods on nonmatching grids*, East-West J. Numer. Math., 3 (1995), pp. 1–28.

[2] Y. Achdou, Y. A. Kuznetsov, and O. Pironneau, *Substructuring preconditioners for the Q_1 mortar element method*, Numer. Math., 71 (1995), pp. 419–449.

[3] Y. Achdou, Y. Maday, and O. B. Widlund, *Iterative substructuring preconditioners for mortar element methods in two dimensions*, SIAM J. Numer. Anal., 36 (1999), pp. 551–580.

[4] F. Ben Belgacem, Y. Maday, and A. Buffa, *The mortar finite element method for Maxwell equations in 3D*, C. R. Acad. Sci. Paris Sér. I Math., 329 (1999), pp. 903–908.

[5] F. Ben Belgacem, *The mortar element method with Lagrange multipliers*, Numer. Math., 84 (1999), pp. 173–197.

[6] F. Ben Belgacem and Y. Maday, *Non-conforming Spectral Element Method for Second Order Elliptic Problem in 3D*, Technical Report R93039, Laboratoire d'Analyse Numérique, Université Pierre et Marie Curie—Centre National de la Recherche Scientifique, Paris, 1993.

[7] F. Ben Belgacem and Y. Maday, *The mortar element method for three dimensional finite elements*, RAIRO Modél. Math. Anal. Numér., 31 (1997), pp. 289–302.

[8] C. Bernardi, Y. Maday, and A. T. Patera, *A new nonconforming approach to domain decomposition: The mortar element method*, in Collège de France Seminar, H. Brezis and J.-L. Lions, eds., Longman Scientific and Technical, Harlow, UK, 1994, pp. 13–51.

[9] D. Braess, W. Dahmen, and C. Wieners, *A multigrid algorithm for the mortar finite element method*, SIAM J. Numer. Anal., 37 (1999), pp. 48–69.

[10] D. Braess, M. Dryja, and W. Hackbusch, *Multigrid method for nonconforming FE–discretisations with application to nonmatching grids*, Computing, 63 (1999), pp. 1–25.

[11] S. C. Brenner, *A new look at FETI*, Technical Report, University of South Carolina, Columbia, SC, 2000.

[12] A. Buffa, Y. Maday, and F. Rapetti, *A Sliding Mesh-Mortar Method for a Two-Dimensional Eddy Currents Model of Electric Engines*, Technical Report R99002, Laboratoire d'Analyse Numérique, Université Pierre et Marie Curie—Centre National de la Recherche Scientifique, Paris, 1999.

[13] X.-C. Cai, M. Dryja, and M. Sarkis, *Overlapping nonmatching grid mortar element methods for elliptic problems*, SIAM J. Numer. Anal., 36 (1999), pp. 581–606.

[14] M. A. Casarin and O. B. Widlund, *A hierarchical preconditioner for the mortar finite element method*, Electron. Trans. Numer. Anal., 4 (1996), pp. 75–88.

[15] M. Dryja, *Additive Schwarz methods for elliptic mortar finite element problems*, in Modeling and Optimization of Distributed Parameter Systems with Applications to Engineering, K. Malanowski, Z. Nahorski, and M. Peszynska, eds., Chapman and Hall, New York, 1996, pp. 31–50.

[16] M. Dryja, *An iterative substructuring method for elliptic mortar finite element problems with a new coarse space*, East-West J. Numer. Math., 5 (1997), pp. 79–98.

[17] M. Dryja, B. F. Smith, and O. B. Widlund, *Schwarz analysis of iterative substructuring algorithms for elliptic problems in three dimensions*, SIAM J. Numer. Anal., 31 (1994), pp. 1662–1694.

[18] C. Farhat, P.-S. Chen, and J. Mandel, *A scalable Lagrange multiplier based domain decomposition method for time-dependent problems*, Internat. J. Numer. Methods Engrg., 38 (1995), pp. 3831–3853.

[19] C. Farhat, P.-S. Chen, and F.-X. Roux, *The two-level FETI method* II*: Extensions to shell problems, parallel implementation, and performance results*, Comput. Methods Appl. Mech. Engrg., 155 (1998), pp. 153–179.

[20] C. Farhat, M. Lesoinne, P. Le Tallec, K. Pierson, and D. Rixen, *FETI-DP: A dual-primal unified FETI method* I*: A faster alternative to the two-level FETI method*, Internat. J. Numer. Methods Engrg., 50 (2001), pp. 1523–1544.

[21] C. Farhat, M. Lesoinne, and K. Pierson, *A scalable dual-primal domain decomposition method*, Numer. Linear Algebra Appl., 7 (2000), pp. 687–714.

[22] C. Farhat, A. P. Macedo, and M. Lesoinne, *A two-level domain decomposition method for the iterative solution of high frequency exterior Helmholtz problems*, Numer. Math., 85 (2000), pp. 283–308.

[23] C. Farhat, A. P. Macedo, M. Lesoinne, F.-X. Roux, F. Magoulès, and A. de La Bourdonnaie, *A non-overlapping domain decomposition method for the exterior Helmholtz problem*, in Proceedings of the Tenth International Conference of Domain Decomposition Methods, J. Mandel, C. Farhat, and X.-C. Cai, eds., Contemp. Math. 218, AMS, Providence, RI, 1998, pp. 42–66.

[24] C. Farhat and J. Mandel, *The two-level FETI method for static and dynamic plate problems part* I*: An optimal iterative solver for biharmonic systems*, Comput. Methods Appl. Mech. Engrg., 155 (1998), pp. 129–151.

[25] C. Farhat, J. Mandel, and F.-X. Roux, *Optimal convergence properties of the FETI domain decomposition method*, Comput. Methods Appl. Mech. Engrg., 115 (1994), pp. 367–388.

[26] C. Farhat and F.-X. Roux, *A method of finite element tearing and interconnecting and its parallel solution algorithm*, Internat. J. Numer. Methods Engrg., 32 (1991), pp. 1205–1227.

[27] C. Farhat and F.-X. Roux, *Implicit parallel processing in structural mechanics*, in Computational Mechanics Advances, J. Tinsley Oden, ed., North-Holland, Amsterdam, 1994, pp. 1–124.

[28] L. Franca, C. Farhat, A. P. Macedo, and M. Lesoinne, *Residual-free bubbles for the Helmholtz equation*, Internat. J. Numer. Methods Engrg., 40 (1997), pp. 4003–4009.

[29] L. Franca and A. P. Macedo, *A two-level finite element method and its application to the Helmholtz equation*, Internat. J. Numer. Methods Engrg., 43 (1998), pp. 23–32.

[30] R. Hoppe, Y. Iliash, Y. Kuznetsov, Y. Vassilevski, and B. Wohlmuth, *Analysis and parallel implementation of adaptive mortar finite element methods*, East-West J. Numer. Math., 6 (1998), pp. 223–248.

[31] R. H. W. Hoppe, *Mortar edge element methods in R^3*, East-West J. Numer. Math., 7 (1999), pp. 159–173.

[32] A. Klawonn and O. B. Widlund, *A domain decomposition method with Lagrange multipliers*

for linear elasticity, SIAM J. Sci. Comput., 22 (2000), pp. 1199–1219.

[33] A. Klawonn and O. B. Widlund, *FETI and Neumann–Neumann iterative substructuring methods: Connections and new results*, Comm. Pure Appl. Math., 54 (2001), pp. 57–90.

[34] C. Lacour, *Analyse et Resolution Numérique de Méthodes de Sous-Domaines Non Conformes pour des Problèmes de Plaques*, Ph.D. thesis, Université Pierre et Marie Curie, Paris, 1997.

[35] C. Lacour, *Iterative substructuring preconditioners for the mortar finite element method*, in Proceedings of the Ninth International Conference on Domain Decomposition Methods, P. Bjørstad, M. Espedal, and D. Keyes, eds., Bergen, Norway, 1996, pp. 406–412.

[36] C. Lacour and Y. Maday, *Two different approaches for matching nonconforming grids: The mortar element method and the FETI method*, BIT, 37 (1997), pp. 720–738.

[37] P. Le Tallec, *Neumann-Neumann domain decomposition algorithms for solving 2D elliptic problems with nonmatching grids*, East-West J. Numer. Math., 1 (1993), pp. 129–146.

[38] P. Le Tallec, T. Sassi, and M. Vidrascu, *Three-dimensional domain decomposition methods with nonmatching grids and unstructured grid solvers*, in Proceedings on the Seventh International Conference of Domain Decomposition Methods in Scientific and Engineering Computing, D. E. Keyes and J. Xu, eds., Contemp. Math. 180, AMS, Providence, RI, 1994, pp. 61–74.

[39] J. Mandel, *Balancing domain decomposition*, Comm. Numer. Methods Engrg., 9 (1993), pp. 233–241.

[40] J. Mandel and R. Tezaur, *Convergence of a substructuring method with Lagrange multipliers*, Numer. Math., 73 (1996), pp. 473–487.

[41] J. Mandel, R. Tezaur, and C. Farhat, *A scalable substructuring method by Lagrange multipliers for plate bending problems*, SIAM J. Numer. Anal., 36 (1999), pp. 1370–1391.

[42] F. Rapetti and A. Toselli, *A FETI preconditioner for two dimensional edge element approximations of Maxwell's equations on nonmatching grids*, SIAM J. Sci. Comput., 23 (2001), pp. 92–108.

[43] D. Rixen and C. Farhat, *Preconditioning the FETI method for problems with intra- and inter-subdomain coefficient jumps*, in Proceedings of the Ninth International Conference on Domain Decomposition Methods, P. Bjørstad, M. Espedal, and D. Keyes, eds., Bergen, Norway, 1996, pp. 472–479.

[44] D. Rixen and C. Farhat, *A simple and efficient extension of a class of substructure based preconditioners to heterogeneous structural mechanics problems*, Internat. J. Numer. Methods Engrg., 44 (1999), pp. 489–516.

[45] P. Seshaiyer and M. Suri, *Uniform hp convergence results for the mortar finite element method*, Math. Comp., 69 (1997), pp. 481–500.

[46] D. Stefanica, *Domain Decomposition Methods for Mortar Finite Elements*, Ph.D. thesis, Courant Institute of Mathematical Sciences, 1999, Technical Report 804, Department of Computer Science, Courant Institute of Mathematical Sciences, New York University, 1999.

[47] D. Stefanica, *FETI algorithms for 3D mortar finite elements*, in Finite Element Methods: Three-dimensional Problems, M. Krízek and P. Neittaanmäki, eds., GAKUTO Internat. Ser. Math. Sci. Appl., to appear.

[48] D. Stefanica and A. Klawonn, *The FETI method for mortar finite elements*, in Proceedings of the Eleventh International Conference on Domain Decomposition Methods, C-H. Lai, P. Bjørstad, M. Cross, and O. Widlund, eds., Greenwich, UK, 1998, pp. 121–129.

[49] R. Tezaur, *Analysis of Lagrange Multiplier Based Domain Decomposition Methods*, Ph.D. thesis, University of Colorado, Denver, CO, 1998.

[50] A. Toselli and A. Klawonn, *A FETI domain decomposition method for edge element approximations in two dimensions with discontinuous coefficients*, SIAM J. Numer. Anal., 39 (2001), pp. 932–956.

[51] C. Wieners and B. Wohlmuth, *A General Framework for Multigrid Methods for Mortar Finite Elements*, Technical Report 415, Mathematisch-Naturwissenschaftliche Fakultät, Universität Augsburg, Augsburg, Germany, 1999.

[52] B. Wohlmuth, *A Mortar Finite Element Method Using Dual Spaces for the Lagrange Multiplier*, Technical Report 407, Mathematish-Naturwissenschaftliche Fakultät, Universität Augsburg, Augsburg, Germany, 1998.

[53] B. I. Wohlmuth, *Discretization Methods and Iterative Solvers Based on Domain Decomposition*, Habilitationsschrift Mathematisch-Naturwissenschaftliche Fakultät, Universität Augsburg, Augsburg, Germany, 1999.

SIAM J. Sci. Comput.
Vol. 23, No. 4, pp. 1161–1171

NEAR-OPTIMAL PARAMETERS FOR TIKHONOV AND OTHER REGULARIZATION METHODS*

DIANNE P. O'LEARY†

Abstract. Choosing the regularization parameter for an ill-posed problem is an art based on good heuristics and prior knowledge of the noise in the observations. In this work, we propose choosing the parameter, without a priori information, by approximately minimizing the distance between the true solution to the discrete problem and the family of regularized solutions. We demonstrate the usefulness of this approach for Tikhonov regularization and for an alternate family of solutions. Further, we prove convergence of the regularization parameter to zero as the standard deviation of the noise goes to zero.

Key words. ill-posed problems, regularization, Tikhonov

AMS subject classifications. 65R30, 65F20

PII. S1064827599354147

1. Introduction.

Linear, discrete ill-posed problems of the form

$$\min_x \|Ax-b\|_2, \quad \text{or, equivalently,} \quad A^*Ax = A^*b \tag{1.1}$$

arise, for example, from the discretization of first-kind Fredholm integral equations and occur in a variety of applications. We shall assume the following:

1. The full-rank matrix A is $m \times n$ with $m \geq n$.
2. A is ill-conditioned with no significant gap in the singular value spectrum. (A gap would make the problem somewhat easier.) The problem is normalized so that the largest singular value is 1.
3. The right-hand side b consists of true data plus random noise: $b = b_t + e$, where the components of e are an independent sample from a probability distribution with mean 0 and standard deviation s.
4. The discretization error caused by making a finite dimensional approximation to the continuous operator is much smaller than the noise.
5. The system satisfies the *discrete Picard condition*, which we will define in section 2 after introducing some notation.

The noise in the measurements, in combination with the ill conditioning of A, means that the exact solution of (1.1) has little relationship to the noise-free solution and is worthless. Instead, we use a *regularization* method to determine a solution that approximates the noise-free solution. Regularization methods replace the original operator by a better-conditioned, but related, one in order to diminish the effects of noise in the data and produce a *regularized solution* to the original problem. In this work, we first consider Tikhonov regularization, in which the problem (1.1) is replaced by

$$\min_x \left(\|Ax-b\|_2^2 + \lambda\|Lx\|_2^2\right), \quad \text{or, equivalently,} \quad (A^*A + \lambda L^*L)x = A^*b, \tag{1.2}$$

*Received by the editors April 14, 1999; accepted for publication (in revised form) June 12, 2001; published electronically November 7, 2001. This work was completed at the Departement Informatik, ETH Zürich, Switzerland. This work was supported in part by the U.S. National Science Foundation under grant CCR-97-32022 and by the National Institute of Standards and Technology.

http://www.siam.org/journals/sisc/23-4/35414.html

†Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742 (oleary@cs.umd.edu).

where L is a regularization operator chosen to obtain a solution with desirable properties, such as a small norm ($L = I$) or good smoothness (L a discrete approximation to a derivative operator), and $\lambda > 0$ is a scalar parameter. Throughout this paper we will use the 2-norm and denote it by $\| * \|$.

The central question in Tikhonov regularization is how to choose the parameter λ in order to produce a solution x close to the true noise-free solution x_{true}. Hoerl and Kennard [11] showed that on average a smaller error is produced using a nonzero λ, and numerous heuristics have been proposed for the choice of this parameter. Some of these (e.g., the discrepancy principle [14]) assume that the standard deviation of the noise is known. Others (e.g., generalized cross-validation [6] and the L-curve [8]) work with less knowledge of the noise properties. An interesting recent approach of Rust [17] uses visualization of residual and singular component plots to choose reasonable parameters. Pierce and Rust [15] minimize the lengths of confidence intervals using appropriate parameter choices, and Kilmer and O'Leary [13] discuss the choice of parameters when iterative solution methods are used.

In this work, we propose another rule for parameter choice. We go back to first principles: among all solutions in a given family such as Tikhonov, we want the solution that is a minimal distance from the true solution. Kay [12] has developed asymptotic expressions for this distance as the size of the problem grows large. Others have determined a Tikhonov parameter by minimizing a bound on this distance; Raus [16], Gfrerer [5], and Engl and Gfrerer [3] propose minimizing one such bound, while Hanke and Raus [7] propose an alternative. Rather than using asymptotic results or minimizing a bound, we compute in section 2 a parameter that approximately minimizes the distance to the true solution to the discretized problem and accomplishes this goal without a priori knowledge of the standard deviation or distribution of the noise in the observations. We discuss convergence of this choice in section 3. Section 4 contains a similar development for an alternative to Tikhonov regularization. Section 5 discusses some algorithmic issues, and in section 6 we show the effectiveness of these methods on numerical examples.

2. Choosing the Tikhonov regularization parameter. In order to analyze the problem, we convert to the coordinate system of the singular value decomposition (SVD) of A. For simplicity of exposition, we assume that the regularization operator L is the identity matrix. A similar development, using the generalized SVD, could be done for general L (see, for example, [10, sect. 2.1.2]), but the resulting function is considerably more complicated to compute and minimize.

Suppose $A = U\Sigma V^*$, where U and V have orthonormal columns and Σ is a matrix of zeros except for diagonal entries $\sigma_1 \geq \cdots \geq \sigma_n > 0$. Exploiting the property that $\|Uz\| = \|z\|$ and $\|V^* z\| = \|z\|$, the problem (1.2) takes the form

$$\min_z \|\Sigma z - \beta\|^2 + \lambda \|z\|^2 ,$$

where $\beta_i \equiv u_i^* b$ and $z = V^* x$. Setting the derivative equal to zero, we find that for a fixed value of λ, we need to solve the equation

$$(\Sigma^T \Sigma + \lambda I) z = \Sigma^T \beta.$$

Thus, the Tikhonov solution is

$$x_{tik} = \sum_{i=1}^{n} \frac{\beta_i \sigma_i}{\sigma_i^2 + \lambda} v_i, \tag{2.1}$$

where v_i is the ith column of V.

The true solution to the discrete (noise-free) problem is

$$x_{true} = \sum_{i=1}^{n} \frac{\beta_i - \epsilon_i}{\sigma_i} v_i ,$$

where $\epsilon_i \equiv u_i^* e$ represents the unknown noise component.

The goal in regularization is to produce a solution as close as possible to the true solution, so let us (rather naively) try to minimize this distance:

$$\min_{\lambda} \|x_{tik} - x_{true}\|^2 \equiv \min_{\lambda} f(\lambda) \, .$$

Using the singular value representation, we see that

$$f(\lambda) = \sum_{i=1}^{n} \left[\frac{\beta_i \sigma_i}{\sigma_i^2 + \lambda} - \frac{\beta_i - \epsilon_i}{\sigma_i} \right]^2 \, .$$

Setting the derivative equal to zero yields

$$\begin{aligned} 0 = g(\lambda) \equiv \frac{1}{2} f'(\lambda) &= - \sum_{i=1}^{n} \left[\frac{\beta_i \sigma_i}{\sigma_i^2 + \lambda} - \frac{\beta_i - \epsilon_i}{\sigma_i} \right] \left[\frac{\beta_i \sigma_i}{(\sigma_i^2 + \lambda)^2} \right] \\ &= \sum_{i=1}^{n} \frac{\beta_i^2 \lambda}{(\sigma_i^2 + \lambda)^3} - \sum_{i=1}^{n} \frac{\beta_i \epsilon_i}{(\sigma_i^2 + \lambda)^2} \, . \end{aligned}$$

Now the first summation in this last expression is computable, but the second is not because the noise values ϵ_i are unknown. However, there are two interesting properties of the second summation.

- First, the terms for $i \approx n$ tend to be the largest because the denominators are the smallest.
- Second, the system satisfies the discrete Picard condition, meaning that for large enough values of the discretization parameter n, the sequence of true data values $\{\beta_i - \epsilon_i\}$ goes to zero faster than the sequence of singular values $\{\sigma_i\}$. Thus, for terms with i greater than or equal to some parameter k, $\epsilon_i \approx \beta_i$.

Therefore, although we cannot compute the function $g(\lambda)$, we can compute the following approximation to it:

$$\hat{g}(\lambda) \equiv \sum_{i=1}^{n} \frac{\beta_i^2 \lambda}{(\sigma_i^2 + \lambda)^3} - \sum_{i=k}^{n} \frac{\beta_i^2}{(\sigma_i^2 + \lambda)^2} - \mathcal{E} \left(\sum_{i=1}^{k-1} \frac{\beta_i \epsilon_i}{(\sigma_i^2 + \lambda)^2} \right)$$

for a suitable index k, depending on the standard deviation s. Finding the zero of this function yields an approximation to the optimal value of λ. The last term denotes the expected value of the quantity. Under assumption 3 of section 1, β_i is some true value plus noise ϵ_i, so $\mathcal{E}(\beta_i \epsilon_i) = \mathcal{E}(\epsilon_i^2) = s^2$, and

$$\hat{g}(\lambda) = \sum_{i=1}^{n} \frac{\beta_i^2 \lambda}{(\sigma_i^2 + \lambda)^3} - \sum_{i=k}^{n} \frac{\beta_i^2}{(\sigma_i^2 + \lambda)^2} - s^2 \sum_{i=1}^{k-1} \frac{1}{(\sigma_i^2 + \lambda)^2} \, . \tag{2.2}$$

As λ increases from zero, this function is monotonically increasing, and we denote the first zero by λ_{hat} and the corresponding solution vector x_{hat}.

3. Convergence for the Tikhonov parameter choice. We know that we cannot, in general, compute the optimal Tikhonov parameter. How far do we stray from the optimal vector when we use a nonoptimal parameter? The following theorem bounds the relative distance between the optimal solution and the computed one.

THEOREM 3.1. *Let λ_{opt} be the optimal parameter for the Tikhonov family (i.e., the (generally uncomputable) one that produces the solution closest to x_{true}). Then for any value of λ,*

$$\frac{\|x_{tik}(\lambda_{opt}) - x_{tik}(\lambda)\|}{\|x_{tik}(\lambda_{opt})\|} \le \frac{|\lambda_{opt} - \lambda|}{\sigma_n^2 + \lambda}\,.$$

Proof. The result follows from the computation

$$\begin{aligned}\|x_{tik}(\lambda_{opt}) - x_{tik}(\lambda)\|^2 &= \sum_{i=1}^{n} \left(\frac{\beta_i \sigma_i}{\sigma_i^2 + \lambda_{opt}} - \frac{\beta_i \sigma_i}{\sigma_i^2 + \lambda} \right)^2 \\ &= \sum_{i=1}^{n} \beta_i^2 \sigma_i^2 \left(\frac{\lambda - \lambda_{opt}}{(\sigma_i^2 + \lambda_{opt})(\sigma_i^2 + \lambda)} \right)^2 \\ &\le \frac{(\lambda - \lambda_{opt})^2}{(\sigma_n^2 + \lambda)^2} \sum_{i=1}^{n} \left(\frac{\beta_i \sigma_i}{\sigma_i^2 + \lambda_{opt}} \right)^2 \\ &= \frac{(\lambda - \lambda_{opt})^2}{(\sigma_n^2 + \lambda)^2} \|x_{tik}(\lambda_{opt})\|^2\,. \qquad \square\end{aligned}$$

Our algorithm for choosing the regularization parameter also behaves well as the size of the observation noise is decreased.

THEOREM 3.2. *If we choose the parameter k so that $\epsilon_i \approx \beta_i$ for $i \ge k$, then in the limit as the standard deviation s of the noise converges to zero, the solution x_{hat} produced by our algorithm converges to the correct discrete solution x_{true}.*

Proof. As the standard deviation of the noise goes to zero, the value k increases to $n+1$, and the solution to $\hat{g}(\lambda) = 0$ becomes $\lambda = 0$, as desired. Thus, as the noise goes to zero, our solution converges to the noise-free solution. □

4. An alternate family of solutions. We have studied how the regularization parameter might be chosen for one family of solutions, the Tikhonov solutions, which take the form

$$x_{tik} = \sum_{i=1}^{n} \frac{\beta_i \sigma_i}{\sigma_i^2 + \lambda} v_i\,.$$

A similar algorithm can be found for other solution families, and in this section we consider the family

$$x_{alt} = \sum_{i=1}^{n} \frac{\beta_i}{\sigma_i + \lambda} v_i\,.$$

This family was proposed by Franklin [4] for Hermitian positive definite A and is also associated with Lavrentiev [10, p.107]. Ekstrom and Rhoads [2] discussed the use of the algorithm for convolution problems symmetrized by reordering, and this method was also considered by Cullum [1].

In his Regularization Tool Package for Matlab [9], Hansen includes a function `dsvd` that can be used to apply the method to general problems. In this more general context, there is more than one interpretation. The solution x_{alt} satisfies the regularized equation

$$(A + \lambda U V^*)x = b\,.$$

However, it may be more intuitive to interpret the family as a set of filter factors [10, sect. 4.2]

$$\frac{\sigma_i}{\sigma_i + \lambda},$$

multiplying the corresponding terms in the least squares solution

$$\sum_{i=1}^{n} \frac{\beta_i}{\sigma_i} v_i\,.$$

To choose the parameter λ, we mimic the procedure in section 2: we naively try to minimize the distance between our solution and the true one:

$$\min_{\lambda} \|x_{alt} - x_{true}\|^2 \equiv \min_{\lambda} f(\lambda)\,.$$

Using the singular value representation, we see that

$$f(\lambda) = \sum_{i=1}^{n} \left[\frac{\beta_i}{\sigma_i + \lambda} - \frac{\beta_i - \epsilon_i}{\sigma_i} \right]^2 .$$

Setting the derivative equal to zero yields

$$\begin{aligned} 0 = g(\lambda) \equiv \frac{1}{2} f'(\lambda) &= -\sum_{i=1}^{n} \left[\frac{\beta_i}{\sigma_i + \lambda} - \frac{\beta_i - \epsilon_i}{\sigma_i} \right] \left[\frac{\beta_i}{(\sigma_i + \lambda)^2} \right] \\ &= \sum_{i=1}^{n} \frac{\beta_i^2 \lambda}{\sigma_i(\sigma_i + \lambda)^3} - \sum_{i=1}^{n} \frac{\beta_i \epsilon_i}{\sigma_i(\sigma_i + \lambda)^2}\,. \end{aligned}$$

Again, the first summation in this last expression is computable. The second is not, because the observation noise values ϵ_i are unknown, but the terms for $i \approx n$ dominate, and for these $\epsilon_i \approx \beta_i$, so our approximate function becomes

$$\hat{g}(\lambda) \equiv \sum_{i=1}^{n} \frac{\beta_i^2 \lambda}{\sigma_i(\sigma_i + \lambda)^3} - \sum_{i=k}^{n} \frac{\beta_i^2}{\sigma_i(\sigma_i + \lambda)^2} - \mathcal{E} \left(\sum_{i=1}^{k-1} \frac{\beta_i \epsilon_i}{\sigma_i(\sigma_i + \lambda)^2} \right)$$

for a suitable index k that depends on the standard deviation of the noise. Finding the zero of the function

$$\hat{g}(\lambda) \equiv \sum_{i=1}^{n} \frac{\beta_i^2 \lambda}{\sigma_i(\sigma_i + \lambda)^3} - \sum_{i=k}^{n} \frac{\beta_i^2}{\sigma_i(\sigma_i + \lambda)^2} - s^2 \sum_{i=1}^{k-1} \frac{1}{\sigma_i(\sigma_i + \lambda)^2} \tag{4.1}$$

yields an approximation to the optimal value of λ.

We have a bound for the relative distance between the optimal solution and the computed one similar to the Tikhonov case.

THEOREM 4.1. *Let λ_{alt} be the optimal parameter for the alternate family (i.e., the one that produces the solution closest to x_{true}). Then for any value of λ,*

$$\frac{\|x_{alt}(\lambda_{alt}) - x_{alt}(\lambda)\|}{\|x_{alt}(\lambda_{alt})\|} \leq \frac{|\lambda_{alt} - \lambda|}{\sigma_n^2 + \lambda}.$$

Proof. The result follows from a computation similar to that in the proof of Theorem 3.1. □

Again, we can show that the solution converges to the true solution as the observation noise goes to zero.

THEOREM 4.2. *If we choose the parameter k so that $\epsilon_i \approx \beta_i$ for $i \geq k$, then in the limit as the standard deviation s of the noise converges to zero, the solution x_{hat} produced by our algorithm converges to x_{true}.*

Proof. The proof is the same as above. □

5. Algorithmic notes. The standard deviation s of the noise is not assumed to be known, so we estimate it using the last $\max(m-n, 10)$ components of the right-hand side. If $|b_n| > 3.5s$, then we choose $k = n$. Otherwise we use a T-test to determine the index k. We choose k as the smallest index, among the values $n-9, n-14, \ldots$, for which a T-test with 0.05 significance level indicates that the sequence $\beta_k, \ldots, \beta_n$ has zero mean. If the mean of the noise-free sequence is likely to be near zero, then this test would not be appropriate, but many alternatives are available. One would be to use the Mann–Whitney Test, a nonparametric test to determine whether two independent groups of sampled data are taken from the same underlying distribution without making assumptions on the distribution.

A root of either function (2.2) or (4.1) can be found using standard algorithms (e.g., `fzero` in Matlab). Since $\hat{g}(0) < 0$ for both functions, we can find a lower bound on the root by searching $s, s/10, s/100, \ldots$ for a negative function value. The simple strategy of searching $100s, 1000s, \ldots$ has proved effective in finding a value for which $\hat{g}$ is positive, thus providing the root finder with an initial interval containing the root.

6. Performance of the algorithms. The ideas of the previous sections were tested using two sets of test problems. In the first, the 200×200 matrix was diagonal, with entries ranging between 1 and 10^{-5}, evenly spaced on a log scale. The true solution was assumed to be the vector with elements evenly spaced between 1.0 and 0.9, and 100 sets of random noise were generated for the right-hand side. The value of the standard deviation of the noise was not made available to the algorithms; instead, we estimated it as in section 5. We generated solutions using the Tikhonov and the alternate method and calculated the distance between these computed solutions and the exact noise-free solution, tabulating the relative x-error $\|x - x_{true}\|/\|x_{true}\|$. Then we calculated the *optimal* Tikhonov and alternate solutions, the ones corresponding to the parameter values that minimize the distance to the noise-free solution. These optimal solutions, of course, cannot be computed in practical situations since the noise-free solution is unknown, but the results tell us how far we are from optimal. We also compared our results with three other methods:

1. We computed the the Tikhonov parameter by minimizing the generalized cross-validation (GCV) function using Matlab's `fmin` with tolerance 1.0e-07. In some sense this is an unfair comparison, since GCV aims to minimize the residual norm, not the x-error.

Table 6.1
Relative errors in experiments on a diagonal matrix of size 200.

Standard dev.	Optimal	Computed	Optimal	Computed	GCV	Hanke–Raus	Discrep.
of noise	Tikhonov	Tikhonov	alternate	alternate	Tikhonov	Tikhonov	Tikhonov
Mean							
1.0e-03	6.17e-01	7.05e-01	6.59e-01	1.12e+00	8.99e-01	8.20e-01	6.80e-01
1.0e-04	4.34e-01	4.61e-01	4.63e-01	4.94e-01	8.37e-01	6.82e-01	5.16e-01
1.0e-05	1.75e-01	2.11e-01	1.79e-01	1.97e-01	7.70e-01	4.97e-01	3.09e-01
1.0e-06	2.18e-02	6.52e-02	2.19e-02	5.34e-02	7.32e-01	4.58e-01	1.38e-01
Median							
1.0e-03	6.17e-01	6.43e-01	6.62e-01	6.71e-01	9.01e-01	8.21e-01	6.71e-01
1.0e-04	4.37e-01	4.57e-01	4.66e-01	4.74e-01	8.38e-01	6.83e-01	5.16e-01
1.0e-05	1.73e-01	2.10e-01	1.76e-01	1.95e-01	7.72e-01	4.83e-01	3.07e-01
1.0e-06	2.13e-02	4.10e-02	2.13e-02	3.05e-02	7.35e-01	4.63e-01	1.37e-01
Maximum							
1.0e-03	6.46e-01	4.95e+00	6.88e-01	7.34e+00	9.21e-01	8.49e-01	1.62e+00
1.0e-04	4.78e-01	9.86e-01	5.11e-01	9.93e-01	8.68e-01	7.52e-01	5.75e-01
1.0e-05	2.42e-01	2.82e-01	2.60e-01	2.68e-01	7.99e-01	5.89e-01	3.85e-01
1.0e-06	3.41e-02	1.61e-01	3.41e-02	1.40e-01	7.53e-01	4.71e-01	1.83e-01

2. We also compared our results with those of the Tikhonov algorithm of Hanke and Raus [7], which chooses the parameter by minimizing

$$f(\lambda) = \sqrt{1 + 1/\lambda}\sqrt{r_1^T(\lambda) r_0(\lambda)}\,,$$

where

$$\begin{aligned} x_0 &= (A^*A + \lambda I)^{-1} A^* b\,, \\ r_0(\lambda) &= b - A x_0\,, \\ x_1 &= (A^*A + \lambda I)^{-1} A^* r_0 + x_0\,, \\ r_1(\lambda) &= b - A x_1\,. \end{aligned}$$

3. We used our estimate of the standard deviation of the noise to apply the discrepancy principle [14].

The results are summarized in Table 6.1. Several trends are apparent. First, the average relative x-errors in the solutions computed by our algorithm are within a factor of 2 of the average relative x-errors for the optimal parameter values. Second, for large noise in the observations, the Tikhonov solution is on average closer to the true solution, but for small noise the alternate algorithm does somewhat better than Tikhonov. Third, the Tikhonov solutions computed by our algorithm are on average better than the GCV Tikhonov solutions and the Hanke–Raus solutions over the full range of noise values, and for small noise, the alternate solutions are better too. Our Tikhonov solutions are better than the discrepancy principle solutions except for a noise level of 10^{-3}.

The trends in the medians are similar to those of the averages, except that our values are always better than the discrepancy principle. The maximum relative errors show that only in the small number of cases in which the standard deviation of the error fails to be computed accurately are the GCV and Hanke–Raus solutions much better than our solutions.

Our algorithm for choosing the parameter k tested values in increments of 5. (See section 5.) Results are relatively insensitive to this increment: for experiments with noise level 1.0e-03 and increments of 1, 5, or 10, for example, the mean for

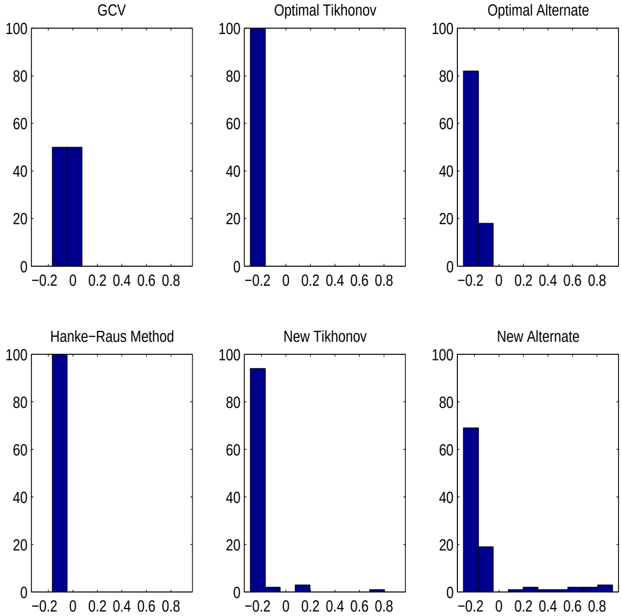


FIG. 6.1. *Histograms of the relative errors in the solutions computed for the diagonal matrix problem with standard deviation of the observation noise equal to* 1.0e-03. *The horizontal axis indicates the log of the relative error. The bars have height equal to the number of test problems yielding errors in that range.*

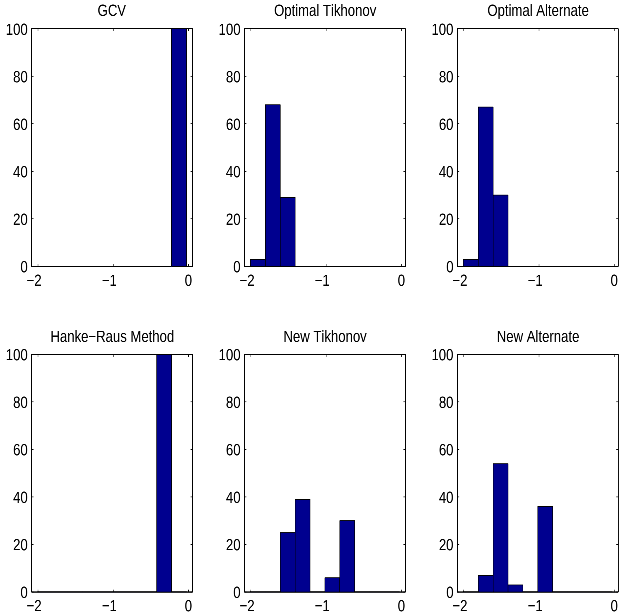


FIG. 6.2. *Histograms of the relative errors in the solutions computed for the diagonal matrix problem with standard deviation of the observation noise equal to* 1.0e-06. *The horizontal axis indicates the log of the relative error. The bars have height equal to the number of test problems yielding errors in that range.*

Table 6.2
Relative errors in experiments on a helioseismatic matrix of size 212×100.

Standard dev. of noise	Optimal Tikhonov	Computed Tikhonov	Optimal alternate	Computed alternate	GCV Tikhonov	Hanke–Raus
Mean values:						
1.0e-02	5.83e-01	5.75e+04	6.46e-01	5.08e+04	8.87e-01	8.63e-01
1.0e-04	4.58e-01	5.75e+02	4.91e-01	5.08e+02	6.60e-01	5.88e-01
1.0e-06	3.54e-01	3.49e+00	3.71e-01	3.35e+00	5.71e-01	5.71e-01
Median values:						
1.0e-02	5.92e-01	6.02e-01	6.49e-01	6.52e-01	8.87e-01	8.63e-01
1.0e-04	4.58e-01	4.90e-01	4.93e-01	4.97e-01	6.61e-01	5.88e-01
1.0e-06	3.54e-01	3.62e-01	3.73e-01	3.75e-01	5.71e-01	5.71e-01
Max values:						
1.0e-02	6.11e-01	4.55e+06	6.88e-01	4.00e+06	8.92e-01	8.66e-01
1.0e-04	4.95e-01	4.55e+04	5.18e-01	4.00e+04	6.68e-01	5.88e-01
1.0e-06	3.70e-01	3.10e+02	3.95e-01	2.96e+02	5.74e-01	5.74e-01

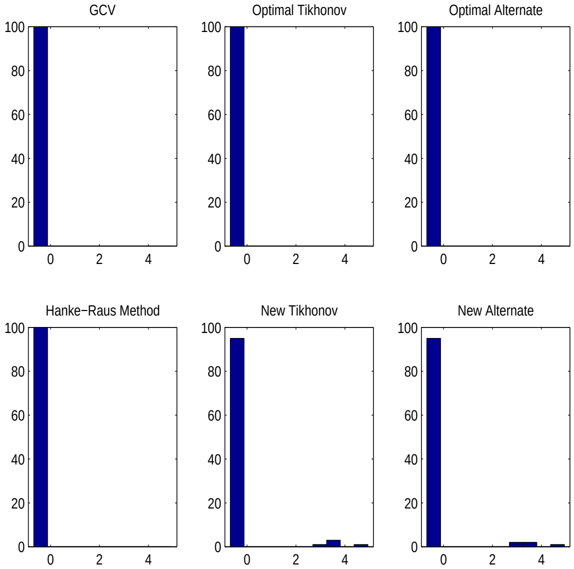


Fig. 6.3. *Histograms of the relative errors in the solutions computed for the helioseismatic matrix problem with standard deviation of the observation noise equal to* 1.0e-04. *The horizontal axis indicates the log of the relative error. The bars have height equal to the number of test problems yielding errors in that range.*

the computed Tikhonov values was between 7.01e-01 and 7.56e-01, while the median changed by at most 3 in the third significant digit. Similarly, the mean for the computed alternate values was between 1.06 and 1.34, while the median changed by at most 1 in the third significant digit.

Histograms of the relative errors are presented in Figures 6.1 and 6.2.

The second experiment used the inverse helioseismatic data of Hansen (`helio.mat`, taken from the Regularization Tool Package homepage [9]). The problem is an integral equation of the first kind modeling internal rotation of the sun as a function of radius. The matrix A of size 212×100 and the true solution x were obtained from there, and random observation noise was added as before. The right-hand side values had a mean close to zero, so a rather primitive scheme was used to determine k; it was determined so that the values b_j for $j > k$ were not larger than 3.5 times the estimated standard deviation. The results (Table 6.2) show that the median relative x-errors

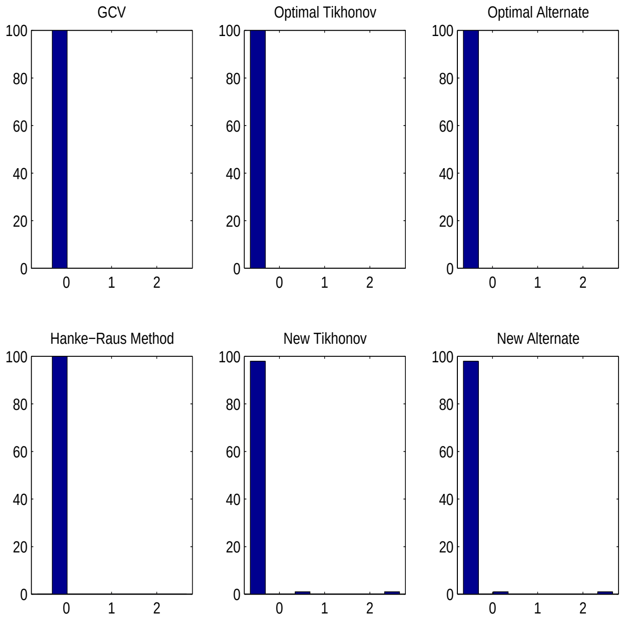


Fig. 6.4. *Histograms of the relative errors in the solutions computed for the helioseismatic matrix problem with standard deviation of the observation noise equal to* 1.0e-06*. The horizontal axis indicates the log of the relative error. The bars have height equal to the number of test problems yielding errors in that range.*

are at most 1.1 times as large as the optimal and at most 0.8 times the GCV values or the Hanke–Raus values. The trends are similar to the diagonal matrix problem: when the value of k is estimated well, the new algorithms perform much better than GCV and Hanke–Raus. However, since the k estimation problem is more difficult with this right-hand side, the mean and maximum values of the relative errors are not well behaved.

Still, the histograms of the relative errors presented in Figures 6.3 and 6.4 show that the new algorithms can be expected to produce much better results than GCV or Hanke–Raus when the errors are small enough that k is easily estimated.

7. Conclusions.

We have proposed a method for choosing a regularization parameter that approximately minimizes the Euclidean distance between the computed solution and the noise-free solution, and we have demonstrated by numerical experiments that it produces solutions quite close to optimal.

We have demonstrated the use of these methods of parameter choice when the SVD of the matrix A can be explicitly computed. If the problem is too large for this to be practical, the ideas could be used in conjunction with iterative methods by applying them in the subspace generated by the iteration. For example, the SVD of the reduced matrix produced by a GMRES iteration could be substituted for the SVD of the full matrix. The effectiveness of this general methodology is discussed in [13].

Acknowledgments. I am grateful for the hospitality provided by Professor Walter Gander and the Departement Informatik, ETH Zürich, Switzerland, which enabled this work to be completed. The manuscript benefited greatly from helpful comments by Misha E. Kilmer, Per Christian Hansen, Bert W. Rust, and the referees.

REFERENCES

[1] J. Cullum, *Ill-posed deconvolutions: Regularization and singular value decompositions*, in Proceedings of the 19th IEEE Conference on Decision and Control, Albuquerque, NM, 1980, pp. 29–35.

[2] M. P. Ekstrom and R. L. Rhoads, *On the application of eigenvector expansions to numerical deconvolution*, J. Comput. Phys., 14 (1974), pp. 319–340.

[3] H. W. Engl and H. Gfrerer, *A posteriori parameter choice for general regularization methods for solving linear ill-posed problems*, Appl. Numer. Math., 4 (1988), pp. 395–417.

[4] J. N. Franklin, *Minimum principles for ill-posed problems*, SIAM J. Math. Anal., 9 (1978), pp. 638–650.

[5] H. Gfrerer, *An a posteriori parameter choice for ordinary and iterated Tikhonov regularization of ill-posed problems leading to optimal convergence rates*, Math. Comp., 49 (1987), pp. 507–522.

[6] G. Golub, M. Heath, and G. Wahba, *Generalized cross-validation as a method for choosing a good ridge parameter*, Technometrics, 21 (1979), pp. 215–223.

[7] M. Hanke and T. Raus, *A general heuristic for choosing the regularization parameter in ill-posed problems*, SIAM J. Sci. Comput., 17 (1996), pp. 956–972.

[8] P. C. Hansen, *Analysis of discrete ill-posed problems by means of the L-curve*, SIAM Rev., 34 (1992), pp. 561–580.

[9] P. C. Hansen, *Regularization tools: A Matlab package for analysis and solution of discrete ill-posed problems*, Numer. Algorithms, 6 (1994), pp. 1–35; also available online from http://www.imm.dtu.dk/documents/users/pch/Regutools/regutools.html.

[10] P. C. Hansen, *Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion*, SIAM, Philadelphia, 1997.

[11] A. E. Hoerl and R. W. Kennard, *Ridge regression: Biased estimation for nonorthogonal problems*, Technometrics, 12 (1970), pp. 55–67.

[12] J. Kay, *Asymptotic comparison factors for smoothing parameter choices in regression problems*, Statist. Probab. Lett., 15 (1992), pp. 329–335.

[13] M. E. Kilmer and D. P. O'Leary, *Choosing regularization parameters in iterative methods for ill-posed problems*, SIAM J. Matrix Anal. Appl., 22 (2001), pp. 1204–1221.

[14] V. A. Morozov, *On the solution of functional equations by the method of regularization*, Soviet Math. Dokl., 7 (1966), pp. 414–417. Cited in [10].

[15] J. E. Pierce and B. W. Rust, *Constrained least squares interval estimation*, SIAM J. Sci. Statist. Comput., 6 (1985), pp. 670–683.

[16] T. Raus, *The principle of the residual in the solution of ill-posed problems with nonselfadjoint operator*, Uchen. Zap. Tartu Gos. Univ., 715 (1985), pp. 12–20 (in Russian). Cited in [7].

[17] B. W. Rust, *Truncating the Singular Value Decomposition for Ill-Posed Problems*, Tech. Rep. NISTIR 6131, National Institute of Standards and Technology, U.S. Department of Commerce, Gaithersburg, MD, 1998.

SIAM J. SCI. COMPUT.
Vol. 23, No. 4, pp. 1172–1188

PRACTICAL CONSTRUCTION OF MODIFIED HAMILTONIANS*

ROBERT D. SKEEL† AND DAVID J. HARDY†

Abstract. One of the most fruitful ways to analyze the effects of discretization error in the numerical solution of a system of differential equations is to examine the "modified equations," which are equations that are exactly satisfied by the (approximate) discrete solution. These do not actually exist in general but rather are defined by an asymptotic expansion in powers of the discretization parameter. Nonetheless, if the expansion is suitably truncated, the resulting modified equations have a solution which is remarkably close to the discrete solution. In the case of a Hamiltonian system of ordinary differential equations, the modified equations are also Hamiltonian if and only if the integrator is symplectic. Evidence for the existence of a Hamiltonian for a particular calculation is obtained by calculating modified Hamiltonians and monitoring how well they are conserved. Also, energy drifts caused by numerical instability are better revealed by evaluating modified Hamiltonians. Doing this calculation would normally be complicated and highly dependent on the details of the method, even if differences are used to approximate derivatives. A relatively simple procedure is presented here, nearly independent of the internal structure of the integrator, for obtaining highly accurate estimates for modified Hamiltonians. As a bonus of the method of construction, the modified Hamiltonians are exactly conserved by a numerical solution in the case of a quadratic Hamiltonian.

Key words. symplectic, Hamiltonian, modified equation, integrator, backward error, numerical

AMS subject classifications. 65P10, 65L05

PII. S106482750138318X

1. Introduction.

One of the most fruitful ways to analyze the effects of discretization error in the numerical solution of differential equations is to examine the "modified equations," which are equations that are exactly satisfied by the (approximate) discrete solution. These do not actually exist (in general) but rather are defined by an asymptotic expansion in powers of the discretization parameter. Nonetheless, if the expansion is suitably truncated, the resulting modified equations have a solution which is remarkably close to the discrete solution [9] (over relatively short time intervals). In the case of a Hamiltonian system of ordinary differential equations, the modified equations are also Hamiltonian if and only if the integrator is symplectic. The existence of a modified (or "shadow" [4]) Hamiltonian is an indicator of the validity of statistical estimates calculated from long time integration of chaotic Hamiltonian systems [20]. In addition, the modified Hamiltonian is a more sensitive indicator than the original Hamiltonian of drift in the energy (caused by instability). Evidence for the existence of a Hamiltonian for a particular calculation can be obtained by calculating modified Hamiltonians and monitoring how well they are conserved. Doing this calculation would normally be complicated and highly dependent on the details of the method, even if differences are used to approximate higher derivatives. Presented here is a relatively simple procedure, nearly independent of the internal structure of the integrator, for obtaining highly accurate estimates for modified Hamiltonians.

Consider a step by step numerical integrator $x^{n+1} = \Phi_h(x^n)$ which evolves an

*Received by the editors January 4, 2001; accepted for publication (in revised form) June 22, 2001; published electronically November 7, 2001.

http://www.siam.org/journals/sisc/23-4/38318.html

†Department of Computer Science and Beckman Institute, University of Illinois, 1304 West Springfield Avenue, Urbana, Illinois 61801-2987 (skeel@cs.uiuc.edu, dhardy@cs.uiuc.edu). The work of the first author was supported in part by NSF grants DMS-9971830 and DBI-9974555, and NIH grant P41RR05969, and completed while visiting the Mathematics Department, University of California, San Diego.

approximate solution $x^n \approx x(nh)$ for a system of ordinary differential equations $\dot{x} = f(x)$. For such discrete solutions, there exists modified equations $\dot{x}_h = f_h(x_h)$ defined by an asymptotic expansion such that *formally* the numerical solution $x^n = x_h(nh)$. The modified right-hand-side function f_h is defined uniquely by postulating an asymptotic expansion $f_0 + hf_1 + h^2 f_2 + \cdots$ in powers of h, substituting this into the equations for the numerical solution, expanding in powers of h, and equating coefficients [17, 27, 6, 24]. The asymptotic expansion does not generally converge except for (reasonable integrators applied to) linear differential equations.

A Hamiltonian system is of the form

$$\dot{x}(t) = JH_x(x(t)), \qquad J = \begin{bmatrix} 0 & I \\ -I & 0 \end{bmatrix},$$

for some Hamiltonian $H(x)$, $x = [q^{\mathrm{T}}, p^{\mathrm{T}}]^{\mathrm{T}}$. The modified equation for an integrator Φ_h applied to this system is Hamiltonian; i.e., $f_h = JH_{h,x}(x)$ for some modified Hamiltonian $H_h(x)$ if and only if the integrator is symplectic [25, 22]. The integrator is symplectic if $\Phi_{h,x}(x)J^{\mathrm{T}}\Phi_{h,x}(x) \equiv J$. There is theoretical [18, 2, 8, 20] and empirical evidence that

$$x_n = x_h(nh) + \text{ very small error}$$

for a very long time, where x_h is the solution for a suitably truncated Hamiltonian H_h. In what follows we assume that H_h is such a Hamiltonian and *we neglect the very small error.*

If we plot total energy as a function of time for a numerical integrator such as leapfrog/Störmer/Verlet applied to a molecular dynamics simulation, we get a graph like Figure 3. What we observe are large fluctuations in the original Hamiltonian, as the trajectory moves on a hypersurface of a constant modified Hamiltonian. A small drift or jump in the energy would be obscured by the fluctuations. A plot of a modified Hamiltonian might be more revealing. As an example, the plots of modified Hamiltonians in Figure 4 already show a clear rise in energy in a 400-step simulation. This indicates that plots of suitable modified Hamiltonians can make it easier to test integration algorithms for instability and programming bugs. Details of this and other numerical tests are given in section 2. Before continuing, it is worth emphasizing that the concern of this paper is stability monitoring—not the monitoring and enhancement of accuracy, as in [4] and [16].

The goal is to construct an approximate modified Hamiltonian

$$H_{[2k]}(q,p) = H_h(q,p) + O(h^{2k})$$

that can be conveniently assembled from quantities, such as forces and energies, already available from the numerical integration. These requirements do *not* uniquely determine $H_{[2k]}$. We consider the special separable Hamiltonian $H(q,p) = \frac{1}{2}p^{\mathrm{T}}M^{-1}p + U(q)$ for which the system is of the form

$$\dot{q} = M^{-1}p, \qquad \dot{p} = F(q) \overset{\text{def}}{=} -U_q(q).$$

A "brute force" approach would be to determine an asymptotic expansion for H_h and of the quantities available for making an approximation and then to form a suitable linear combination of the latter. By such a matching of asymptotic expansions one could derive the following modified Hamiltonians for the leapfrog method:

$$H_{[2]}(q^n,p^n) = H(q^n,p^n) - \underbrace{\frac{1}{8}h^2(F^n)^{\mathrm{T}}M^{-1}F^n}_{\text{why?}}, \tag{1.1}$$

$$H_{[4]}(q^n,p^n) = H(q^n,p^n) + \frac{1}{4}\delta^2 U^n + \frac{1}{6}h(p^n)^{\mathrm{T}}M^{-1}\mu\delta F^n + \frac{5}{24}h^2(F^n)^{\mathrm{T}}M^{-1}F^n + \frac{1}{12}h^2(F^n)^{\mathrm{T}}M^{-1}\delta^2F^n, \tag{1.2}$$

$$H_{[6]}(q^n,p^n) = H(q^n,p^n) + \frac{11}{60}\delta^2 U^n + \frac{1}{10}h(p^n)^{\mathrm{T}}M^{-1}\mu\delta F^n + \frac{17}{120}h^2(F^n)^{\mathrm{T}}M^{-1}F^n + \frac{1}{10}h^2(F^n)^{\mathrm{T}}M^{-1}\delta^2F^n + \frac{1}{60}h^2(\mu\delta F^n)^{\mathrm{T}}M^{-1}\mu\delta F^n - \frac{1}{240}h^2(\delta^2F^n)^{\mathrm{T}}M^{-1}\delta^2F^n. \tag{1.3}$$

Here a superscript n denotes evaluation at q^n, the centered difference operator is defined by $\delta w^n = w^{n+1/2} - w^{n-1/2}$, the averaging operator is defined by $\mu w^n = \frac{1}{2}w^{n+1/2} + \frac{1}{2}w^{n-1/2}$, and values $q^{n\pm1}$, $p^{n\pm1}$ are defined in terms of q^n, p^n by the leapfrog method:

$$p^{n\pm1/2} = p^n \pm \frac{h}{2}F^n, \quad q^{n\pm1} = q^n \pm hM^{-1}p^{n\pm1/2}, \quad p^{n\pm1} = p^{n\pm1/2} \pm \frac{h}{2}F^{n\pm1}.$$

Thus, in principle, analytical expressions for the $H_{[2k]}(q,p)$ could be produced.

Note. The terms of order h^2 in $H_{[4]}$ and $H_{[6]}$ are different; but if expanded in powers of h, they agree up to $O(h^4)$.

An easier and more elegant construction is presented in sections 3–5. The technique is developed only for splitting methods. It is likely that a similar construction is also possible for symplectic implicit Runge–Kutta methods. The idea is to add a new position variable and a conjugate momentum variable to get an extended Hamiltonian $\bar{H}_h(y)$ which is homogeneous of order 2. For such a Hamiltonian $\bar{H}_h(y_h(t)) \equiv \frac{1}{2}\dot{y}_h(t)^{\mathrm{T}}\bar{J}y_h(t)$. Thus the problem is reduced to that of forming an approximation for $y_h(t)$ using the numerical solution of an extended Hamiltonian system. It is plausible that such a construction might be useful theoretically due to the availability of robust approximation techniques.

Equation (1.1) for $H_{[2]}$ contains an h^2 term which is not needed for achieving second order accuracy. Similarly, the last terms of $H_{[4]}$ and $H_{[6]}$ are not needed for fourth and sixth order, respectively. They are present because the given "truncations" $H_{[2k]}$ are designed to exactly conserve energy for the numerical solution when H is quadratic. (See sections 4.1 and 5.1.) This is a very useful property because typical applications, including molecular dynamics, are dominated by harmonic motion. The existence of a modified Hamiltonian that is exactly conserved for a quadratic Hamiltonian is noted in [19, Eq. (4.7b)], and the search for similar methods having this property was central to the results of this paper. For a quadratic Hamiltonian the modified Hamiltonian H_h actually exists (if h is not too large), but $H_{[2k]} \neq H_h$. (A simple derivation of H_h for the one-dimensional case is given in [24].) It should also be noted that the Hamiltonians $H_{[2k]}$ will not detect numerical instability in the case of quadratic Hamiltonians H.

2. Numerical experiments. The approximate modified Hamiltonians $H_{[2k]}(x)$, $k = 1, 2, 3, 4$, defined by (5.3), (4.4), (5.2), and (4.3) are computed and plotted as

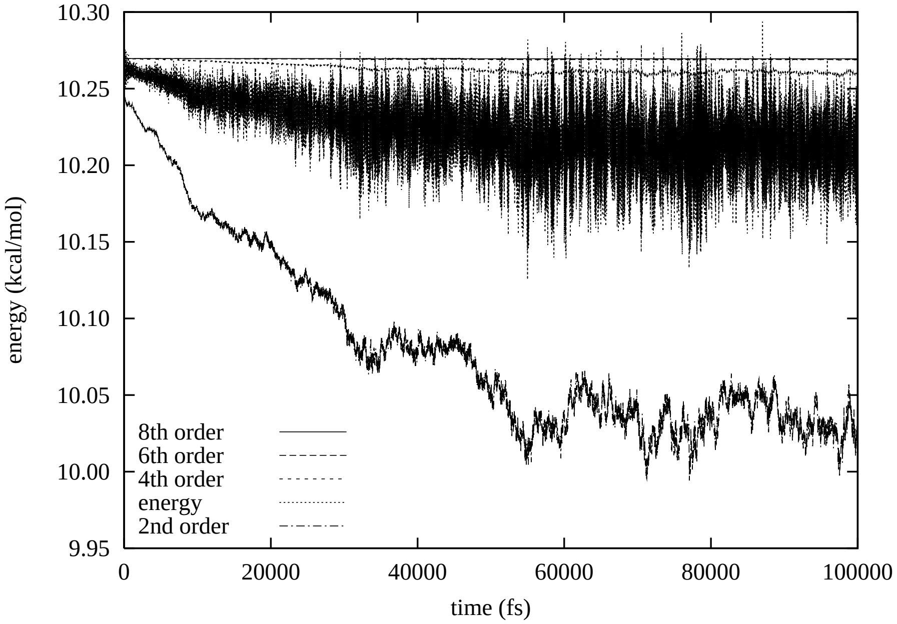


FIG. 1. *Energy and various truncations of the modified Hamiltonian for decalanine.*

functions of time for numerical solutions generated by the leapfrog method. The unmodified Hamiltonians are those of classical molecular dynamics. The testing was done with a molecular dynamics program written by the second author, which is compatible with NAMD [12] but limited in features to facilitate algorithm testing.

The first couple of experiments demonstrate the quality of the modified Hamiltonians. The test problem is a 66-atom peptide, decalanine, in a vacuum [1]. The force field parameters are those of CHARMM 22 for proteins [15, 14] without cutoffs for nonbonded forces.

Figure 1 shows a plot of the Hamiltonian and second, fourth, sixth, and eighth order modified Hamiltonian approximations vs. time for 100,000 fs (femtoseconds) for a step size $h = 1\,\mathrm{fs}$ with the energy sampled every eighth step. The level graph at the top is the eighth order truncation, the one just barely beneath it is sixth order, and the one under that is fourth order. The greatly fluctuating graph is the energy itself, and the undulating one well below it is the second order truncation. Note how well the asymptotic theory holds for the higher order truncations—one could not obtain such flat plots by simply smoothing the original Hamiltonian.

Figure 2 expands the vertical scale to show fluctuations in the eighth, sixth, and fourth order truncations of modified Hamiltonians.

An explanation is in order concerning the initial drop in energy. The initial velocities are zero, so integrating backward in time is the same as integrating forward. Hence the first part of the trajectory is simply the second half of a very unusual fluctuation. In other words, the initial conditions are atypical, i.e., not properly equilibrated (with respect to the original Hamiltonian). This is particularly well revealed by the plot of the second order truncation.

The remaining experiments demonstrate the ability of modified Hamiltonians to detect instability. The test problem is a set of 125 water molecules harmonically

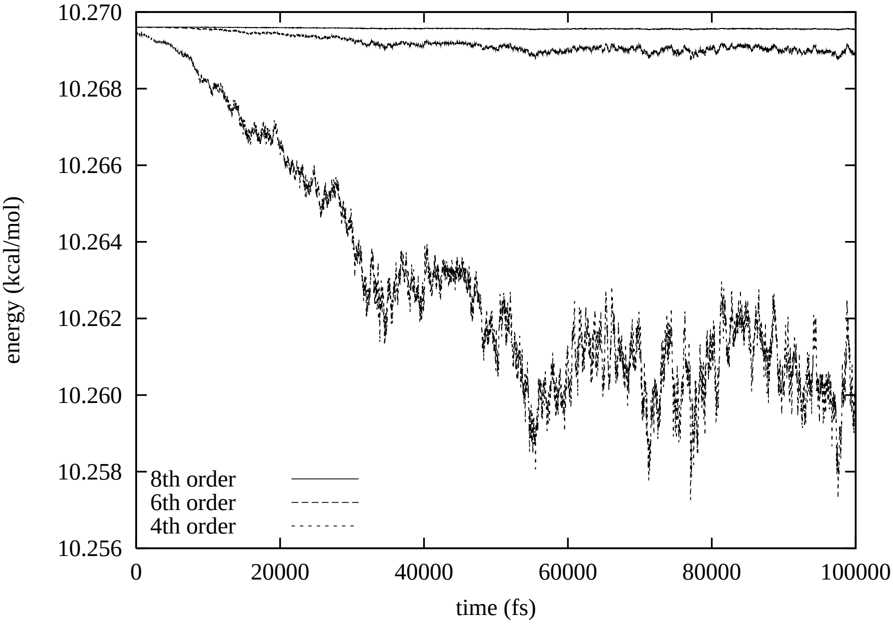


FIG. 2. *A closer look at higher order truncations for decalanine.*

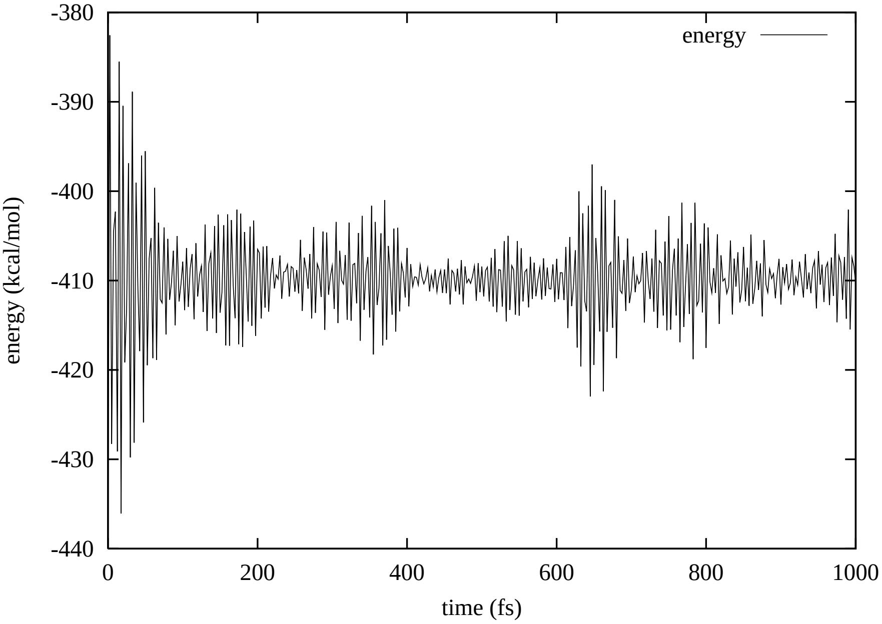


FIG. 3. *Energy for flexible water with step size* 2.5 *fs.*

restrained to a 10 Å-radius sphere. The water is based on the TIP3P model [11] without cutoffs and with flexibility incorporated by adding bond stretching and angle bending harmonic terms (cf. [13]).

Figure 3 shows a plot of the energy vs. time for 1,000 fs for a step size $h = 2.5\,\mathrm{fs}$ with the energy sampled every step. Note that the large fluctuations make it difficult

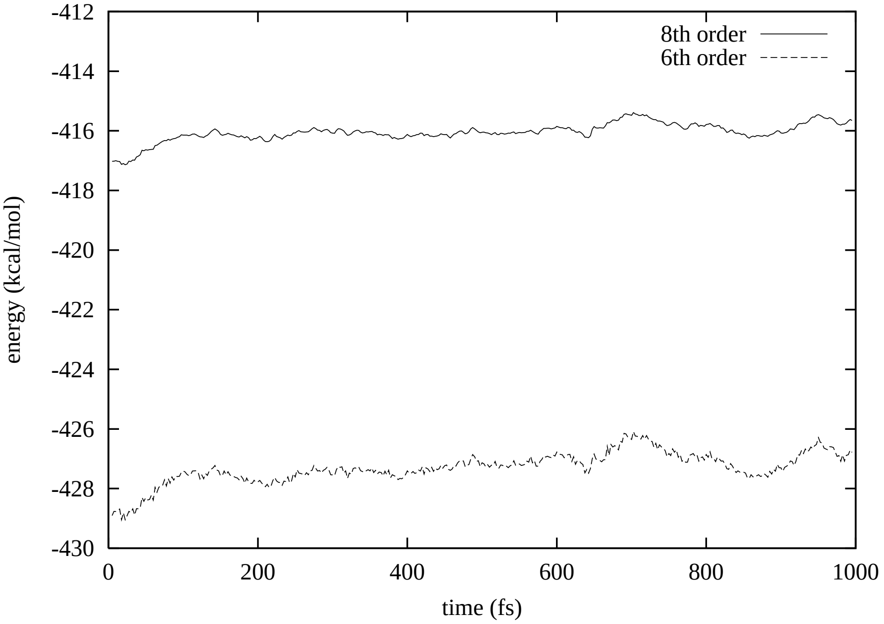


FIG. 4. *Sixth and eighth order truncations with step size* 2.5 *fs.*

to determine whether or not there is energy drift.

Figure 4 shows a plot of the sixth and eighth order modified Hamiltonians for the same step size $h = 2.5\,\mathrm{fs}$. An upward energy drift is now obvious. The second and fourth order approximations are not shown because neither of them was as flat. Normal mode analysis for this system [10] shows that the 250 fastest frequencies have periods in the range 9.8–10.2 fs and use of the formula in [24, p. 131] shows that a 2.5 fs step size is 30% of the effective period for discrete leapfrog dynamics. It is remarkable that the eighth order approximation is the flattest, even for such a large step size.

Figure 5 shows a plot of the sixth and eighth order modified Hamiltonians for step size $h = 2.15\,\mathrm{fs}$. There is no apparent upward drift of the energy. Theoretically, instability due to 4:1 resonance [23] should occur for the leapfrog method at $h \cdot \text{angular\_frequency} = \sqrt{2}$, which is in the range 2.2–2.3 fs for flexible water.

3. Augmenting the integrator. The integrator is augmented to make it homogeneous of order 1. This is motivated by the desire to extend results obtained for homogeneous linear mappings to affine mappings. Affine mappings can be reduced to homogeneous linear mappings through the use of homogeneous coordinates, in which the given set of coordinates is augmented by a scale factor, denoted here by α.

We assume that one step of size h for the given method applied to a system with Hamiltonian H is the composition of exact h-flows for Hamiltonian systems with Hamiltonians $H_1, H_2, \ldots, H_L$. Each $H_l(x)$ is assumed to be sufficiently smooth on some domain containing the infinite time trajectory. For example,

1. the leapfrog method for separable Hamiltonian systems $H(q,p) = K(p)+U(q)$ uses $L = 3$, $H_1(x) = \frac{1}{2}U(q)$, $H_2(x) = K(p)$, and $H_3(x) = \frac{1}{2}U(q)$;
2. the Rowlands method [21] for *special* separable Hamiltonian systems uses $H_1(x) = H_3(x) = \frac{1}{2}U(q) - \frac{1}{48}h^2U_q(q)^{\mathrm{T}}M^{-1}U_q(q)$ and $H_2(x) = \frac{1}{2}p^{\mathrm{T}}M^{-1}p$;

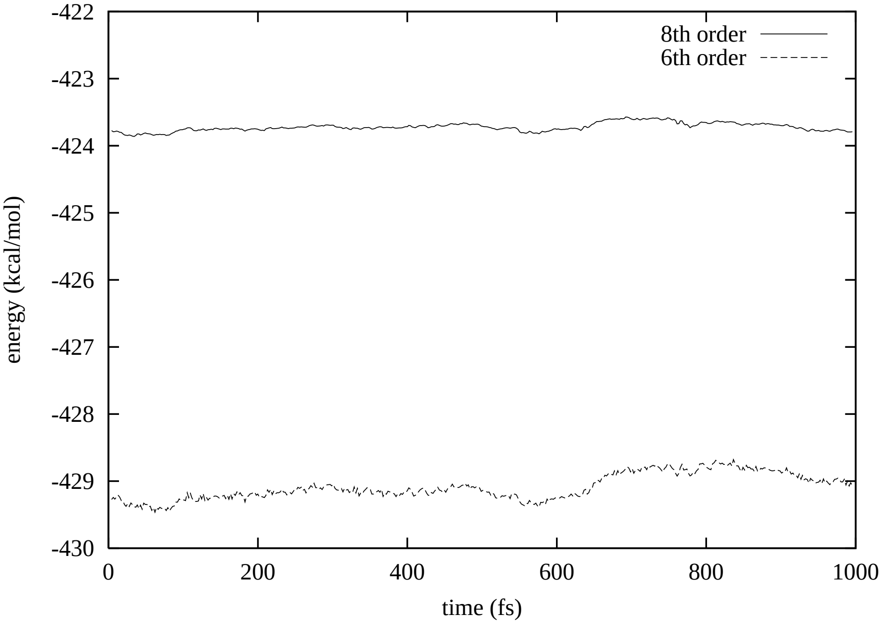


FIG. 5. *Sixth and eighth order truncations with step size* 2.15 *fs.*

3. double time-stepping [7, 26] uses $L = 5$, $H_1(x) = H_5(x) = \frac{1}{2}U^{\mathrm{slow}}(q) + \frac{1}{4}U^{\mathrm{fast}}(q)$, $H_2(x) = H_4(x) = \frac{1}{2}K(p)$, and $H_3(x) = \frac{1}{2}U^{\mathrm{fast}}(q)$,
4. *Molly* [5] does the same as double time-stepping except for the substitution of $U^{\mathrm{slow}}(\mathcal{A}(q))$ for $U^{\mathrm{slow}}(q)$, where $\mathcal{A}(q)$ is a local temporal averaging of q over vibrational motion.

We define the *homogeneous extension* of a Hamiltonian by

$$\bar{H}(q, \alpha, p, \beta) \stackrel{\mathrm{def}}{=} \alpha^2 H(\alpha^{-1}q, \alpha^{-1}p).$$

Then $\bar{H}$ is homogeneous of order 2:

$$\bar{H}(\sigma y) = \sigma^2 \bar{H}(y), \tag{3.1}$$

where $y \stackrel{\mathrm{def}}{=} [q^{\mathrm{T}}, \alpha, p^{\mathrm{T}}, \beta]^{\mathrm{T}}$. The extended Hamiltonian yields the augmented system

$$\dot{q} = \alpha H_p(\alpha^{-1}q, \alpha^{-1}p), \qquad \dot{p} = -\alpha H_q(\alpha^{-1}q, \alpha^{-1}p),$$

$$\dot{\alpha} = 0, \qquad \dot{\beta} = q^{\mathrm{T}} H_q(\alpha^{-1}q, \alpha^{-1}p) + p^{\mathrm{T}} H_p(\alpha^{-1}q, \alpha^{-1}p) - 2\alpha H(\alpha^{-1}q, \alpha^{-1}p).$$

With initial condition $\alpha(0) = 1$, we have $\alpha \equiv 1$ and the system simplifies to

$$\dot{q} = H_p(q, p), \qquad \dot{p} = -H_q(q, p), \qquad \dot{\beta} = q^{\mathrm{T}} H_q(q, p) + p^{\mathrm{T}} H_p(q, p) - 2H(q, p).$$

For $H(q, p) = \frac{1}{2}p^{\mathrm{T}}M^{-1}p + U(q)$, the extended Hamiltonian is $\bar{H}(q, \alpha, p, \beta) = \frac{1}{2}p^{\mathrm{T}}M^{-1}p + \alpha^2 U(\alpha^{-1}q)$ and the simplified augmented system is

$$\dot{q} = M^{-1}p, \qquad \dot{p} = -U_q(q), \qquad \dot{\beta} = q^{\mathrm{T}} U_q(q) - 2U(q).$$

Remark. The association of α with q rather than with p is of practical importance in that it enables one to get values of $\dot\beta$ for free whenever $\dot p$ is calculated.

The following proposition shows that the value of the extended Hamiltonian can be calculated knowing only the solution.

PROPOSITION 1. *Let $\bar H(y)$ be the homogeneous extension of a given Hamiltonian $H(x)$, and let $y(t)$ be a solution of the extended Hamiltonian system with α initially* 1. *Then*

$$H(x(t)) \equiv \frac{1}{2}\dot y(t)^{\mathrm T}\bar J y(t),$$

where $\bar J$ is the matrix $\begin{bmatrix} 0 & I \\ -I & 0 \end{bmatrix}$ of augmented dimension.

Proof. Differentiating (3.1) with respect to σ gives $\bar H_y(y)^{\mathrm T} y = 2\bar H(y)$, so

$$\frac{1}{2}\dot y(t)^{\mathrm T}\bar J y(t) = \frac{1}{2}(\bar J \bar H_y(y(t)))^{\mathrm T}\bar J y(t) = \frac{1}{2}\bar H_y(y(t))^{\mathrm T} y(t) = \bar H(y(t)).$$

Because $\bar H$ is a homogeneous extension of H, the solution of $\bar H$ "includes" that of H and we have $\bar H(y(t)) \equiv H(x(t))$. □

Of course, the goal is not to calculate the original Hamiltonian for which we know a formula but not the solution; rather, it is to calculate a modified Hamiltonian for which we know the solution (at grid points) but not a formula. Therefore, we must augment the integrator so that its solution at grid points is that of the homogeneous extension of the modified Hamiltonian. For an integrator that is a composition of Hamiltonian flows, this is accomplished by using the homogeneous extension of each of the constituent Hamiltonians. More specifically, we define the augmented method $y^{n+1} = \Psi_h(y^n)$ for $\bar H$ to be the composition of exact flows for systems with Hamiltonians $\bar H_1$, $\bar H_2$, ..., $\bar H_L$, where

$$\bar H_l(q, \alpha, p, \beta) \overset{\mathrm{def}}{=} \alpha^2 H_l(\alpha^{-1}q, \alpha^{-1}p).$$

LEMMA 1 (commutativity). *The method Ψ_h defined above has a modified Hamiltonian*

$$\bar H_h(q, \alpha, p, \beta) = \alpha^2 H_h(\alpha^{-1}q, \alpha^{-1}p),$$

where $H_h(q,p)$ is the modified Hamiltonian of the original method Φ_h, i.e., the following diagram commutes:

$$\begin{tikzcd} H \arrow[r, "\textit{homogeneous extension}"] \arrow[d, "\textit{discretization}"'] & \bar H \arrow[d, "\textit{discretization}"] \\ H_h \arrow[r, "\textit{homogeneous extension}"'] & \bar H_h \end{tikzcd}$$

Proof. The modified Hamiltonian $\bar H_h$ for method Ψ_h can be expressed as an asymptotic expansion using the Baker–Campbell–Hausdorf formula [22]. This formula combines Hamiltonians using the operations of scalar multiplication, addition, and the Poisson bracket $\{H, N\} = H_x^{\mathrm T}\, J N_x$. It is thus sufficient to show that each of these three commute with the operation of forming the homogeneous extension. We show this only for the last of these. The homogeneous extension of the Poisson bracket is

$$\alpha^2 H_x(\alpha^{-1}x)^{\mathrm T} J N_x(\alpha^{-1}x).$$

This is exactly the same as the (extended) Poisson bracket of $\alpha^2 H(\alpha^{-1}x)$ and $\alpha^2 N(\alpha^{-1}x)$. □

Remark. The aim is to discretize the extended Hamiltonian so that this commutativity property holds. Extension of this technique to implicit Runge–Kutta methods would require an augmentation of the method so that commutativity holds.

The following proposition allows the value of the Hamiltonian to be approximated from known values of $y_h(t)$ at grid points.

PROPOSITION 2. *Let $x_h(t)$ and $y_h(t)$ be the solutions for modified Hamiltonians H_h and $\bar{H}_h$, respectively. Then*

$$H_h(x_h(t)) \equiv \frac{1}{2}\dot{y}_h(t)^{\mathrm{T}}\bar{J}y_h(t). \tag{3.2}$$

Proof. The proof is similar to that of Proposition 1. □

4. Using full step values. This section presents the construction of $H_{[2k]}$ for *even values of* k. The idea is to use (3.2) in a judiciously chosen weighted average.

Let $y_h(t)$ be the solution of the modified extended Hamiltonian system with initial condition $y(0) = y$. It has values $y_h(jh) = \Psi_h^j(y)$, $j = 0, \pm 1, \dots, \pm k/2$. Let $\pi_k(t)$ be the degree k polynomial interpolant of these values. (For large k it may be preferable, instead, to use trigonometric interpolation suitably modified [3].)

From Proposition 2, $\frac{1}{2}\dot{\pi}_k(t)J\pi_k(t) \approx \bar{H}_h(y)$. Let

$$H_{k,j} \overset{\text{def}}{=} \frac{1}{jh}\int_{-jh/2}^{jh/2}\frac{1}{2}\dot{\pi}_k(t)^{\mathrm{T}}J\pi_k(t)\mathrm{d}t, \qquad j = 2, 4, \dots, k.$$

The interpolant $\pi_k(t) = y_h(t) + e(t)$, where the error $e(t) = (t^2 - \frac{1}{4}k^2h^2)\cdots(t^2 - h^2)ty_h^{[k+1]}(t)$ and $y_h^{[k+1]}(t) \overset{\text{def}}{=} y_h[-\frac{1}{2}kh, \dots, \frac{1}{2}kh, t]$ with the brackets denoting a $(k+1)$th divided difference. Noting that $\dot{e}(t)^{\mathrm{T}}\bar{J}e(t) = O(h^{2k+2})$, we get

$$\begin{aligned}
H_{k,j} &= \bar{H}_h + \frac{1}{jh}\int_{-jh/2}^{jh/2}\frac{1}{2}\dot{y}_h(t)^{\mathrm{T}}\bar{J}e(t)\mathrm{d}t - \frac{1}{jh}\int_{-jh/2}^{jh/2}\frac{1}{2}y_h(t)^{\mathrm{T}}\bar{J}\dot{e}(t)\mathrm{d}t + O(h^{2k+2}) \\
&= \bar{H}_h + \frac{1}{jh/2}\int_{-jh/2}^{jh/2}\frac{1}{2}\dot{y}_h(t)^{\mathrm{T}}\bar{J}e(t)\mathrm{d}t + O(h^{2k+2}) \\
&= \bar{H}_h + \frac{1}{jh/2}\int_{-jh/2}^{jh/2}\left(t^2 - \frac{1}{4}k^2h^2\right)\cdots(t^2 - h^2)t\gamma(t)\mathrm{d}t + O(h^{2k+2}),
\end{aligned}$$

where the second equation is obtained by integration by parts and where $\gamma(t) \overset{\text{def}}{=} \frac{1}{2}\dot{y}_h(t)^{\mathrm{T}}\bar{J}y_h^{[k+1]}(t)$. This can be expressed as an expansion

$$H_{k,j} = \bar{H}_h + c_{j1}h^{k+2}\gamma'(0) + c_{j3}h^{k+4}\gamma'''(0) + \cdots + O(h^{2k+2}).$$

By forming a suitable linear combination of the values $H_{k,j}$, $j = 1, 2, \dots, k/2$, it is expected that one can get $\bar{H}_h$ with the first $k/2 - 1$ leading error terms eliminated:

$$\text{linear combination of the } H_{k,j} = \bar{H}_h + O(h^{2k}).$$

Note. The value $\dot{\pi}_k(0)^{\mathrm{T}}\bar{J}\pi_k(0)$ contains a leading term that is only $O(h^k)$, so it is not useful for eliminating error terms.

The case $k = 2$ is the fourth order accurate formula

$$H_{[4]}(x) \overset{\text{def}}{=} H_{2,2} = \bar{H}_h(y) + O(h^4).$$

For the case $k = 4$, we have

$$\begin{aligned} H_{4,2} &= \bar{H}_h + \frac{1}{h}\int_{-h}^{h}(t^2 - 4h^2)(t^2 - h^2)t\gamma(t)\mathrm{d}t + O(h^{10}) \\ &= \bar{H}_h + \frac{20}{21}h^6\gamma'(0) + O(h^8) \end{aligned}$$

and

$$\begin{aligned} H_{4,4} &= \bar{H}_h + \frac{1}{2h}\int_{-2h}^{2h}(t^2 - 4h^2)(t^2 - h^2)t\gamma(t)\mathrm{d}t + O(h^{10}) \\ &= \bar{H}_h - \frac{64}{21}h^6\gamma'(0) + O(h^8), \end{aligned}$$

and hence

$$H_{[8]}(x) \overset{\text{def}}{=} \frac{16}{21}H_{4,2} + \frac{5}{21}H_{4,4} = \bar{H}_h(y) + O(h^8).$$

Below are given formulas for $H_{[8]}$ and for $H_{[4]}$ in terms of values of $y_h(t)$ at grid points. Let a_j be the jth centered difference of $y_h(t)$ at $t = 0$:

$$a_j = \begin{cases} \delta^j y_h(0), & j = 0, 2, 4, \ldots, \\ \mu\delta^j y_h(0), & j = 1, 3, \ldots, \end{cases}$$

where the centered difference operator is defined by $\delta w(t) = w(t + h/2) - w(t - h/2)$ and the averaging operator is defined by $\mu w(t) = \frac{1}{2}w(t + h/2) + \frac{1}{2}w(t - h/2)$.

The fourth degree interpolant in divided difference form is

$$\pi_4(t) = y_h(0) + t\frac{\mu\delta y_h(0)}{h} + t^2\frac{\delta^2 y_h(0)}{2h^2} + t(t^2 - h^2)\frac{\mu\delta^3 y_h(0)}{6h^3} + t^2(t^2 - h^2)\frac{\delta^4 y_h(0)}{24h^4}.$$

Hence,

$$\pi_4(sh) = a_0 + a_1 s + \frac{1}{2}a_2 s^2 + \frac{1}{6}a_3 s(s^2 - 1) + \frac{1}{24}a_4 s^2(s^2 - 1)$$

and

$$h\dot{\pi}_4(sh) = a_1 + a_2 s + \frac{1}{2}a_3\left(s^2 - \frac{1}{3}\right) + \frac{1}{6}a_4 s\left(s^2 - \frac{1}{2}\right).$$

Define

$$A_{ij} = a_i^{\mathrm{T}}\bar{J}a_j/(2h),$$

and we have

$$\begin{aligned} \frac{1}{2}\dot{\pi}_4(sh)^{\mathrm{T}}\bar{J}\pi_4(sh) = A_{10} &- \frac{1}{2}A_{12}s^2 + \frac{1}{2}A_{30}\left(s^2 - \frac{1}{3}\right) + \frac{1}{12}A_{32}s^2(s^2 + 1) \\ &- \frac{1}{8}A_{14}s^2\left(s^2 - \frac{1}{3}\right) \\ &- \frac{1}{144}A_{34}s^2(s^2 - 1)^2 + \text{ odd powers of } s. \end{aligned}$$

Averaging over $-1 \le s \le 1$ yields

$$H_{4,2} = A_{10} - \frac{1}{6}A_{12} + \frac{2}{45}A_{32} - \frac{1}{90}A_{14} - \frac{1}{1890}A_{34}, \tag{4.1}$$

and averaging over $-2 \le s \le 2$ yields

$$H_{4,4} = A_{10} - \frac{2}{3}A_{12} + \frac{1}{2}A_{30} + \frac{17}{45}A_{32} - \frac{31}{90}A_{14} - \frac{107}{3780}A_{34}.$$

Therefore,

$$H_{[8]}(x) = \frac{16}{21}H_{4,2} + \frac{5}{21}H_{4,4} \tag{4.2}$$

$$= A_{10} - \frac{2}{7}A_{12} + \frac{5}{42}A_{30} + \frac{13}{105}A_{32} - \frac{19}{210}A_{14} - \frac{1}{140}A_{34}. \tag{4.3}$$

For a second degree interpolant, it follows from (4.1) that

$$H_{[4]}(x) = H_{2,2} = A_{10} - \frac{1}{6}A_{12}. \tag{4.4}$$

An implementation of these formulas might calculate $H_{[2k]}(x^n)$ for consecutive values of n in terms of quantities $A^n_{ij} = (a^n_i)^{\mathrm{T}}\bar{J}a^n_j/(2h)$ defined in terms of centered differences of y^n which can be obtained from the x^n. (Only first and higher differences of β^n are needed.)

Example 1. To make this concrete, we calculate $H_{[4]}(x)$ for the leapfrog method, as given by (1.2). The leapfrog method advances one step by

$$\begin{aligned}
p^{n+1/2} &= p^n + \frac{h}{2}F^n,\\
\beta^{n+1/2} &= \beta^n + \frac{h}{2}(-(q^n)^{\mathrm{T}}F^n - 2U^n),\\
q^{n+1} &= q^n + hM^{-1}p^{n+1/2},\\
p^{n+1} &= p^{n+1/2} + \frac{h}{2}F^{n+1},\\
\beta^{n+1} &= \beta^{n+1/2} + \frac{h}{2}(-(q^{n+1})^{\mathrm{T}}F^{n+1} - 2U^{n+1}).
\end{aligned}$$

We have

$$H_{[4]}(q^n, p^n) = \frac{1}{2h}(\mu\delta y^n)^{\mathrm{T}}\bar{J}\left(y^n - \frac{1}{6}\delta^2 y^n\right).$$

Suppressing the n in the superscript,

$$y^{\pm 1} = \begin{bmatrix} q \pm hM^{-1}p + \frac{1}{2}h^2M^{-1}F \\ 1 \\ p \pm \frac{1}{2}hF \pm \frac{1}{2}hF^{\pm 1} \\ \beta \pm \frac{1}{2}h(-2U - q^{\mathrm{T}}F) \pm \frac{1}{2}h(-2U^{\pm 1} - (q^{\pm 1})^{\mathrm{T}}F^{\pm 1}) \end{bmatrix},$$

whence

$$\mu\delta y = \begin{bmatrix} hM^{-1}p \\ 0 \\ hF + \frac{1}{4}h\delta^2 F \\ -2hU - hq^{\mathrm{T}}F - \frac{1}{2}h\delta^2 U - \frac{1}{4}h\delta^2(q^{\mathrm{T}}F) \end{bmatrix}, \quad \delta^2 y = \begin{bmatrix} h^2M^{-1}F \\ 0 \\ h\mu\delta F \\ \text{not needed} \end{bmatrix}. \tag{4.5}$$

From $\delta(ab) = \delta a \cdot \mu b + \mu a \cdot \delta b$ and $\mu^2 = 1 + \frac{1}{4}\delta^2$, we get

$$\delta^2(ab) = \delta^2 a \cdot b + 2\mu\delta a \cdot \mu\delta b + a \cdot \delta^2 b + \frac{1}{2}\delta^2 a \cdot \delta^2 b,$$

so

$$\delta^2(q^{\mathrm{T}}F) = h^2 F^{\mathrm{T}} M^{-1} F + 2hp^{\mathrm{T}} M^{-1}\mu\delta F + q^{\mathrm{T}}\delta^2 F + \frac{1}{2}h^2 F^{\mathrm{T}} M^{-1}\delta^2 F. \tag{4.6}$$

Therefore,

$$\begin{aligned} H_{[4]}(q^n, p^n) = H(q^n, p^n) &+ \frac{1}{4}\delta^2 U^n + \frac{1}{6}h(p^n)^{\mathrm{T}} M^{-1}\mu\delta F^n + \frac{5}{24}h^2(F^n)^{\mathrm{T}} M^{-1} F^n \\ &+ \frac{1}{12}h^2(F^n)^{\mathrm{T}} M^{-1}\delta^2 F^n. \end{aligned}$$

4.1. The case of a quadratic Hamiltonian. The following result implies that, in the case where $H(x)$ is quadratic, the numerical solution exactly conserves an approximate modified Hamiltonian which is a linear functional of $\frac{1}{2}\dot\pi(t)^{\mathrm{T}}\bar J\pi(t)$, where $\pi(t)$ is a linear combination of numerical solution values.

Proposition 3. *Assume that Φ_h is the composition of flows for systems with quadratic Hamiltonians and that Ψ_h is constructed as in Lemma* 1. *Then the quantity*

$$\bar H_*(y) \overset{\text{def}}{=} \sum_{i,j} a_{i,j}\Psi_h^i(y)^{\mathrm{T}}\bar J\Psi_h^j(y),$$

where the sum is taken over a finite set of pairs of integers, is exactly conserved by method Ψ_h.

Proof. The mapping $\Psi_h(y) = Sy$ for some symplectic matrix S because Ψ_h is the composition of flows for systems with homogeneous quadratic Hamiltonians. Then

$$\begin{aligned} \bar H_*(\Psi_h(y)) &= \sum_{i,j} a_{i,j}(S^i S y)^{\mathrm{T}}\bar J(S^j S y) \\ &= \sum_{i,j} a_{i,j}(S^i y)^{\mathrm{T}} S^{\mathrm{T}}\bar J S(S^j y) \\ &= \sum_{i,j} a_{i,j}\Psi_h^i(y)^{\mathrm{T}}\bar J\Psi_h^j(y). \qquad \square \end{aligned}$$

5. Using intermediate values. This section presents the construction of $H_{[2k]}$ for *odd values of k.*

For most numerical integrators one can define "sensible" midstep values

$$y^{n\pm1/2}, y^{n\pm3/2}, \ldots, y^{n\pm k/2},$$

and these can be used instead of full step values to get an estimate accurate up to $O(h^{2k})$. We assume that $\Psi_h = \hat\Psi_{-h/2}^{-1} \circ \hat\Psi_{h/2}$, where $\hat\Psi_{h/2}$ is a composition of exact flows of homogeneously extended Hamiltonians.

Remark. It is not necessary that the midstep values be approximations to $y(t)$ at midpoints nor that Ψ_h be time symmetric ($\Psi_h^{-1} = \Psi_{-h}$). All we need is that $\Psi_h = \Psi_{2,h} \circ \Psi_{1,h}$ where each of $\Psi_{1,h}$, $\Psi_{2,h}$ is a composition of exact flows of homogeneously extended Hamiltonians.

For example, the leapfrog method separates into half steps $\hat{\Psi}_{-h/2}^{-1} \circ \hat{\Psi}_{h/2}$ with $\hat{\Psi}_{h/2}$ as follows:

$$\begin{aligned} p^{n+1/2} &= p^n + \frac{h}{2}F^n, \\ \beta^{n+1/2} &= \beta^n + \frac{h}{2}(-(q^n)^{\mathrm{T}}F^n - 2U^n), \\ q^{n+1/2} &= q^n + \frac{h}{2}M^{-1}p^{n+1/2}. \end{aligned}$$

Remark. For the leapfrog method, the estimate over an interval from $(n - \frac{1}{2}k)h$ to $(n + \frac{1}{2}k)h$, where k is odd, actually uses only values of energy and forces from the shorter interval from $(n - \frac{1}{2}k + \frac{1}{2})h$ to $(n + \frac{1}{2}k - \frac{1}{2})h$.

The midstep values are values at midpoints of some function $z_h(t)$ which can be used to construct the Hamiltonian.

PROPOSITION 4. *Let $z_h(t) = \Phi_{h/2}(y_h(t - h/2))$. Then*

$$H_h(x_h(t)) \equiv \frac{1}{2}\dot{z}_h(t)^{\mathrm{T}}\bar{J}z_h(t).$$

Proof. For any real s, we define $\Psi_h^s = sh$-flow for $\dot{y} = \bar{J}\bar{H}_{h,y}(y)$. Then $z_h(t) = \chi_h(y_h(t))$, where $\chi_h = \Phi_{h/2} \circ \Psi_h^{-1/2}$. Because χ_h is symplectic, $z_h(t) = \chi_h(y_h(t))$ satisfies a Hamiltonian system with Hamiltonian $\bar{H}_h \circ \chi_h^{-1}$. Also, $\chi_h(\sigma z) = \sigma\chi_h(z)$ because χ_h is the composition of flows of Hamiltonians that are second order homogeneous, and hence $\bar{H}_h \circ \chi_h^{-1}$ is homogeneous of order 2. Therefore,

$$\frac{1}{2}\dot{z}_h(t)^{\mathrm{T}}\bar{J}z_h(t) \equiv \bar{H}_h \circ \chi_h^{-1}(z_h(t)) = \bar{H}_h(y_h(t)). \qquad \square$$

Let $\pi_k(t)$ be the degree k polynomial interpolant of the values $z_h(jh) = \hat{\Psi}_{h/2} \circ \Psi_h^{j-1/2}(y)$, $j = \pm\frac{1}{2}, \pm\frac{3}{2}, \dots, \pm k/2$.

As before, let

$$H_{k,j} = \frac{1}{jh}\int_{-jh/2}^{jh/2} \frac{1}{2}\dot{\pi}_k(t)^{\mathrm{T}}\bar{J}\pi_k(t)\mathrm{d}t, \qquad j = 1, 3, \dots, k.$$

The interpolant $\pi_k(t) = z_h(t) + e(t)$, where the error $e(t) = (t^2 - \frac{1}{4}k^2h^2)\cdots(t^2 - \frac{1}{4}h^2)z_h^{[k+1]}(t)$ and $z_h^{[k+1]}(t) \stackrel{\text{def}}{=} z_h[-\frac{1}{2}kh, \dots, \frac{1}{2}kh, t]$. Similar to before, we get

$$H_{k,j} = \bar{H}_h + \frac{1}{jh/2}\int_{-jh/2}^{jh/2}\left(t^2 - \frac{1}{4}k^2h^2\right)\cdots\left(t^2 - \frac{1}{4}h^2\right)\gamma(t)\mathrm{d}t + O(h^{2k+2}),$$

where $\gamma(t) \stackrel{\text{def}}{=} \frac{1}{2}\dot{z}_h(t)^{\mathrm{T}}\bar{J}z_h^{[k+1]}(t)$. This can be expressed as an expansion

$$H_{k,j} = \bar{H}_h + c_{j0}h^{k+1}\gamma(0) + c_{j2}h^{k+3}\gamma''(0) + \cdots + O(h^{2k+2}).$$

Again, it is expected that a suitable linear combination of the $(k+1)/2$ different values of $H_{k,j}$ yields $\bar{H}_h$ with the first $(k+1)/2 - 1$ leading error terms eliminated:

$$\text{linear combination of the } H_{k,j} = \bar{H}_h + O(h^{2k}).$$

Note. It does not seem possible to combine values obtained from full steps with those from half steps to further increase the order of accuracy because the error expansions for the two kinds of averages do not have terms in common that can cancel.

For $k = 1$, we have the second order formula

$$H_{[2]}(x) \overset{\text{def}}{=} H_{1,1} = \bar{H}_h(y) + O(h^2).$$

For $k = 3$, we have

$$\begin{aligned} H_{3,1} &= \bar{H}_h + \frac{1}{h/2}\int_{-h/2}^{h/2}\left(t^2 - \frac{9}{4}h^2\right)\left(t^2 - \frac{1}{4}h^2\right)\gamma(t)\mathrm{d}t + O(h^8) \\ &= \bar{H}_h + \frac{11}{15}h^4\gamma(0) + O(h^6) \end{aligned}$$

and

$$\begin{aligned} H_{3,3} &= \bar{H}_h + \frac{1}{3h/2}\int_{-3h/2}^{3h/2}\left(t^2 - \frac{9}{4}h^2\right)\left(t^2 - \frac{1}{4}h^2\right)\gamma(t)\mathrm{d}t + O(h^8) \\ &= \bar{H}_h - \frac{3}{5}h^4\gamma(0) + O(h^6), \end{aligned}$$

whence

$$H_{[6]}(x) \overset{\text{def}}{=} \frac{9}{20}H_{3,1} + \frac{11}{20}H_{3,3} = \bar{H}_h(y) + O(h^6).$$

Let b_j be the jth centered difference of $z_h(t)$ at $t = 0$ using midstep values

$$b_j = \begin{cases} \mu\delta^j z_h(0), & j = 0, 2, 4, \ldots, \\ \delta^j z_h(0), & j = 1, 3, \ldots. \end{cases}$$

The third degree interpolant is

$$\pi_3(t) = \mu z_h(0) + t\frac{\delta z_h(0)}{h} + \left(t^2 - \frac{1}{4}h^2\right)\frac{\mu\delta^2 z_h(0)}{2h^2} + \left(t^2 - \frac{1}{4}h^2\right)t\frac{\delta^3 z_h(0)}{6h^3}.$$

Hence,

$$\pi_3(sh) = b_0 + b_1 s + \frac{1}{2}b_2\left(s^2 - \frac{1}{4}\right) + \frac{1}{6}b_3 s\left(s^2 - \frac{1}{4}\right)$$

and

$$h\dot{\pi}_3(sh) = b_1 + b_2 s + \frac{1}{2}b_3\left(s^2 - \frac{1}{12}\right).$$

Define

$$B_{ij} = b_i^{\mathrm{T}}\bar{J}b_j/(2h),$$

and we have

$$\begin{aligned} \frac{1}{2}\dot{\pi}_3(sh)^{\mathrm{T}}\bar{J}\pi_3(sh) = B_{10} - \frac{1}{2}B_{12}\left(s^2 + \frac{1}{4}\right) + \frac{1}{2}B_{30}\left(s^2 - \frac{1}{12}\right) \\ + \frac{1}{12}B_{32}\left(s^2 - \frac{1}{4}\right)^2 + \text{ odd powers of } s. \end{aligned}$$

Averaging over $-\frac{1}{2} \le s \le \frac{1}{2}$ yields

$$H_{3,1} = B_{10} - \frac{1}{6}B_{12} + \frac{1}{360}B_{32}, \tag{5.1}$$

and averaging over $-\frac{3}{2} \le s \le \frac{3}{2}$ yields

$$H_{3,3} = B_{10} - \frac{1}{2}B_{12} + \frac{1}{3}B_{30} + \frac{7}{120}B_{32}.$$

Therefore,

$$H_{[6]}(x) = B_{10} - \frac{7}{20}B_{12} + \frac{11}{60}B_{30} + \frac{1}{30}B_{32}. \tag{5.2}$$

For a first degree interpolant, it follows from (5.1) that

$$H_{[2]}(x) = B_{10}. \tag{5.3}$$

Example 2. We calculate $H_{[2]}(x)$ for the leapfrog method, as given by (1.1). We have

$$H_{[2]}(q^n, p^n) = \frac{1}{2h}(\delta y^n)^{\mathrm{T}} \bar{J}(\mu y^n).$$

Suppressing the n in the superscript,

$$y^{\pm 1/2} = \begin{bmatrix} q \pm \frac{1}{2}hM^{-1}p + \frac{1}{4}h^2M^{-1}F \\ 1 \\ p \pm \frac{1}{2}hF \\ \pm\frac{1}{2}h(-2U - q^{\mathrm{T}}F) \end{bmatrix},$$

whence

$$\delta y = \begin{bmatrix} hM^{-1}p \\ 0 \\ hF \\ -2hU - hq^{\mathrm{T}}F \end{bmatrix}, \qquad \mu y = \begin{bmatrix} q + \frac{1}{4}h^2M^{-1}F \\ 1 \\ p \\ 0 \end{bmatrix}. \tag{5.4}$$

Therefore,

$$H_{[2]}(q^n, p^n) = H(q^n, p^n) - \frac{1}{8}h^2(F^n)^{\mathrm{T}}M^{-1}F^n.$$

Example 3. We calculate $H_{[6]}(x)$ for the leapfrog method, as given by (1.3). From (5.4) we have

$$\delta^3 y = \begin{bmatrix} hM^{-1}\delta^2 p \\ 0 \\ h\delta^2 F \\ -2h\delta^2 U - h\delta^2(q^{\mathrm{T}}F) \end{bmatrix}, \qquad \mu\delta^2 y = \begin{bmatrix} \delta^2 q + \frac{1}{4}h^2M^{-1}\delta^2 F \\ 0 \\ \delta^2 p \\ 0 \end{bmatrix}.$$

We have $\delta^2 q = h^2M^{-1}F$ and $\delta^2 p = h\mu\delta F$ from the second part of (4.5), and $\delta^2(q^{\mathrm{T}}F)$ is given by (4.6). Then

$$
\begin{aligned}
b_1^{\mathrm{T}}\bar{J}b_0 &= 2hU + hp^{\mathrm{T}}M^{-1}p - \frac{1}{4}h^3F^{\mathrm{T}}M^{-1}F,\\
b_1^{\mathrm{T}}\bar{J}b_2 &= h^2p^{\mathrm{T}}M^{-1}\mu\delta F - h^3F^{\mathrm{T}}M^{-1}F - \frac{1}{4}h^3F^{\mathrm{T}}M^{-1}\delta^2F,\\
b_3^{\mathrm{T}}\bar{J}b_0 &= 2h\delta^2U + 3h^2p^{\mathrm{T}}M^{-1}\mu\delta F + h^3F^{\mathrm{T}}M^{-1}F + \frac{1}{4}h^3F^{\mathrm{T}}M^{-1}\delta^2F,\\
b_3^{\mathrm{T}}\bar{J}b_2 &= -h^3F^{\mathrm{T}}M^{-1}\delta^2F + h^3(\mu\delta F)^{\mathrm{T}}M^{-1}\mu\delta F - \frac{1}{4}h^3(\delta^2F)^{\mathrm{T}}M^{-1}\delta^2F,
\end{aligned}
$$

and, therefore,

$$
\begin{aligned}
H_{[6]}(q^n,p^n) = H(q^n,p^n) &+ \frac{11}{60}\delta^2U^n + \frac{1}{10}h(p^n)^{\mathrm{T}}M^{-1}\mu\delta F^n + \frac{17}{120}h^2(F^n)^{\mathrm{T}}M^{-1}F^n\\
&+ \frac{1}{10}h^2(F^n)^{\mathrm{T}}M^{-1}\delta^2F^n + \frac{1}{60}h^2(\mu\delta F^n)^{\mathrm{T}}M^{-1}\mu\delta F^n\\
&- \frac{1}{240}h^2(\delta^2F^n)^{\mathrm{T}}M^{-1}\delta^2F^n.
\end{aligned}
$$

5.1. The case of a quadratic Hamiltonian.

PROPOSITION 5. *Assume that Φ_h is the composition of flows for systems with quadratic Hamiltonians, that Ψ_h is constructed as in Lemma* 1, *and that $\hat{\Psi}_{h/2}$ is as assumed at the beginning of this section. Then the quantity*

$$
\bar{H}_*(y) \overset{\mathrm{def}}{=} \sum_{i,j} a_{i,j}\hat{\Psi}_{h/2}\circ\Psi_h^i(y)^{\mathrm{T}}\bar{J}\hat{\Psi}_{h/2}\circ\Psi_h^j(y),
$$

where the sum is taken over a finite set of pairs of integers, is exactly conserved by method Ψ_h.

Proof. Because $\hat{\Psi}_{h/2}$, Ψ_h are the compositions of flows for systems with homogeneous quadratic Hamiltonians, the mappings $\hat{\Psi}_{h/2}(y) = S_1y$ and $\Psi_h(y) = S_2S_1y$ for some symplectic matrices S_1, S_2. Then

$$
\begin{aligned}
\bar{H}_*(\Psi_h(y)) &= \sum_{i,j} a_{i,j}(S_1S^iSy)^{\mathrm{T}}\bar{J}(S_1S^jSy)\\
&= \sum_{i,j} a_{i,j}(S_1S^iy)^{\mathrm{T}}(S_1S_2)^{\mathrm{T}}\bar{J}S_1S_2(S_1S^jy)\\
&= \sum_{i,j} a_{i,j}\hat{\Psi}_{h/2}\circ\Psi_h^i(y)^{\mathrm{T}}\bar{J}\hat{\Psi}_{h/2}\circ\Psi_h^j(y). \qquad \square
\end{aligned}
$$

Acknowledgment. The authors are grateful for the assistance of Justin Wozniak, who did preliminary tests of the second order truncation for the Hénon–Heiles Hamiltonian and for decalanine.

REFERENCES

[1] http://www.ks.uiuc.edu/Research/namd/utilities.

[2] G. BENETTIN AND A. GIORGILLI, *On the Hamiltonian interpolation of near to the identity symplectic mappings with application to symplectic integration algorithms*, J. Statist. Phys., 74 (1994), pp. 1117–1143.

[3] J. GANS, J. I. CHAN, AND D. SHALLOWAY, *Residual Acceleration as a Measure of the Accuracy of Molecular Dynamics Simulations*, manuscript.

[4] J. GANS AND D. SHALLOWAY, *Shadow mass and the relationship between velocity and momentum in symplectic numerical integration*, Phys. Rev. E (3), 61 (2000), pp. 4587–4592.

[5] B. García-Archilla, J. M. Sanz-Serna, and R. D. Skeel, *Long-time-step methods for oscillatory differential equations*, SIAM J. Sci. Comput., 20 (1998), pp. 930–963.

[6] D. F. Griffiths and J. M. Sanz-Serna, *On the scope of the method of modified equations*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 994–1008.

[7] H. Grubmüller, H. Heller, A. Windemuth, and K. Schulten, *Generalized Verlet algorithm for efficient molecular dynamics simulations with long-range interactions*, Molecular Simulation, 6 (1991), pp. 121–142.

[8] E. Hairer and C. Lubich, *The life-span of backward error analysis for numerical integrators*, Numer. Math., 76 (1997), pp. 441–462.

[9] E. Hairer and C. Lubich, *Asymptotic expansions and backward analysis for numerical integrators*, in Dynamics of Algorithms, R. de la Llave, L. Petzold, and J. Lorenz, eds., IMA Vol. Math. Appl. 118, Springer-Verlag, New York, 2000, pp. 91–106.

[10] J. Izaguirre, S. Reich, and R. D. Skeel, *Longer time steps for molecular dynamics*, J. Chem. Phys., 110 (1999), pp. 9853–9864.

[11] W. L. Jorgensen, J. Chandrasekhar, J. D. Madura, R. W. Impey, and M. L. Klein, *Comparison of simple potential functions for simulating liquid water*, J. Chem. Phys., 79 (1983), pp. 926–935.

[12] L. Kalé, R. Skeel, R. Brunner, M. Bhandarkar, A. Gursoy, N. Krawetz, J. Phillips, A. Shinozaki, K. Varadarajan, and K. Schulten, *NAMD*2*: Greater scalability for parallel molecular dynamics*, J. Comput. Phys, 151 (1999), pp. 283–312.

[13] A. R. Leach, *Molecular Modelling: Principles and Applications*, Addison-Wesley, Reading, MA, 1996.

[14] A. D. MacKerell Jr., D. Bashford, M. Bellott, R. L. Dunbrack Jr., J. D. Evanseck, M. J. Field, S. Fischer, J. Gao, H. Guo, S. Ha, D. Joseph-McCarthy, L. Kuchnir, K. Kuczera, F. T. K. Lau, C. Mattos, S. Michnick, T. Ngo, D. T. Nguyen, B. Prodhom, W. E. Reiher, III, B. Roux, M. Schlenkrich, J. C. Smith, R. Stote, J. Straub, M. Watanabe, J. Wiorkiewicz-Kuczera, D. Yin, and M. Karplus, *All-atom empirical potential for molecular modeling and dynamics studies of proteins*, J. Phys. Chem. B, 102 (1998), pp. 3586–3616.

[15] A. D. MacKerell Jr., D. Bashford, M. Bellott, R. L. Dunbrack Jr., M. J. Field, S. Fischer, J. Gao, H. Guo, S. Ha, D. Joseph, L. Kuchnir, K. Kuczera, F. T. K. Lau, C. Mattos, S. Michnick, T. Ngo, D. T. Nguyen, B. Prodhom, B. Roux, M. Schlenkrich, J. C. Smith, R. Stote, J. Straub, J. Wiorkiewicz-Kuczera, and M. Karplus, *Self-consistent parameterization of biomolecules for molecular modeling and condensed phase simulations*, FASEB J., 6 (1992), p. A143.

[16] A. K. Mazur, *Common molecular dynamics algorithms revisited: Accuracy and optimal time steps of Störmer–leapfrog*, J. Comput. Phys., 136 (1997), pp. 354–365.

[17] J. K. Moser, *Lectures in Hamiltonian systems*, Mem. Amer. Math. Soc., 81 (1968), pp. 1–60.

[18] A. I. Neishtadt, *The separation of motions in systems with rapidly rotating phase*, J. Appl. Math. Mech., 48 (1984), pp. 133–139.

[19] R. W. Pastor, B. R. Brooks, and A. Szabo, *An analysis of the accuracy of Langevin and molecular dynamics algorithm*, Molecular Phys., 65 (1988), pp. 1409–1419.

[20] S. Reich, *Backward error analysis for numerical integrators*, SIAM J. Numer. Anal., 36 (1999), pp. 1549–1570.

[21] G. Rowlands, *A numerical algorithm for Hamiltonian systems*, J. Comput. Phys., 97 (1991), pp. 235–239.

[22] J. Sanz-Serna and M. Calvo, *Numerical Hamiltonian Problems*, Chapman and Hall, London, 1994.

[23] T. Schlick, M. Mandziuk, R. D. Skeel, and K. Srinivas, *Nonlinear resonance artifacts in molecular dynamics simulations*, J. Comput. Phys., 139 (1998), pp. 1–29.

[24] R. D. Skeel, *Integration schemes for molecular dynamics and related applications*, in The Graduate Student's Guide to Numerical Analysis, M. Ainsworth, J. Levesley, and M. Marletta, eds., Springer Ser. Comput. Math. 26, Springer-Verlag, Berlin, 1999, pp. 119–176.

[25] D. M. Stoffer, *Some Geometric and Numerical Methods for Perturbed Integrable Systems*, Ph.D. thesis, Swiss Federal Institute of Technology, Zürich, Switzerland, 1988.

[26] M. Tuckerman, B. J. Berne, and G. J. Martyna, *Reversible multiple time scale molecular dynamics*, J. Chem. Phys., 97 (1992), pp. 1990–2001.

[27] R. F. Warming and B. J. Hyett, *The modified equation approach to the stability and accuracy analysis of finite difference methods*, J. Comput. Phys., 14 (1974), pp. 159–179.

SIAM J. Sci. Comput.
Vol. 23, No. 4, pp. 1189–1203

A CHANGING-CHART SYMPLECTIC ALGORITHM FOR RIGID BODIES AND OTHER HAMILTONIAN SYSTEMS ON MANIFOLDS*

GIANCARLO BENETTIN[†], ANNA MARIA CHERUBINI[‡], AND FRANCESCO FASSÒ[†]

Abstract. We revive the elementary idea of constructing symplectic integrators for Hamiltonian flows on manifolds by covering the manifold with the charts of an atlas, implementing the algorithm in each chart (thus using coordinates) and switching among the charts whenever a coordinate singularity is approached. We show that this program can be implemented successfully by using a splitting algorithm if the Hamiltonian is the sum $H_1 + H_2$ of two (or more) integrable Hamiltonians. Profiting from integrability, we compute exactly the flows of H_1 and H_2 in each chart and thus compute the splitting algorithm on the manifold by means of its representative in any chart. This produces a symplectic algorithm on the manifold which possesses an interpolating Hamiltonian, and hence it has excellent properties of conservation of energy. We exemplify the method for a point constrained to the sphere and for a symmetric rigid body under the influence of positional potential forces.

Key words. symplectic integrators, constrained Hamiltonian systems, rigid body

AMS subject classifications. 65P10, 70H99, 70E17

PII. S1064827500381720

1. Introduction and description of the method.

Symplectic integrators are currently very popular for Hamiltonian systems, particularly in view of their stability and excellent energy conservation. Since the phase space of a number of Hamiltonian systems of relevant interest (e.g., rigid bodies) is not the Euclidean space $\mathbb{R}^{2n}$, but a symplectic manifold M, symplectic integrators on manifolds are of special importance.

If a manifold does not possess a global system of coordinates, integrations performed using coordinates must necessarily deal with the presence of coordinate singularities. Probably the simplest idea for overcoming this situation is to consider a number of different systems of coordinates which have the singularities in different positions, and hence they form an atlas for M. One then implements an algorithm within each coordinate system and switches from one to the other during the integration whenever a coordinate singularity is approached. A changing-chart method of this kind has indeed been used in early molecular dynamics integrations of diatomic molecules [1] (in which case the manifold M is the cotangent bundle of the two-dimensional sphere), but the adopted algorithm was not symplectic. In fact, as noticed, e.g., in [13], even using a symplectic algorithm in each chart (e.g., a leapfrog) does not necessarily produce an algorithm on the manifold which has the stability and energy conservation of symplectic algorithms.[1]

To our knowledge, no symplectic changing-chart method has ever been developed.

*Received by the editors November 29, 2000; accepted for publication (in revised form) January 30, 2001; published electronically November 7, 2001.
http://www.siam.org/journals/sisc/23-4/38172.html

[†]Università di Padova, Dipartimento di Matematica Pura e Applicata, GNFM, and INFM, Via G. Belzoni 7, 35131 Padova, Italy (benettin@math.unipd.it, fasso@math.unipd.it). The work of the third author was partially supported by EU contract HPRN-CT-2000-0113 for the project *MASIE—Mechanics and Symmetry in Europe.*

[‡]Università di Lecce, Dipartimento di Matematica, and GNFM, Via per Arnesano, 73100 Lecce, Italy (anna.cherubini@unile.it).

[1] More precisely, of symplectic integrators which possess a global interpolating Hamiltonian; see section 2.

From the (vast) literature on the subject, one receives the impression that the presence of singularities has generally been felt as a serious difficulty for any approach based on the use of local coordinates (e.g., Euler angles in the rigid body case). A variety of alternative "geometric" approaches have thus been developed. A number of them share the common idea of embedding the manifold M in a higher-dimensional space $\mathbb{R}^{2N}$ for some $N > n$, where $2n$ is the dimension of M (see, e.g., [6], [7], [10], [13], [14], [15], [16], [17]); this achieves the goal of obtaining a global parametrization but at the price of introducing more variables. In addition, the confinement to M has to be enforced in some way: for instance, the so called RATTLE algorithm, successfully applied, in particular, to the rigid body, resorts to the use of $2(N-n)$ Lagrange multipliers, so the number of variables to be managed further increases.[2] Moreover, these geometric algorithms are often implicit; i.e., numeric inversions are necessary at each time-step. Nevertheless, in the literature these geometric schemes are generally considered superior to changing-charts methods.

In this article we propose a different point of view; namely, we suggest the following.

(i) It is possible to implement changing-chart symplectic algorithms simply by *choosing the algorithms in the individual charts in such a way that they are the local representatives of a symplectic map on the manifold.* This is indeed enough to ensure, as is crucial, that the trajectories computed in the local charts are the local representatives of trajectories of a symplectic map on the manifold (the "geometric integrator"); all properties of the map are then automatically preserved, up to the round-off errors, by the local algorithms.

(ii) A natural way of implementing this program is by means of the so-called "splitting methods"; moreover, such an implementation ensures the existence on the manifold of the so-called "interpolating Hamiltonian"—a rather crucial fact for symplectic integrators.

(iii) In various situations, this leads to computationally efficient explicit algorithms, which are competitive with other geometric algorithms.

We will discuss the method theoretically in section 2. In addition, in section 3 we will illustrate the method by means of two examples, which are of interest, for instance, in molecular dynamics and in celestial mechanics. The first example is the point constrained to a sphere and subject to conservative forces, for which $M = T^*S^2$. The second is the symmetric rigid body in a positional conservative force field; assuming for the purpose of the discussion that the body has a fixed point, then $M = T^*\mathrm{SO}(3)$. We will use an atlas for the sphere made of two systems of spherical coordinates and an atlas for SO(3) made of two systems of Euler angles.

We now shortly describe the method. We use the following notations: if (M, σ) is a symplectic manifold and $F : M \to \mathbb{R}$ is a smooth function, then we denote by Φ^F_ϵ the map at time ϵ of the flow of the Hamiltonian vector field X_F of F; as is well known, for each ϵ, Φ^F_ϵ is a symplectic diffeomorphism from M to itself. (The vector field X_F is assumed to be complete.)

The splitting methods (see, e.g., [18], [19], [17]) are based on the idea of decomposing the Hamiltonian as the sum of two or more terms, the flows of which are easily computable. If

$$H = H_1 + H_2,$$

[2] For the rigid body, schemes using quaternions [6], [7] require only a normalization of the quaternion, but this seems to destroy symplecticity.

then the map

$$\Psi_\epsilon = \Phi^{H_2}_{\epsilon/2} \circ \Phi^{H_1}_{\epsilon} \circ \Phi^{H_2}_{\epsilon/2} \tag{1.1}$$

is easily seen to be a symplectic map from M to M, which differs from the flow Φ^H_ϵ of H by terms of order ϵ^3. Therefore, Ψ_ϵ is a symplectic algorithm of second order on the manifold. (Higher order splitting methods can be constructed; see [18], [19].) In our application to the two problems mentioned above, we shall take as H_1 and H_2 the kinetic energy and, respectively, the potential energy. Both H_1 and H_2 are then individually integrable, and even more, exact explicit expressions of $\Phi^{H_1}_\epsilon$ and $\Phi^{H_2}_{\epsilon/2}$ in coordinates, actually very manageable ones, are easily written. (H_1 corresponds in one case to the geodesic motion on the sphere, in the other case to the Euler–Poinsot motion of the free rigid body, while in both cases H_2 is independent of momenta and thus trivially integrable.) The fact that $\Phi^{H_1}_\epsilon$ and $\Phi^{H_2}_{\epsilon/2}$ are computed exactly (up to the round-off errors) in each coordinate system automatically fulfills the above requirement (i). In other cases, like the triaxial rigid body, the kinetic energy H_1 is still integrable, but computing $\Phi^{H_1}_\epsilon$ requires handling elliptic functions. In such cases, a further convenient splitting of H_1, leading to more manageable expressions, could be preferable; we shall come back to this point in section 5.

As a final remark, we mention that the use of the splitting algorithm (1.1) to integrate the motion of a (triaxial) rigid body has already been proposed in [17], [16], [15]; the first of these references also contains numerical integrations of rigid body motions. The two approaches, however, though similar for the use of (1.1), differ in the main point; namely, we implement Ψ_ϵ in coordinates (two systems of Euler angles), while the point of view of the mentioned references is that, in order to avoid the coordinate singularities, one should implement the algorithm in a geometric way, specifically, using a Lie–Poisson integrator (coupled with the consideration of either rotational matrices or quaternions for the body orientation).

2. Theoretical description of the method.

The good energy conservation of a class of symplectic algorithms is explained in terms of the existence of an "interpolating Hamiltonian flow." We begin by showing that this is the case for the splitting method on a manifold. In this context, it is important to distinguish between Hamiltonian and locally Hamiltonian flows. (See the remark below for some comments.) Recall that a vector field X on a symplectic manifold (M, σ) is called *Hamiltonian* if the one-form $\sigma(X, \cdot)$ is exact and *locally Hamiltonian* if this one-form is only closed. In the former case, there exists a Hamiltonian function, namely, a function $H : M \to \mathbb{R}$ such that $\sigma(X, \cdot) = -dH$; in the latter case, such functions exist only locally.

By a *one-parameter family of analytic symplectic diffeomorphisms* on an analytic symplectic manifold (M, σ) we mean an analytic map

$$\Psi : I \times M \to M, \qquad (\epsilon, m) \mapsto \Psi_\epsilon(m),$$

where $I \subset \mathbb{R}$ is an interval containing zero, such that $\Psi_0 =$ identity and, for any $\epsilon \in I$, Ψ_ϵ is an analytic symplectic diffeomorphism. An elementary generalization of known results for the Euclidean case $M = \mathbb{R}^{2n}$, $\sigma = \sum_i dp_i \wedge dq_i$, gives the following proposition.

PROPOSITION 1. *Let Ψ_ϵ be a one-parameter family of analytic symplectic diffeomorphisms on a symplectic manifold (M, σ). Then, for any ϵ sufficiently near zero,*

(i) *there exists a locally Hamiltonian ("interpolating") vector field X_ϵ such that*

$$\Psi_\epsilon = \Phi_\epsilon^{X_\epsilon} + \mathcal{O}\big(e^{-1/\epsilon}\big). \tag{2.1}$$

Assume, moreover, that Ψ_ϵ is a composition of a finite number of flows of analytic Hamiltonian vector fields, each acting for a time which depends analytically on ϵ:

$$\Psi_\epsilon = \Phi_{\epsilon_1(\epsilon)}^{F_1} \circ \Phi_{\epsilon_2(\epsilon)}^{F_2} \circ \cdots \circ \Phi_{\epsilon_s(\epsilon)}^{F_s}$$

for analytic functions $\epsilon_j(\epsilon)$, which vanish at $\epsilon = 0$, and analytic functions $F_j : M \to \mathbb{R}$. Then

(ii) *the vector field X_ϵ is Hamiltonian.*

(iii) *If Ψ_ϵ is an algorithm of order $r \geq 1$ for a given Hamiltonian H, that is,*

$$\Psi_\epsilon = \Phi_\epsilon^H + \mathcal{O}(\epsilon^{r+1}),$$

then the Hamiltonian K_ϵ of X_ϵ satisfies

$$K_\epsilon = H + \mathcal{O}(\epsilon^r). \tag{2.2}$$

Proof. First note that all statements are true in the Euclidean case: statements (i) and (iii) are proven, e.g., in [5], while statement (ii) is obvious since $\mathbb{R}^{2n}$ is simply connected. The fact that statements (i) and (iii) remain true on a manifold is seen by observing that, within the quoted proof, X_ϵ and K_ϵ are constructed as formal series, the coefficients of which are commutators of vector fields and Poisson brackets of functions, respectively. Since these objects are intrinsically defined on the manifold (i.e., their local representatives commute with the change of charts), the argument of [5] applies as is to the case of a manifold. Estimates, which allow one going beyond the formal level and obtaining (2.1), are performed locally within each chart. (Analyticity is required for this.)

Now let us prove statement (ii). Consider, for simplicity, the case of the composition of two flows. Then, formally

$$\Phi_{\epsilon_1}^{X_1} \circ \Phi_{\epsilon_2}^{X_2} = \Phi_1^{\epsilon_1 X_1} \circ \Phi_1^{\epsilon_2 X_2} = \Phi_1^X$$

for a vector field X which is given, as a formal series, by the BCH formula. The fact that if X_1 and X_2 are Hamiltonian, then X is also Hamiltonian, not only locally Hamiltonian, is seen by observing that the terms of the BCH series are built up of commutators of X_1 and X_2 and recalling that the commutator of two Hamiltonian vector fields is Hamiltonian. (The Hamiltonian is the Poisson bracket of the Hamiltonians of X_1 and X_2.) The extension to any number of flows is obvious. □

The Hamiltonian K_ϵ is called the *interpolating Hamiltonian* of the algorithm Ψ_ϵ (*Modified Hamiltonian* is also used.) The existence of the interpolating Hamiltonian explains the very good energy conservation of symplectic algorithms. Up to quantities $\mathcal{O}(e^{-1/\epsilon})$, which can easily be made smaller than the round-off even in multiple precision [2] by taking the step-size ϵ small enough, the iteration of the symplectic integrator follows exactly the flow of the nearby vector field X_ϵ. If this vector field is Hamiltonian (not just locally Hamiltonian), then its Hamiltonian K_ϵ is preserved by the algorithm up to an exponentially small error. Accumulation of these extremely small errors produces acceptable overall effects on significantly long integration times.[3]

[3] In *special* situations, the errors on the energy do not accumulate; see [5] and, for numerical results, [2].

$$\begin{tikzcd}
 & M \arrow[dl, "z"'] \arrow[dr, "z'"] & \\
\mathbb{R}^{2n} \arrow[rr, "\mathcal{C}"] \arrow[d, "\psi_\epsilon"] & & \mathbb{R}^{2n} \arrow[d, "\psi'_\epsilon"] \\
\mathbb{R}^{2n} \arrow[rr, "\mathcal{C}"] & & \mathbb{R}^{2n}
\end{tikzcd}$$

FIG. 1. *The local representatives of the algorithm commute with the change of charts.*

Remark. The distinction between the existence of a Hamiltonian interpolating vector field and a locally Hamiltonian one is important, even though so far it has not drawn much attention. (For some results, see [5] and [8].) A locally Hamiltonian interpolating vector field does not ensure energy conservation: any local interpolating Hamiltonian is conserved by the algorithm, but this conservation law might be broken by orbits winding around the manifold. Of course, the problem does not exist in the Euclidean case because of the simple connectedness of $\mathbb{R}^{2n}$. However, simple examples demonstrate that, on a manifold, some condition is necessary to ensure the Hamiltonian character of the interpolating vector field (e.g., $M = \mathbb{R} \times S^1 \ni (I, \varphi)$, $\sigma = dI \wedge d\varphi$, $\Psi_\epsilon(I, \varphi) = (I + \epsilon, \varphi)$).

By Proposition 1, the algorithm Ψ_ϵ on M defined by (1.1) has a global interpolating Hamiltonian $K_\epsilon = H_1 + H_2 + \mathcal{O}(\epsilon^2)$. This fact alone makes clear that, however implemented in local coordinates, this algorithm will exhibit good energy conservation.

For the sake of the discussion, assume that M has a symplectic atlas which consists of just two charts with coordinates $z = (q, p)$ and $z' = (q', p')$. Let $\mathcal{C}$ be the change of chart, so that $z' = \mathcal{C} \circ z$. If $h = h_1 + h_2$ and $h' = h'_1 + h'_2$ are the local representatives of the Hamiltonian $H = H_1 + H_2$ in the two charts,[4] then the local representatives ψ_ϵ and ψ'_ϵ of the map Ψ_ϵ are

$$\psi_\epsilon = \Phi^{h_2}_{\epsilon/2} \circ \Phi^{h_1}_{\epsilon} \circ \Phi^{h_2}_{\epsilon/2}, \qquad \psi'_\epsilon = \Phi^{h'_2}_{\epsilon/2} \circ \Phi^{h'_1}_{\epsilon} \circ \Phi^{h'_2}_{\epsilon/2}.$$

By construction, these two maps commute with the change of chart, i.e.,

$$\mathcal{C} \circ \psi_\epsilon = \psi'_\epsilon \circ \mathcal{C} \tag{2.3}$$

(see Figure 1). It follows that *we can freely switch from one chart to the other and keep following the iteration of the map Ψ_ϵ on the manifold.* In other words, this changing-chart method is an actual implementation of the algorithm (1.1) on the manifold. Note that the coordinate systems used, and the adopted criterion for when to switch

[4] This means that the restriction of H to the first chart equals $h \circ z$, etc. Note that $h = h' \circ \mathcal{C}$ in the intersection of the chart domains.

charts, are completely immaterial: different choices may produce more or less simple and efficient implementations but do not affect the algorithm.[5] The implementations of the changing-chart methods mentioned in the introduction did not satisfy (2.3), and this makes the difference.

In the next section, we illustrate this procedure on the two cases mentioned in the introduction, in which H_1 is the kinetic energy of an integrable system and H_2 is a positional potential. The flow of H_2 is trivially integrated: if $h_2(q)$ is the local representative of H_2 in a chart with coordinates (p,q), then

$$(p,q) \;\mapsto\; \left(p + t\,\frac{\partial h_2}{\partial q}(q), q\right).$$

The local representatives of H_1 depend on both coordinates and momenta, but since H_1 is integrable, in principle one knows the solution in each chart. The crucial step, on which the efficiency of the algorithm relies, is to have an analytical representation of the flow $\Phi_\epsilon^{H_1}$ of the kinetic energy which is suitable for numerical computations.

3. Two examples.

3.1. The point constrained to the sphere.

As a first elementary example, we consider a point constrained to the unit sphere in $\mathbb{R}^3$ subject to positional potential forces. The flow $\Phi_\epsilon^{H_1}$ of the kinetic energy is the geodesic flow on the sphere and is easily written in any system of coordinates. We used two systems of spherical coordinates and the corresponding momenta, that we denote $(\varphi, \theta, p_\varphi, p_\theta)$ and $(\varphi', \theta', p'_\varphi, p'_\theta)$, relative to two orthogonal frames $\{e_x, e_y, e_z\}$ and, respectively, $\{e'_x, e'_y, e'_z\} = \{e_x, -e_z, e_y\}$. The change of charts are given by

$$\begin{cases} \cos\theta = -\sin\theta'\,\sin\varphi', \\ \sin\theta = \sqrt{1-\cos^2\theta}, \\ \cos\varphi = \frac{\sin\theta'\,\cos\varphi'}{\sin\theta}, \\ \sin\varphi = \frac{\cos\theta'}{\sin\theta}, \\ p_\theta = \frac{\cos\theta'\,\sin\varphi'}{\sin\theta}\,p'_\theta + \frac{\cos\varphi'}{\sin\theta\,\sin\theta'}\,p'_\varphi, \\ p_\varphi = -\cos\varphi'\,p'_\theta + \frac{\sin\varphi'\,\cos\theta'}{\sin\theta'}\,p'_\varphi, \end{cases} \qquad \begin{cases} \cos\theta' = \sin\theta\,\sin\varphi, \\ \sin\theta' = \sqrt{1-\cos^2\theta'}, \\ \cos\varphi' = \frac{\sin\theta\,\cos\varphi}{\sin\theta'}, \\ \sin\varphi' = -\frac{\cos\theta}{\sin\theta'}, \\ p'_\theta = -\frac{\cos\theta\,\sin\varphi}{\sin\theta'}\,p_\theta - \frac{\cos\varphi}{\sin\theta\,\sin\theta'}\,p_\varphi, \\ p'_\varphi = \cos\varphi\,p_\theta - \frac{\sin\varphi\,\cos\theta}{\sin\theta}\,p_\varphi. \end{cases}$$

Due to rotational invariance, both the kinetic energy and the geodesic flow have the same expression in the two charts, so we write them down only in the first chart. The kinetic energy is

$$h_1(\varphi, \theta, p_\varphi, p_\theta) = \frac{p_\theta^2}{2} + \frac{p_\varphi^2}{2\sin^2\theta}.$$

The geodesic flow is determined either by integrating Hamilton's equations or just by observing that it consists of a uniform translation along a great circle. Consider the initial datum $(\varphi(0), \theta(0), p_\varphi(0), p_\theta(0)) = (\varphi_0, \theta_0, J, P_0)$ and denote

$$G = \sqrt{P_0^2 + \Big(\frac{J}{\sin\theta_0}\Big)^2}, \qquad \hat{P}_0 = \frac{P_0}{G}, \qquad \hat{J} = \frac{J}{G}\,;$$

[5] In fact, it is not even necessary that the coordinates z and z' be symplectic, even though this is likely to be the simplest choice. If the coordinates are not symplectic, one should repeat the previous argument by considering not the Hamiltonians and their local representatives but the corresponding Hamiltonian vector fields.

G is the norm of the angular momentum and, as p_φ, is an integral of motion. (This is the reason why we do not append the subscript 0 to G and J.) Then the solution $(\varphi(t), \theta(t), p_\varphi(t), p_\theta(t))$ of the system with Hamiltonian h_1 can be written as

$$\begin{pmatrix} \cos\theta(t) \\ \frac{p_\theta(t)}{G}\sin\theta(t) \end{pmatrix} = \begin{pmatrix} \cos Gt & -\sin Gt \\ \sin Gt & \cos Gt \end{pmatrix} \begin{pmatrix} \cos\theta_0 \\ \hat{P}_0 \sin\theta_0 \end{pmatrix},$$

$$\begin{pmatrix} \cos\varphi(t) \\ \sin\varphi(t) \end{pmatrix} = \frac{1}{\sin\theta(t)} \begin{pmatrix} C & -S \\ S & C \end{pmatrix} \begin{pmatrix} \cos\varphi_0 \\ \sin\varphi_0 \end{pmatrix}$$

for

$$\begin{aligned} C &= \sin\theta_0 \, \cos Gt + \hat{P}_0 \, \cos\theta_0 \, \sin Gt, \\ S &= \frac{\hat{J}}{\sin\theta_0} \, \sin Gt. \end{aligned}$$

3.2. The symmetric rigid body. We now consider the symmetric rigid body with a fixed point. (Here symmetric just means that two of the three moments of inertia relative to the fixed point are equal.) Let $\{e_1, e_2, e_3\}$ be an orthogonal frame centered on the body's fixed point with axes oriented like the principal axes of inertia, and let $I_1 = I_2$ and I_3 be the corresponding principal moments of inertia. In order to obtain an atlas for $T^*\mathrm{SO}(3)$, we consider two systems of standard Euler angles (see Figure 2), say, $q = (\varphi, \psi, \theta)$ and $q' = (\varphi', \psi', \theta')$, relative to different spatial frames $\{e_x, e_y, e_z\}$ and, respectively, $\{e'_x, e'_y, e'_z\} = \{e_z, e_x, e_y\}$. These coordinates and their conjugate momenta $p = (p_\varphi, p_\psi, p_\theta)$ and $p' = (p'_\varphi, p'_\psi, p'_\theta)$ are related by the equations

$$\begin{cases} \cos\theta = \sin\theta' \, \sin\varphi', \\ \sin\theta = \sqrt{1 - \cos^2\theta}, \\ \cos\varphi = -\cos\theta' / \sin\theta, \\ \sin\varphi = -\sin\theta' \, \cos\varphi' / \sin\theta, \\ \cos\psi = -(\cos\varphi' \, \sin\psi', \\ \qquad + \cos\theta' \, \sin\varphi' \, \cos\psi') / \sin\theta, \\ \sin\psi = (\cos\varphi' \, \cos\psi' \\ \qquad - \cos\theta' \, \sin\varphi' \, \sin\psi') / \sin\theta, \\ p_\theta = -\frac{\sin\varphi' \cos\theta'}{\sin\theta} p'_\theta, \\ \qquad - \frac{\cos\varphi'}{\sin\theta \sin\theta'} (p'_\varphi + \cos\theta' p'_\psi), \\ p_\varphi = \cos\varphi \, p'_\theta - \frac{\sin\varphi'}{\sin\theta'} (\cos\theta' p'_\varphi + p'_\psi), \\ p_\psi = p'_\psi, \end{cases} \qquad \begin{cases} \cos\theta' = - \sin\theta \, \cos\varphi, \\ \sin\theta' = \sqrt{1 - \cos^2\theta'}, \\ \cos\varphi' = -\sin\theta \sin\varphi / \sin\theta', \\ \sin\varphi' = \cos\theta / \sin\theta', \\ \cos\psi' = -(\sin\varphi \, \sin\psi \\ \qquad - \cos\theta \, \cos\varphi \, \cos\psi) / \sin\theta', \\ \sin\psi' = (\sin\varphi \, \cos\psi \\ \qquad + \cos\theta \, \cos\varphi \, \sin\psi) / \sin\theta', \\ p'_\theta = \frac{\cos\varphi \, \cos\theta}{\sin\theta'} p_\theta \\ \qquad - \frac{\sin\varphi}{\sin\theta \, \sin\theta'} (p_\varphi - \cos\theta \, p_\psi), \\ p'_\varphi = \sin\varphi \, p_\theta \\ \qquad + \frac{\cos\varphi}{\sin\theta} (\cos\theta p_\varphi + p_\psi), \\ p'_\psi = p_\psi. \end{cases}$$

Here too the kinetic energy has the same form in both coordinate systems, precisely

$$h_1(q, p) = \frac{p_\theta^2}{2I_1} + \frac{(p_\varphi - p_\psi \cos\theta)^2}{2I_1 \sin^2\theta} + \frac{p_\psi^2}{2I_3}.$$

The components of the angular momentum along the axes e_z and e_3, namely, p_φ and p_ψ, are constants of motion of the free system; their values will hereafter be denoted J and L, respectively. Also the norm G of the angular momentum vector is constant; one has

$$G = \sqrt{p_\theta^2 + p_\psi^2 + \Big(\frac{p_\varphi - p_\psi \cos\theta}{\sin\theta}\Big)^2}. \tag{3.1}$$

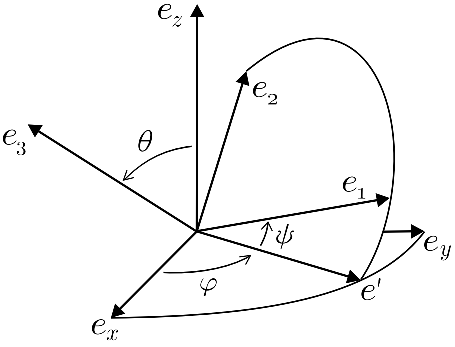


FIG. 2. *The Euler angles φ, ψ, θ. (e' is the unit vector in the direction of $e_z \times e_3$.)*

In the classical textbooks and manuals we consulted, we found the expression of the solution of the Euler–Poinsot system only in special systems of Euler angles with the spatial axis e_z parallel to the angular momentum. Since this is not sufficient for our purposes, we computed the solution in a generic system of Euler angles. (This is elementary in the symmetric case.)

PROPOSITION 2. *Consider an initial datum $(\varphi(0), \psi(0), \theta(0), p_\varphi(0), p_\psi(0), p_\theta(0)) = (\varphi_0, \psi_0, \theta_0, J, L, P_0)$, and let G be the corresponding norm of the angular momentum. Denote*

$$\hat{J} = \frac{J}{G}, \qquad \hat{L} = \frac{L}{G}, \qquad \hat{P}_0 = \frac{\hat{P}_0}{G}, \qquad \Omega = \frac{G}{I_1}, \qquad \Lambda = \frac{I_1 - I_3}{I_1 I_3} L.$$

Then, for all t such that $\sin\theta(t) \neq 0$, the solution of the system of Hamiltonian h_1 is given by

$$\begin{pmatrix} \cos\theta(t) - \hat{J}\hat{L} \\ \frac{p_\theta(t)}{G}\sin\theta(t) \end{pmatrix} = \begin{pmatrix} \cos\Omega t & -\sin\Omega t \\ \sin\Omega t & \cos\Omega t \end{pmatrix} \begin{pmatrix} \cos\theta_0 - \hat{J}\hat{L} \\ \hat{P}_0 \sin\theta_0 \end{pmatrix}, \tag{3.2}$$

$$\begin{pmatrix} \cos\varphi(t) \\ \sin\varphi(t) \end{pmatrix} = \frac{1}{\sin\theta_0 \, \sin\theta} \begin{pmatrix} C(\hat{J},\hat{L}) & S(\hat{J},\hat{L}) \\ -S(\hat{J},\hat{L}) & C(\hat{J},\hat{L}) \end{pmatrix} \begin{pmatrix} \cos\varphi_0 \\ \sin\varphi_0 \end{pmatrix}, \tag{3.3}$$

$$\begin{aligned} \begin{pmatrix} \cos\psi(t) \\ \sin\psi(t) \end{pmatrix} = \frac{1}{\sin\theta_0 \, \sin\theta} & \begin{pmatrix} C(\hat{L},\hat{J}) & S(\hat{L},\hat{J}) \\ -S(\hat{L},\hat{J}) & C(\hat{L},\hat{J}) \end{pmatrix} \\ & \times \begin{pmatrix} \cos\Lambda t & -\sin\Lambda t \\ \sin\Lambda t & \cos\Lambda t \end{pmatrix} \begin{pmatrix} \cos\psi_0 \\ \sin\psi_0 \end{pmatrix}, \end{aligned} \tag{3.4}$$

where

$$\begin{aligned} C(x,y) = {} & y(y - x\cos\theta_0) \\ & + \left[\sin^2\theta_0 - y(y - x\cos\theta_0)\right]\cos\Omega t + \hat{P}_0 \sin\theta_0 \cos\theta_0 \sin\Omega t, \end{aligned} \tag{3.5}$$

$$S(x,y) = (1 - \cos\Omega t)\hat{P}_0 \sin\theta_0 y + (x - y\cos\theta_0)\sin\Omega t. \tag{3.6}$$

A proof of this proposition is given in the appendix.

Remark. If the algorithm is regarded as mapping sine and cosine of the angles at time zero to sine and cosine of the angles at time t, then only four trigonometric functions must be computed, namely, sine and cosine of Ωt and of Λt; all other operations are algebraic (elementary operations and a few square roots). Developing an efficient, accurate algorithm for the symmetric rigid body was the original motivation of this work. We were interested in performing long-time integrations of rigid bodies under generic potential force fields, so as to investigate the presence of the chaotic behavior predicted in the works [3] and [4]. We will publish the results of these integrations somewhere else. We just mention here that the algorithm appeared to be simple, efficient, and stable. Overall, it seemed to perform exceptionally well.

4. Numerical tests and comparisons. Our algorithm exhibits the typical very good conservation of energy of the symplectic algorithms which possess an interpolating Hamiltonian.

The best way of illustrating this fact and of enlightening the mechanism underneath is probably by means of the comparison with a slightly (but crucially) different algorithm. Specifically, we consider within each chart a (generalized) leapfrog, namely, the map $\tilde{\psi}^h_\epsilon = \tilde{\psi}^h_{II} \circ \tilde{\psi}^h_I$, where $\tilde{\psi}^h_I : (q,p) \mapsto (q^I, p^I)$ and $\tilde{\psi}^h_{II} : (q^I, p^I) \mapsto (q^{II}, p^{II})$ are (implicitly) defined by

$$\tilde{\psi}^h_I : \begin{cases} p^I = p - \frac{\epsilon}{2}\frac{\partial h}{\partial q}(p^I, q), \\ q^I = q + \frac{\epsilon}{2}\frac{\partial h}{\partial p}(p^I, q), \end{cases} \qquad \tilde{\psi}^h_{II} : \begin{cases} q^{II} = q^I + \frac{\epsilon}{2}\frac{\partial h}{\partial p}(p^I, q^{II}), \\ p^{II} = p^I - \frac{\epsilon}{2}\frac{\partial h}{\partial q}(p^I, q^{II}). \end{cases} \tag{4.1}$$

The standard leapfrog (or Verlet) algorithm is obtained when $h(q,p)$ is separated, namely, $h(q,p) = h_1(p) + h_2(q)$; in such a case, as we have already mentioned, this algorithm also coincides with the splitting algorithm (1.1). In the two cases considered above, the point on the sphere in polar coordinates and the symmetric rigid body in Euler angles, the Hamiltonians are not separated but the generalized leapfrogs $\tilde{\psi}^h_\epsilon$ and $\tilde{\psi}^{h'}_\epsilon$ in the two charts are explicit anyway.

As is well known, the generalized leapfrog $\tilde{\psi}^h_\epsilon$ in each chart is symplectic, possesses an interpolating Hamiltonian, and differs from Φ^h_ϵ by $\mathcal{O}(\epsilon^3)$. *However, the two generalized leapfrogs do not commute with the change of chart, namely,* $\mathcal{C} \circ \tilde{\psi}^h_\epsilon \neq \tilde{\psi}^{h'}_\epsilon \circ \mathcal{C}$. This means that the two algorithms are not the local representatives of a single algorithm on the manifold. In particular, *the interpolating Hamiltonians of the individual algorithms in the two charts are not the local representatives of a single function on the manifold.* The consequence is that each change of chart introduces a change of order ϵ^2 in the conserved quantity (i.e., the interpolating Hamiltonian of the local algorithm), and this results typically in a progressive drift of the energy proportional to ϵ^2 and to the number of chart changes.[6]

This is illustrated in Figure 3, which refers to the simple case of the spherical pendulum, namely, a heavy point on the sphere; the potential energy in the two charts are $h_2 = \cos\theta$ and $h'_2 = -\sin\theta' \sin\varphi'$. All figures report the relative error in the energy along the same orbit, which has been computed with the splitting algorithm (solid line) and switching between the two generalized leapfrogs (dotted line). The two methods are implemented using the same coordinate systems, the same criterion for switching charts, and the same integration step. The superimposed dots identify the instants at which the chart is switched. In all figures, the initial datum is $(\varphi, \theta, p_\varphi, p_\theta) = (\pi/3, \pi/3, 1.1, 0.35)$ (in the first chart) and the integration step is $\epsilon = 0.05$. Figures 3(a), 3(b), and 3(c) differ only for the total integration time, which equals 50, 400, and 1000 time-steps, respectively. The difference between the two methods is evident: switching charts has no consequence for the splitting algorithm, while it produces accumulation of errors for the pair of generalized leapfrogs.

[6] This mechanism was already observed in [13].

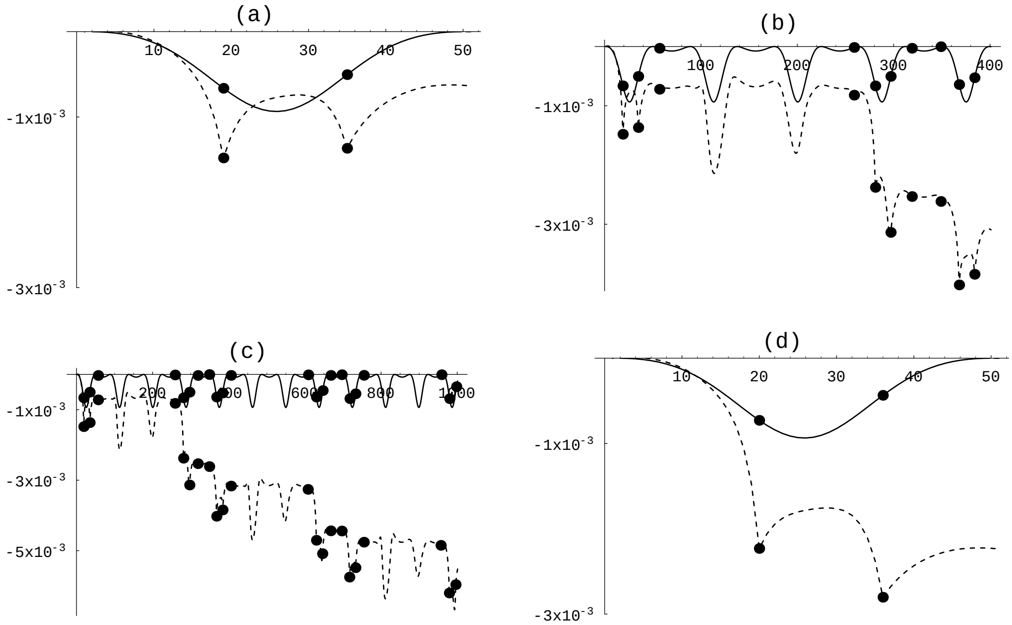


FIG. 3. *Error in the energy along an orbit of the spherical pendulum for our algorithm (solid line) and for a pair of leapfrogs (dotted line). Charts are switched when* $\arcsin\theta$ *reaches* 0.7 *in* (a), (b)*, and* (c)*, and* 0.5 *in* (d).

In Figures 3(a), 3(b), and 3(c), the chart was changed whenever the colatitude θ (or θ') reached the value arcsin(0.7). In Figure 3(d) we used a different criterion for changing charts, so as to force the change of charts at different instants. Specifically, we changed chart when the colatitude reached arcsin(0.5). Comparison with Figure 3(a) indicates that this has a consequence for the pair of generalized leapfrogs, but not for our algorithm, which is completely transparent to the change of charts.

The same behavior is found in the symmetric rigid body. Figure 4 refers to a rigid body with moments of inertia $I_1 = I_2 = 1$ and $I_3 = 2/3$. The potential energy in the two Euler angles charts introduced in the previous section is

$$\begin{aligned} h_2(q,p) &= \sin\theta\cos\varphi + (\sin\theta\cos\psi)^2, \\ h_2'(q',p') &= -\sin\theta'\sin\varphi' + (\sin\varphi\sin\psi - \cos\varphi\cos\psi\cos\theta)^2. \end{aligned}$$

Figures 4(a)–(c) show the energy error on 120 time-steps for the orbit with initial datum $(\varphi,\psi,\theta,p_\varphi,p_\psi,p_\theta) = (0.5,-0.45,\pi/3,2.2,0.8,1.3)$; the integration step is 0.05 in all cases, while the chart is switched when $\sin\varphi$ reaches the values 0.7, 0.5, 0.2, respectively. Figure 4(d) is the same as Figure 4(a) but for a longer integration time.

Remark. Figure 3 indicates that, even within a given chart, the error on the energy is smaller for the splitting algorithm than for the generalized leapfrog. This is largely due to the fact that, within our algorithm, the flow of the kinetic energy is computed exactly, which produces a more stable algorithm. In fact, the leapfrog has a factor $(\sin\theta)^{-3}$ which degrades the precision already at moderate distances from the singularities. The same situation is met with the symmetric rigid body: Figures 4(a)–

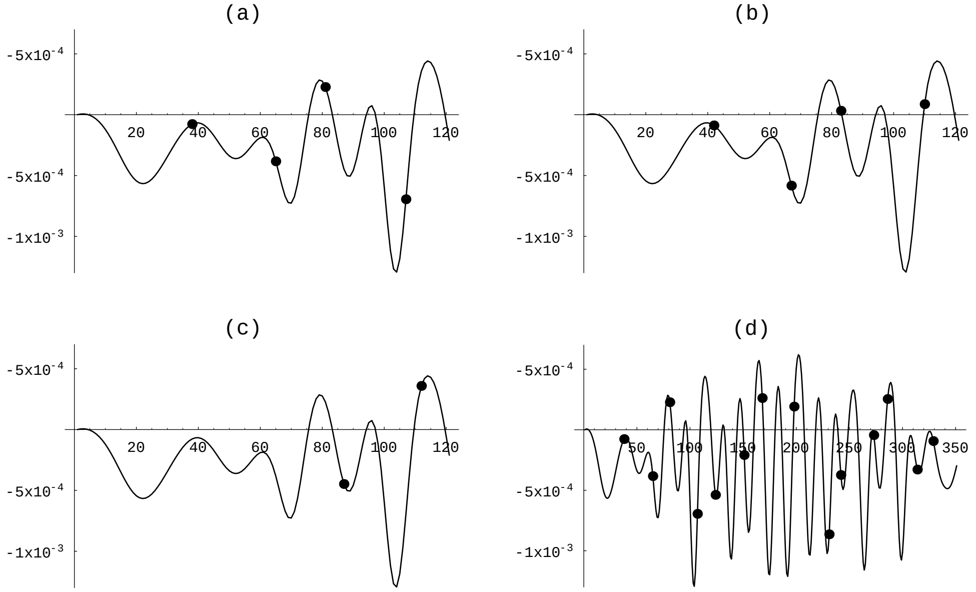


FIG. 4. *Error in the energy along an orbit of the symmetric rigid body in an external potential computed with our algorithm. The chart is switched when* $\arcsin\theta$ *reaches* 0.7 *in* (a) *and* (d)*,* 0.5 *in* (b)*, and* 0.2 *in* (c).

(c) indicate that the energy conservation does not get worse when the singularities are approached (up to arcsin(0.2) in the case of Figure 4(c)). In this case, too, the energy error with a pair of leapfrogs is much bigger.

As a further evidence that the algorithm, though written in coordinates, is nevertheless coordinate independent, let us propose another elementary test. Consider a Lagrange top that is a heavy symmetric rigid body, and let e_g be the unit vector in the direction of gravity. The component of the angular momentum along e_g, that we shall denote m, is then a constant of motion. Quite clearly, if we use a chart of Euler angles, relative to a spatial frame $\{e_x, e_y, e_z\}$ with $e_g = e_z$, then $m = p_\varphi$, and since the Hamiltonian is independent of φ, m is exactly conserved. Now, *since the algorithm commutes with the change of charts, the same must be true,* up to the round-off errors, *in any other chart,* for example, a chart of Euler angles relative to a spatial frame $\{e_x, e_y, e_z\}$ such that e_g has no special direction. This is precisely what we found numerically: working in double precision, for several choices of e_g and of the initial data, the error we found in m, after many thousands of time-steps and hundreds of chart changes, was at most of order 10^{-13}, independently of the step-size, and reduced to 10^{-30} in quadruple precision. Such an especially good conservation law is not obvious, for a second order algorithm, and reflects in an essential way its geometrical nature, while the implementation in coordinates is irrelevant.

5. Conclusions. In our opinion, the method described here has potentially many advantages and is worthy of further consideration.

A crucial fact which is missing in the present analysis is a comparison with other methods. We made some comparisons with RATTLE in the case of the point on the sphere and found that the two algorithms perform in a comparable way. For the same step-size, RATTLE was somewhat faster (by about a factor 1.5 to 2, depending on the potential), but our algorithm was more stable. (The same energy conservation was obtained, typically, with a step-size bigger by a factor 1.5.)

As is clear, this comparison is not conclusive. In the case of the rigid body, a thorough comparison with other methods (RATTLE [14], [10], Lie–Poisson integrators [17], [15], [16], and the methods of [11], [12]) is necessary and is in our programs. As already remarked, in the case of the rigid body, RATTLE is not explicit and the codimension $2(N-n)$ is 12; hence, the algorithm must handle 18 variables and in addition to a total of 12 Lagrange multipliers.[7] Our algorithm instead uses only six variables, and for a symmetric rigid body it is explicit and fully manageable, requiring only the evaluation of four trigonometric functions at each time-step. We expect in this case for our algorithm to be faster. (We guess significantly faster.)

The situation is more delicate for the triaxial rigid body. Since the representation of the Euler–Poinsot flow in Euler angles uses elliptic functions, the exact integration of the flow of the kinetic energy might significantly lengthen the integration time. Alternatives are based on the further splitting of the Euler–Poinsot Hamiltonian as the sum of two or more integrable Hamiltonians, each of which is (more) easily integrated. This approach was already proposed and used in [17], [15], [16], though, as already remarked, not using coordinates. Convenient splittings of the Euler–Poinsot system are expected to produce quite manageable algorithms with good (perhaps slightly worse) stability. A careful analysis in our opinion is necessary to evaluate the different possibilities.

We prefer to leave these tests and comparisons to a dedicated work. In addition to such tests, of course, it is also important to experiment with the performance of the algorithm in concrete problems, especially when accuracy is important. As we have already remarked in section 3, we are presently working on the symmetric rigid body in order to evaluate the degree of optimality of some previous perturbative studies; such an investigation is numerically delicate, but the preliminary results are rather satisfying. Other studies, for example in molecular dynamics, would be of great interest.

In this regard, we should mention a further positive feature of the splitting method. The estimate (2.2) on the interpolating Hamiltonian given in Proposition 1 can be further detailed: $H-K_\epsilon$ is in fact of the order of ϵ^r times the biggest of (some norm of) the Poisson brackets $\{F_i,F_j\}$, $i,j=1,\dots,s$. Now consider the case in which one has a nearly integrable Hamiltonian H, say,

$$H = H_1 + \mu H_2, \qquad \mu \ll 1.$$

If one can integrate exactly the flow of H_1 and H_2, then it is possible to use the second order splitting algorithm $\Phi^{H_2}_{\epsilon/2}\circ\Phi^{H_1}_{\epsilon}\circ\Phi^{H_2}_{\epsilon/2}$. In this case, the interpolating Hamiltonian gets the form

$$K_\epsilon = H_1 + \mu\big[H_2 + \mathcal{O}(\epsilon^2)\big];$$

i.e., the correction due to the algorithm remains small with respect to the perturbation (as is obviously necessary in a perturbation problem) for small ϵ, with no need to

[7] There are formulations of RATTLE which use fewer coordinates (e.g., the four Euler parameters; see [9]), so that these numbers can be drastically reduced, but to our knowledge they are not symplectic.

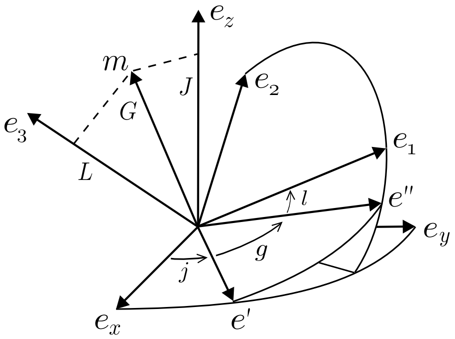


FIG. 5. *The action-angle coordinates of the symmetric Euler–Poinsot system; m is the angular momentum, and e' and e'' are unit vectors in the directions of $e_z \times m$ and, respectively, of $m \times e_3$.*

reduce ϵ dependently on μ. This fact is especially useful if one is interested in exploring numerically the case $\mu \to 0$. Unfortunately, in the case of a triaxial body, this positive feature is destroyed by a splitting of the Euler–Poinsot Hamiltonian, so for very small μ the exact integration of H_1 via elliptic functions could become competitive.

Anyhow, beyond these special applications, it seems to us that the possibility described here of constructing changing-chart symplectic algorithms on manifolds, just by implementing splitting algorithms in coordinates, has a character of simplicity which makes it of interest by itself, even independently of its performance in specific situations.

Appendix. Proof of Proposition 2. Probably the simplest way of obtaining the expressions for the flow of the symmetric Euler–Poinsot system given in Proposition 2 is by referring to the action-angle coordinates (G, L, J, g, l, j) of the system in which the flow is trivial. These coordinates, which are also known as Andoyer or Deprit coordinates, are defined as follows: G, L, and J are, respectively, the length, the component along the symmetry axis, and the component along the z-axis of the angular momentum vector; their three conjugate angles g, l, and j are defined as shown in Figure 5, where e_1, e_2, e_3 is an orthonormal frame fixed in the body, with e_3 along the symmetry axis. Clearly, these coordinates are singular when the angular momentum is aligned with either e_3 or e_z, namely, for either $G = \pm L$ or $G = \pm J$.

The advantage of the action-angle coordinates is that the flow of the symmetric Euler–Poinsot system is linear in these coordinates. In fact, the Hamiltonian is

$$H(G, L) = \frac{1}{2I_1}\, G^2 + \frac{I_1 - I_3}{2I_1 I_3}\, L^2,$$

and hence the flow is given by G, L, J, j constant and

$$g(t) = g_0 + \Omega\, t, \qquad l(t) = l_0 + \Lambda\, t, \tag{A.1}$$

where $\Omega = G/I_1$ and $\Lambda = \frac{I_1 - I_3}{2I_1 I_3} L$. Obviously, this solution is valid as long as the system stays away from the singularities of the action-angle coordinates; this limitation will be eliminated below.

In order to prove the proposition, we convert the flow written in action-angle coordinates to Euler angles. The formulas relating the Euler angles to the action-angle coordinates which are needed here are as follows:

$$G = \sqrt{p_\theta^2 + p_\psi^2 + \Big(\frac{p_\varphi - p_\psi \cos\theta}{\sin\theta}\Big)^2}, \qquad L = p_\psi, \qquad J = p_\varphi, \tag{A.2}$$

$$\cos g = \frac{\hat J\,\hat L - \cos\theta}{\sqrt{1-\hat J^2}\sqrt{1-\hat L^2}}, \qquad \sin g = -\,\frac{p_\theta \sin\theta}{G\sqrt{1-\hat J^2}\sqrt{1-\hat L^2}}, \tag{A.3}$$

$$j = \varphi + \alpha, \qquad l = \psi + \beta, \tag{A.4}$$

where $\hat J = J/G$, $\hat L = L/G$, the angle α satisfies

$$\cos\alpha = \frac{\hat L - \hat J\cos\theta}{\sin\theta\sqrt{1-\hat J^2}}, \qquad \sin\alpha = \frac{p_\theta\,\sin\theta}{G\sqrt{G^2-J^2}},$$

and the angle β satisfies these same expressions with $\hat J$ and $\hat L$ interchanged. These equalities are easily proven. (See, e.g., [3], where a few of them are proven.) In what follows, we use the notation

$$R_\delta = \begin{pmatrix} \cos\delta & -\sin\delta \\ \sin\delta & \cos\delta \end{pmatrix}.$$

Equations (A.3) and (A.1) together imply

$$\begin{pmatrix} \hat J\hat L - \cos\theta(t) \\ \frac{p_\theta(t)}{G}\sin\theta(t) \end{pmatrix} = R_{\Omega t}\begin{pmatrix} \hat J\hat L - \cos\theta(t) \\ \frac{p_\theta(0)}{G}\sin\theta(0) \end{pmatrix},$$

namely, (3.2). Next, since $j = \varphi + \alpha$, one has

$$\begin{pmatrix} \cos\varphi(t) \\ \sin\varphi(t) \end{pmatrix} = R_{\alpha(t)}\begin{pmatrix} \cos j_0 \\ \sin j_0 \end{pmatrix} = R_{\alpha(t)}\,R_{-\alpha(0)}\begin{pmatrix} \cos\varphi_0 \\ \sin\varphi_0 \end{pmatrix};$$

a trivial although somewhat lengthy, computation shows that the product of the two matrices can be written in the form given in (3.3). Finally,

$$\begin{pmatrix} \cos\psi(t) \\ \sin\psi(t) \end{pmatrix} = R_{\beta(t)}\begin{pmatrix} \cos l(t) \\ \sin l(t) \end{pmatrix} = R_{\beta(t)}\,R_{-\beta(0)}\,R_{\Lambda t}\begin{pmatrix} \cos\psi_0 \\ \sin\psi_0 \end{pmatrix},$$

where we have used the fact that planar rotations commute; (3.4) is now obtained by observing that $R_{\beta(t)}R_{-\beta(0)}$ equals $R_{\alpha(t)}R_{-\alpha(0)}$ after switching $\hat J$ and $\hat L$.

In this way, we have proven the expressions (3.3)–(3.4) for times t such that the solution remains within the domain of the action-angle coordinates for all times $0 < t' < t$. However, analyticity in t of the solution ensures that these expressions are in fact valid for all t such that $\sin\theta(t) \neq 0$. This remains true (continuity is sufficient for this) even if $\sin\theta$ had vanished at some earlier time $t' < t$.

Acknowledgments. We wish to thank Ben Leimkuhler and Massimiliano Guzzo for useful discussions.

REFERENCES

[1] J. Barojas and D. Levesque, *Simulation of diatomic homonuclear liquids*, Phys. Rev. A, 7 (1973), pp. 1092–1105.

[2] G. Benettin and F. Fassò, *From Hamiltonian perturbation theory to symplectic integrators and back*, Appl. Numer. Math., 29 (1999), pp. 73–87.

[3] G. Benettin and F. Fassò, *Fast rotations of the rigid body: A study by Hamiltonian perturbation theory. Part* I, Nonlinearity, 9 (1996), pp. 137–186.

[4] G. Benettin, F. Fassò, and M. Guzzo, *Fast rotations of the rigid body: A study by Hamiltonian perturbation theory. Part* II: *Gyroscopic rotations*, Nonlinearity, 10 (1997), pp. 1695–1717.

[5] G. Benettin and A. Giorgilli, *On the Hamiltonian interpolation of near to the identity symplectic mappings, with application to symplectic integration algorithms*, J. Statist. Phys., 74 (1994), pp. 1117–1144.

[6] D. Evans, *On the representation of orientation space*, Molecular Phys., 34 (1977), pp. 317–325.

[7] D. Evans and S. Murad, *Singularity free algorithms for molecular dynamics simulation of rigid polyatoms*, Molecular Phys., 34 (1977), pp. 327–331.

[8] E. Hairer, *Backward analysis of numerical integrators and symplectic methods*, Ann. Numer. Math., 1 (1994), pp. 107–132.

[9] E.J. Haug, *Computer Aided Kinematics and Dynamics of Mechanical Systems, Part* I: *Basic Methods*, Allyn and Bacon, Boston, 1989.

[10] A. Kol, B. Laird, and B. Leimkuhler, *A symplectic method for rigid–body molecular simulation*, J. Chem. Phys., 107 (1997), pp. 2580–2588.

[11] D. Lewis and J.C. Simò, *Conserving algorithms for the dynamics of Hamiltonian systems on Lie groups*, J. Nonlinear Sci., 4 (1994), pp. 253–299.

[12] D. Lewis and J.C. Simò, *Conserving algorithms for the N dimensional rigid body*, Fields Inst. Commun., 10 (1996), pp. 121–139.

[13] B. Leimkuhler and S. Reich, *Symplectic integration of constrained Hamiltonian systems*, Math. Comp., 63 (1994), pp. 589–605.

[14] B. Leimkuhler and R. Skeel, *Symplectic numerical integrators in constrained Hamiltonian systems*, J. Comput. Phys., 112 (1994), pp. 117–125.

[15] R.I. McLachlan, *Explicit Lie-Poisson integration and the Euler equations*, Phys. Rev. Lett., 71 (1993), pp. 3043–3046.

[16] S. Reich, *Momentum conserving symplectic integrators*, Phys. D, 76 (1994), pp. 375–383.

[17] J. Touma and J. Wisdom, *Lie–Poisson integrators for rigid body dynamics in the solar system*, Astronom. J., 107 (1994), pp. 1189–1202.

[18] H. Yoshida, *Construction of higher order symplectic integrators*, Phys. Lett. A, 150 (1990), pp. 262–268.

[19] H. Yoshida, *Recent progress in the theory and application of symplectic integrators*, Celestial Mech. Dynam. Astronom., 56 (1993), pp. 27–43.

SIAM J. SCI. COMPUT.
Vol. 23, No. 4, pp. 1204–1225

THE METHOD OF REGULARIZED STOKESLETS[*]

RICARDO CORTEZ[†]

Abstract. A numerical method for computing Stokes flows in the presence of immersed boundaries and obstacles is presented. The method is based on the smoothing of the forces, leading to regularized Stokeslets. The resulting expressions provide the pressure and velocity field as functions of the forcing. The latter expression can also be inverted to find the forces that impose a given velocity boundary condition. The numerical examples presented demonstrate the wide applicability of the method and its properties. Solutions converge with second-order accuracy when forces are exerted along smooth boundaries. Examples of segmented boundaries and forcing at random points are also presented.

Key words. Stokes flow, immersed boundaries

AMS subject classifications. 76D07, 65M99, 65D32

PII. S106482750038146X

1. Introduction.

The numerical method presented here is for the computation of two- and three-dimensional Stokes flows driven by external forcing. The forces are applied on volumes or along boundaries, which may be curves, segments, or sets of disconnected points. In this way, the method applies to moving interfaces, elastic membranes interacting with a fluid, or flows through an array of fixed obstacles.

Stokes flows are of interest in many physical applications, particularly those in which the relevant length scales are extremely small or the fluid is extremely viscous. Many such applications emerge from biology, including the locomotion of bacteria and other cells, flagellated microorganisms, and flows in small capillaries. Other applications are the study of free surfaces, such as bubble motion, and flows around impurities or through a porous medium, which may be modeled as a collection of point obstacles for dilute cases.

The steady Stokes equations in two or three dimensions are

$$\mu\Delta\mathbf{u} = \nabla p - \mathbf{F}, \tag{1}$$

$$\nabla\cdot\mathbf{u} = 0, \tag{2}$$

where μ is the fluid viscosity, p is the pressure, $\mathbf{u}$ is the velocity, and $\mathbf{F}$ is force. A fundamental solution of these equations is called a *Stokeslet*, and it represents the velocity due to a concentrated external force acting on the fluid at a single point [32, 34, 1, 25]. Other, more singular solutions can also be derived from the Stokeslet by differentiation. Many important models have been created from the superposition of these fundamental solutions. Examples are analyses of flagellar motions [17, 16, 24]; the beating motion of cilia [4]; flows between plates, inside cylinders, or in periodic geometries [26, 27, 15, 33]; and slender body theories [6, 20, 23, 13]. Many of these analyses make use of images to enforce boundary conditions on the surface of spheres or planes.

[*] Received by the editors November 21, 2000; accepted for publication (in revised form) April 9, 2001; published electronically November 7, 2001. This work was supported in part by NSF grant DMS-9816951.

http://www.siam.org/journals/sisc/23-4/38146.html

[†] Department of Mathematics, Tulane University, 6823 St. Charles Ave., New Orleans, LA 70118 (cortez@math.tulane.edu).

The singularity in the Stokeslet is proportional to $1/r$ in three dimensions and $\log(r)$ in two dimensions. Consequently, when forces are concentrated on surfaces embedded in $\mathbb{R}^3$ (curves in $\mathbb{R}^2$), the expression for velocity is integrable and the flow is bounded in the vicinity of the surfaces. Most of the works cited above rely on this fact and the smoothness of the surfaces to get final expressions for the fluid velocity.

When the forces are concentrated along curves in $\mathbb{R}^3$ (points in $\mathbb{R}^2$) or when surfaces are not smooth, the situation is more difficult because the velocity formula is singular. One technique for dealing with this problem is to desingularize the expressions by introducing a small cutoff parameter in the kernels in order to regularize them. This approach has been extremely useful for the modeling of vortex motion [7, 3, 21, 14], interface motion in inviscid fluids [5, 35, 10, 8, 36], and other processes [11, 9, 12]. In all of these applications, the errors involved have been analyzed extensively and are well understood.

Our approach here is to consider the forces to be applied over a small ball, where they vary smoothly from a maximum value at the center to zero on its surface, rather than being concentrated at points (as Dirac measures). The radius of the support of the forces is a numerical parameter that can be controlled independently from any boundary discretization. Sometimes the radius of the ball is infinite, but the forces decay fast away from the center. From this starting point, expressions for the pressure and velocity due to this regularized force are derived. These expressions are bounded in any bounded set and differ only from the standard Stokeslet near the points where the forces are exerted. The resulting method is applicable to any situation in which forces drive the motion, whether they are concentrated along interfaces or points. This gives the method wide applicability. The derivation of the expressions is presented in section 2 for radially symmetric regularizations and a specific example is presented in detail. Section 3 contains several numerical examples with particular focus on the performance of the method. The case of a smooth closed boundary in $\mathbb{R}^2$ is further developed in section 4, where the velocity expressions can be written in terms of single and double layer potentials. Recent work by Beale and Lai [2] is used to achieve second-order accuracy everywhere. The final section contains concluding remarks and future directions.

2. Equations.

We first consider the generic situation in which the forces are spread over a small ball centered at the points $\mathbf{x}_0$. The force is given by

$$\mathbf{F}(\mathbf{x}) = \mathbf{f}_0\ \phi_\epsilon(\mathbf{x} - \mathbf{x}_0), \tag{3}$$

where ϕ_ϵ is a radially symmetric smooth function with the property that $\int \phi_\epsilon(\mathbf{x})d\mathbf{x} = 1$. Examples of these functions, called *blobs* or *cutoffs*, are

$$\text{in } \mathbb{R}^2: \qquad \phi_\epsilon(\mathbf{x}) = \frac{1}{\pi\epsilon^2}e^{-|\mathbf{x}|^2/\epsilon^2} \quad \text{and} \quad \phi_\epsilon(\mathbf{x}) = \frac{3\epsilon^3}{2\pi(|\mathbf{x}|^2+\epsilon^2)^{5/2}},$$

$$\text{in } \mathbb{R}^3: \qquad \phi_\epsilon(\mathbf{x}) = \frac{3}{4\pi\epsilon^3}e^{-|\mathbf{x}|^3/\epsilon^3} \quad \text{and} \quad \phi_\epsilon(\mathbf{x}) = \frac{15\epsilon^4}{8\pi(r^2+\epsilon^2)^{7/2}}.$$

A typical blob is displayed in Figure 1. The graph shows the same blob with two different values of the parameter ϵ, which controls the width, or spreading. A tighter function (smaller ϵ) must necessarily be taller for the total integral to be 1. In the limit $\epsilon \to 0$, the blob approaches a Dirac delta.

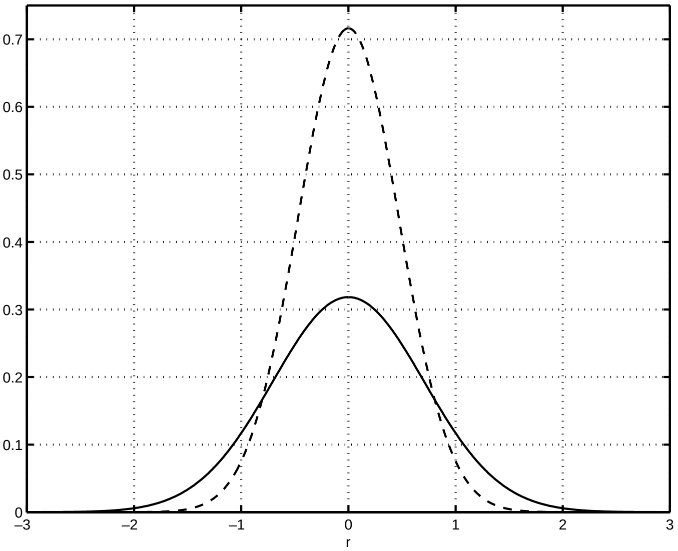


FIG. 1. *Typical blob for two different values of ϵ.*

To derive the solution of the problem in (1)–(2) in the case when the force is given by (3), we will use the following definitions. Let

$$G_\epsilon(\mathbf{x}) \text{ be the solution of } \Delta G_\epsilon = \phi_\epsilon(\mathbf{x}) \text{ in infinite space,}$$
$$B_\epsilon(\mathbf{x}) \text{ be the solution of } \Delta B_\epsilon = G_\epsilon(\mathbf{x}) \text{ in infinite space.}$$

The function $G_\epsilon(\mathbf{x})$ is a smooth approximation of the Green's function

$$\text{in } \mathbb{R}^2\text{: } G(\mathbf{x}) = \frac{1}{2\pi}\ln(|\mathbf{x}|), \qquad \text{in } \mathbb{R}^3\text{: } G(\mathbf{x}) = \frac{-1}{4\pi|\mathbf{x}|} \tag{4}$$

for $|\mathbf{x}| > \epsilon$. However, for small values of $|\mathbf{x}|$, the function G_ϵ is bounded. Similarly, $B_\epsilon(\mathbf{x})$ is smooth and approximates

$$\text{in } \mathbb{R}^2\text{: } B(\mathbf{x}) = \frac{|\mathbf{x}|^2}{8\pi}[\log(|\mathbf{x}|) - 1], \qquad \text{in } \mathbb{R}^3\text{: } B(\mathbf{x}) = \frac{-|\mathbf{x}|}{8\pi}, \tag{5}$$

which is the solution of the biharmonic equation $\Delta^2 B(\mathbf{x}) = \delta(\mathbf{x})$.

Taking the divergence of (1) and using (2) we find that the pressure satisfies

$$\Delta p = \nabla \cdot \mathbf{F},$$

which gives the particular solution

$$p = \mathbf{f}_0 \cdot \nabla G_\epsilon.$$

Now we use this expression to rewrite the equation for $\mathbf{u}$ as $\mu\Delta\mathbf{u} = (\mathbf{f}_0 \cdot \nabla)\nabla G_\epsilon - \mathbf{f}_0\, \phi_\epsilon$, whose particular solution is

$$\mu\mathbf{u}(\mathbf{x}) = (\mathbf{f}_0 \cdot \nabla)\nabla B_\epsilon(\mathbf{x} - \mathbf{x}_0) - \mathbf{f}_0 G_\epsilon(\mathbf{x} - \mathbf{x}_0).$$

This might be referred to as a *regularized Stokeslet* velocity.

If there are N forces, $\mathbf{f}_k$, centered at points $\mathbf{x}_k$, the pressure and velocity are given by

$$p(\mathbf{x}) = \sum_{k=1}^{N} \mathbf{f}_k \cdot \nabla G_\epsilon(\mathbf{x}-\mathbf{x}_k), \tag{6}$$

$$\mathbf{u}(\mathbf{x}) = \mathbf{U}_o + \frac{1}{\mu}\sum_{k=1}^{N} \{(\mathbf{f}_k\cdot\nabla)\nabla B_\epsilon(\mathbf{x}-\mathbf{x}_k) - \mathbf{f}_k G_\epsilon(\mathbf{x}-\mathbf{x}_k)\}, \tag{7}$$

where we have added a constant flow $\mathbf{U}_o$ to be determined shortly. The functions G_ϵ and B_ϵ are completely determined by the blob used. They are radially symmetric just like their singular counterparts. Consequently, the velocity expression can be modified using the formulas (for $r = |\mathbf{x}|$)

$$\nabla B_\epsilon = B_\epsilon{}'(r)\frac{\mathbf{x}}{r} \quad \text{and} \quad (\mathbf{f}\cdot\nabla)\nabla B_\epsilon = \mathbf{f}\,\frac{B_\epsilon{}'(r)}{r} + (\mathbf{f}\cdot\mathbf{x})\mathbf{x}\left[\frac{rB_\epsilon{}''(r) - B_\epsilon{}'(r)}{r^3}\right],$$

which yield for $r_k = |\mathbf{x}-\mathbf{x}_k|$

$$p(\mathbf{x}) = \sum_{k=1}^{N} [\mathbf{f}_k\cdot(\mathbf{x}-\mathbf{x}_k)]\left[\frac{G_\epsilon{}'(r_k)}{r_k}\right], \tag{8}$$

$$\begin{aligned}\mathbf{u}(\mathbf{x}) = \mathbf{U}_o + \frac{1}{\mu}\sum_{k=1}^{N}\Bigg\{&\mathbf{f}_k\left[\frac{B_\epsilon{}'(r_k)}{r_k} - G_\epsilon(r_k)\right] \\ &+ [\mathbf{f}_k\cdot(\mathbf{x}-\mathbf{x}_k)](\mathbf{x}-\mathbf{x}_k)\left[\frac{r_kB_\epsilon{}''(r_k) - B_\epsilon{}'(r_k)}{r_k^3}\right]\Bigg\}.\end{aligned} \tag{9}$$

In two dimensions, one can use (4)–(5) to check that $B'(r)/r - G(r) = -G(r)/2 - 1/8\pi$, so one can choose $\mathbf{U}_o = \sum_k \mathbf{f}_k/8\pi\mu$ in order to eliminate this remaining constant flow. In three dimensions, one can simply let $\mathbf{U}_o = \mathbf{0}$.

The method of regularized Stokeslets uses (8)–(9) to find the velocity induced by given forces. One important property of this formulation is that the flow given by (7) satisfies the incompressibility constraint $\nabla\cdot\mathbf{u} = 0$ analytically everywhere. This important property is key in the conservation of fluid volumes bounded by elastic membranes.

2.1. Deriving G_ϵ and B_ϵ. Since ϕ_ϵ is radially symmetric and $\Delta G_\epsilon = \phi_\epsilon$, we have that

$$\Delta G_\epsilon = \frac{1}{r}[rG_\epsilon{}'(r)]' = \phi_\epsilon(r)$$

and

$$G_\epsilon{}'(r) = \frac{1}{r}\int_0^r s\ \phi_\epsilon(s)ds.$$

After integrating once more, we obtain $G_\epsilon(r)$. Similarly,

$$\frac{1}{r}[rB_\epsilon{}'(r)]' = G_\epsilon(r)$$

so that $B_\epsilon(r)$ can be found in the same way.

2.1.1. A specific choice of blob in two dimensions. We present a concrete example of the regularized Stokeslet method by deriving the two-dimensional pressure and velocity formulas for the specific blob

$$\phi_\epsilon(\mathbf{x}) = \frac{3\epsilon^3}{2\pi(|\mathbf{x}|^2+\epsilon^2)^{5/2}}.$$

The corresponding regularized Green's function is

$$G_\epsilon(r) = \frac{1}{2\pi}\left[\ln\left(\sqrt{r^2+\epsilon^2}+\epsilon\right) - \frac{\epsilon}{\sqrt{r^2+\epsilon^2}}\right]$$

and

$$B_\epsilon{}'(r) = \frac{1}{8\pi}\left[2r\ln\left(\sqrt{r^2+\epsilon^2}+\epsilon\right) - r - \frac{2r\epsilon}{\sqrt{r^2+\epsilon^2}+\epsilon}\right].$$

With these functions we can write the final expressions to be used in the numerical method

$$p(\mathbf{x}) = \sum_{k=1}^{N}\frac{1}{2\pi}[\mathbf{f}_k\cdot(\mathbf{x}-\mathbf{x}_k)]\left[\frac{r_k^2+2\epsilon^2+\epsilon\sqrt{r_k^2+\epsilon^2}}{\left(\sqrt{r_k^2+\epsilon^2}+\epsilon\right)(r_k^2+\epsilon^2)^{3/2}}\right], \tag{10}$$

$$\begin{aligned}\mathbf{u}(\mathbf{x}) = \sum_{k=1}^{N}\frac{-\mathbf{f}_k}{4\pi\mu}&\left[\ln\left(\sqrt{r_k^2+\epsilon^2}+\epsilon\right) - \frac{\epsilon\left(\sqrt{r_k^2+\epsilon^2}+2\epsilon\right)}{\left(\sqrt{r_k^2+\epsilon^2}+\epsilon\right)\sqrt{r_k^2+\epsilon^2}}\right]\\ &+\frac{1}{4\pi\mu}[\mathbf{f}_k\cdot(\mathbf{x}-\mathbf{x}_k)](\mathbf{x}-\mathbf{x}_k)\left[\frac{\sqrt{r_k^2+\epsilon^2}+2\epsilon}{\left(\sqrt{r_k^2+\epsilon^2}+\epsilon\right)^2\sqrt{r_k^2+\epsilon^2}}\right],\end{aligned} \tag{11}$$

where $r_k = |\mathbf{x}-\mathbf{x}_k|$.

One can verify that these formulas are consistent with the standard Stokeslet expressions for the case of a Dirac delta distribution of forces. Taking the limit $\epsilon \to 0$ in (10)–(11), we obtain the well-known formulas (in $\mathbb{R}^2$)

$$p(\mathbf{x}) = \sum_{k=1}^{N}\frac{[\mathbf{f}_k\cdot(\mathbf{x}-\mathbf{x}_k)]}{2\pi r_k^2}, \tag{12}$$

$$\mathbf{u}(\mathbf{x}) = \sum_{k=1}^{N}\frac{-\mathbf{f}_k}{4\pi\mu}\ln(r_k) + [\mathbf{f}_k\cdot(\mathbf{x}-\mathbf{x}_k)]\,\frac{(\mathbf{x}-\mathbf{x}_k)}{4\pi\mu r_k^2}. \tag{13}$$

Notice that (13) is singular at the point where the force is centered. When the sum is replaced by an integral, the velocity expression is generally not problematic when the forces are defined along smooth closed curves, since the kernel is integrable. The same is not true for the pressure which typically displays discontinuities across boundaries. A much different scenario is when the forces are defined at disconnected points, not closed curves. In this case, (12)–(13) cannot be interpreted as discretizations of integrals but simply as the superposition of singular solutions. The singularities in these formulas make the Stokeslet expression somewhat impractical for computation, since

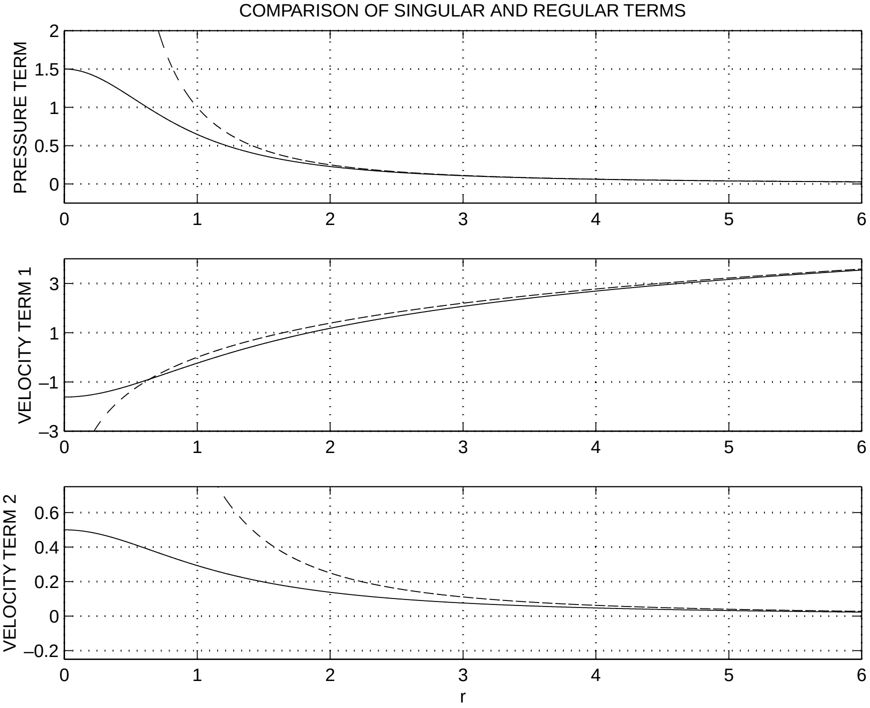


Fig. 2. *Comparison of the radial factors (in brackets) in* (10)–(11) *shown by solid lines. The regularization parameter was set to* $\epsilon = 1$. *The dashed lines are the corresponding singular terms in the Stokeslet formulas* (12)–(13).

the evaluation of the expression at a point very close to an $\mathbf{x}_k$ would lead to extremely large velocities. The regularization in (10)–(11) provides the necessary stability.

A comparison of the terms in brackets in (10)–(11) for $\epsilon = 1$ is shown in Figure 2. The smooth functions eliminate the singularity but approach asymptotically the Stokeslet expressions for $r \gg \epsilon$. In practice, the value of ϵ would be of the same order of magnitude as the particle separation in the spatial discretizations.

2.2. Finding the forces from the velocities. The numerical method derived so far can be used directly to compute the flow due to given forces. For example, elastic membranes interacting with a fluid exert forces given in terms of their configuration, which can be computed from the current position of the membrane. There are situations when the velocity of a given body is known and one is interested in finding the forces that yield that flow. Equation (9) can be used to impose velocity boundary conditions on the surface of a body. One can write, for $i = 1, \ldots, N$, the system of equations

$$\mathbf{u}(\mathbf{x}_i) = \sum_{j=1}^{N} M_{ij}(\mathbf{x}_1, \ldots, \mathbf{x}_N)\ \mathbf{f}_j$$

or, as a matrix equation,

$$\mathcal{U} = \mathcal{M}\mathcal{F}, \tag{14}$$

where, in two dimensions, $\mathcal{U}$ and $\mathcal{F}$ are $2N \times 1$ vectors and $\mathcal{M}$ is $2N \times 2N$. Generally, the matrix $\mathcal{M}$ is not invertible. This is most easily seen for the case of a closed membrane surrounding the fluid. An arbitrary constant can be added to the normal component of the forces, causing a change in the pressure but not in the velocities because of the incompressibility of the fluid. Fortunately, any solution of the system is acceptable in the present context so that one can add the constraint that the normal component of the forces sum to zero or simply find any solution of the system via an iterative process. In this paper, we use GMRES with zero initial guess.

In steady flow problems or in situations where the geometry of the body is fixed, the matrix and any required factorization need be computed only once. In problems of moving interfaces, the matrix must be rebuilt at every time step.

3. Numerical examples.

3.1. Example 1: Flow past a cylinder. As a simple numerical experiment, consider the classical example of a circular cylinder of radius a moving at a constant speed $(1,0)$. The exact solution is given by a combination of a Stokeslet and a dipole at the origin (see [1, p. 245]):

$$\mathbf{u}(\mathbf{x}) = \frac{-\mathbf{f}_o}{8\pi}\left(2\ln|\mathbf{x}| - a^2/|\mathbf{x}|^2\right) + \frac{(\mathbf{f}_o\cdot\mathbf{x})\mathbf{x}}{4\pi|\mathbf{x}|^2}\left(1 - a^2/|\mathbf{x}|^2\right)$$

with

$$\mathbf{f}_o = \frac{8\pi}{1-2\ln(a)}\begin{pmatrix}1\\0\end{pmatrix}.$$

Note that the velocity is unbounded as $|\mathbf{x}| \to \infty$, consistent with two-dimensional flow due to a nonzero net force (Stokes's paradox). However, the solution is valid in a region containing the cylinder.

To compute the solution with the numerical method, we discretize the circle with N particles, set the velocity at each point to $(1,0)$, and find a vector of forces that solves the system in (14). Once the forces are known, we use (11) to find the velocity on a grid, where the exact solution is also computed for comparison. The parameters used in this example are $a = 0.25$, $N = 160$ (with $\Delta s = 2\pi a/N$), and $\epsilon = 0.25\Delta s$. Figure 3 shows the contours of the two components of velocity. Each plot shows the analytic solution (solid) and the computed solution (dashed). The velocities at the boundary points are computed with the accuracy of the iteration solver for the forces; in this case, the velocities are within 10^{-15} of their intended value.

The solutions satisfy $\|\mathbf{u}-\mathbf{u}_{exact}\|_\infty \leq 2.6\times10^{-3}$ for the parameters used, although the error in the velocities is largest near the cylinder and decays nearly as $\sim |\mathbf{x}|^{-2}$ away from the boundary. In section 4, we explain how to improve on this solution. The regularization parameter ϵ is taken to be proportional to the discretization size Δs, where the proportionality constant depends on the blob ϕ_ϵ and on the particular application. In our experience, this constant is typically less than 1.

3.2. Example 2: Flow past fixed obstacles. As a second example, we consider the problem of a uniform background flow around fixed line obstacles. In this problem, the boundaries that exert forces on the fluid are not closed curves; they are disconnected line segments as shown in Figure 4. The background flow is $(1,0)$ and two fixed line obstacles of different lengths are placed at different angles relative to the flow. The boundaries were discretized with 45 particles $\mathbf{x}_k$ (26 on the top segment and 19 on the bottom one) separated by a distance $\Delta s = 0.00447$, and the regularization parameter was set to $\epsilon = \Delta s/2$, although the solution is not very sensitive to this choice.

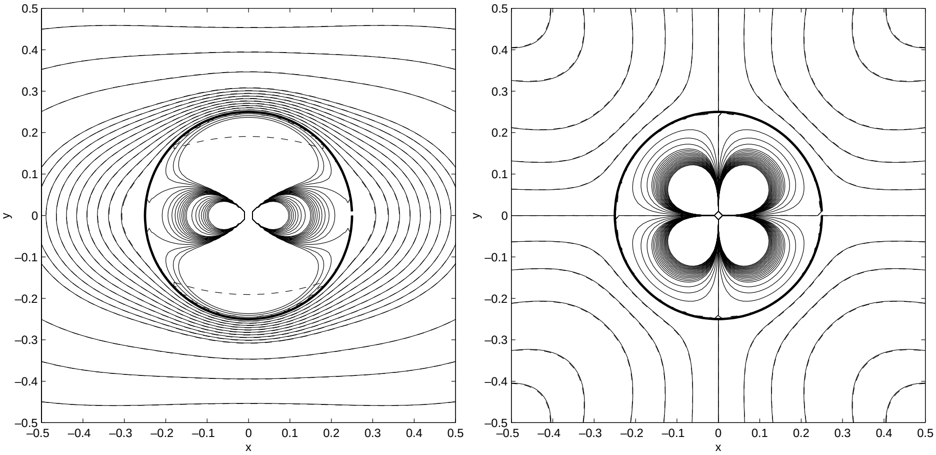


FIG. 3. *Velocity contours of the flow due to a circular cylinder of radius* $a = 0.25$ *moving with velocity* $(1, 0)$*. The left graph compares* u *in* (11) *(dashed) and the exact solution (solid). The right graph compares* v *in the same way.*

In this case, the forces on the boundaries must cancel the background flow at the $\mathbf{x}_k$'s, so the left-hand side of (14) was set to velocities corresponding to $(-1, 0)$. Once the forces are found, the flow at any point $\mathbf{x}$ is the flow induced by the forces plus the background flow. The velocity at the boundaries was no larger than 10^{-14} in magnitude. Using this steady flow computed on a regular grid, a cloud of "smoke" was advected through the obstacles for visualization purposes. This was done by solving an advection equation with the precomputed steady flow using an upwind method. The grid covered the region $[0, 1] \times [0, 0.5]$ and Neumann boundary conditions were used at the edge of the computational domain. Three snapshots are shown in Figure 4, where the fixed segments obstruct the flow and force it to go around and between them.

The same method can be used to compute the flow due to a collection of forces located at random points or the flow around a collection of point obstacles which may represent, for example, a porous medium. As an example, we consider a group of fixed particles placed randomly in a domain blocking a uniform background flow $(1, 0)$. Figure 5 shows a smaller subdomain containing 53 of the point obstacles and the flow around them. The forces at the particles are computed so that they remain stationary. The computational parameter ϵ was set to 0.03 so as to provide a resolution length scale.

3.3. Example 3: Quasi-steady flow. In this example, we consider the unsteady motion of an interface initially given in polar coordinates by $r(\theta) = \sqrt{r_o^2 - a^2/2} + a\cos(2\theta)$, where a is a perturbation amplitude. The area enclosed by this interface equals πr_o^2. The parameters used were $r_o = 0.25$ and $a = 0.05$ (see Figure 6). Since we are interested in the interface motion as it relaxes from its initial shape to a circle, we need to compute the velocity only along the interface using (11). Although the steady Stokes equations are used to compute the velocity, the time-dependent forces impose a time scale. The interface was tracked by computing the velocity as a steady flow at every time step. A similar problem was discussed in [38, 22].

Suppose $\mathbf{x}(s, t)$ defines the interface at time t in terms of the arclength parameter

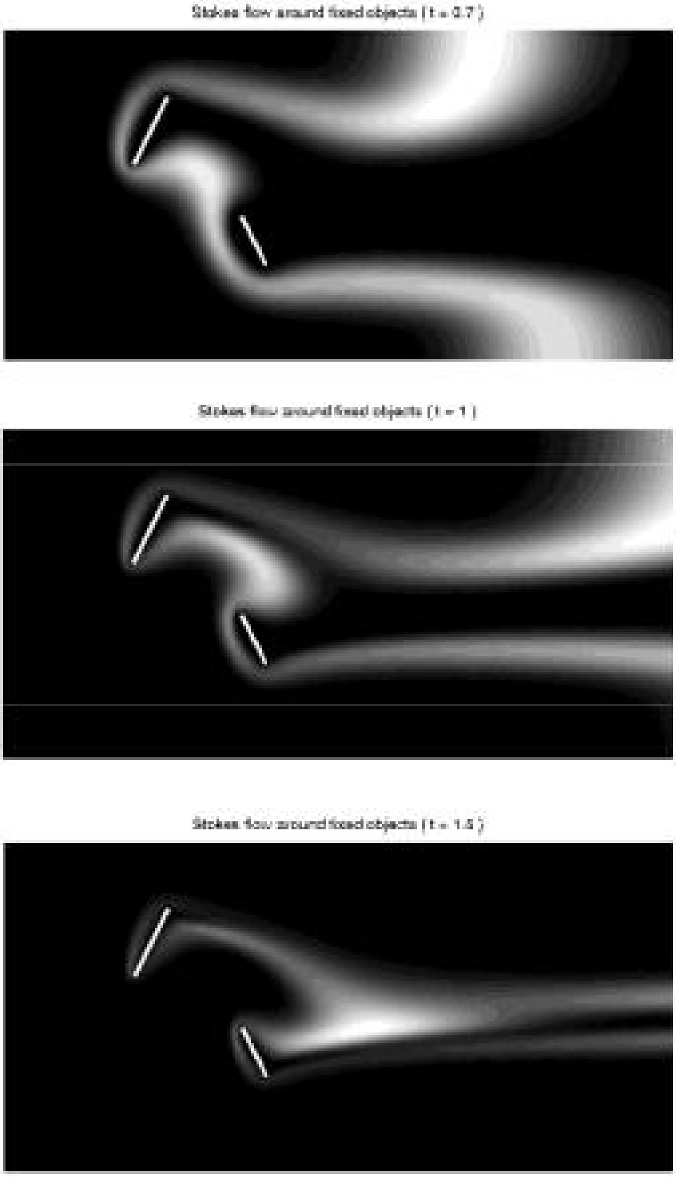


FIG. 4. *Stokes flow around two fixed line obstacles at three different times. The white cloud (initially a wide vertical stripe at the inflow) represents a quantity advected by the precomputed flow for visualization.*

s. A force

$$\mathbf{f}(s) = \frac{\partial^2 \mathbf{x}(s)}{\partial s^2} + \frac{1}{r_o}\hat{n}(s)$$

representing the curvature of the interface minus that of the final state is applied so that the shape will converge to a circle of radius r_o. This restoring force vanishes as the equilibrium state is reached. Following the suggestion in [22], one way to verify

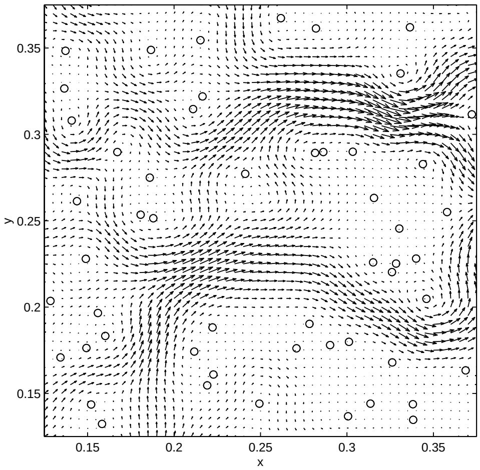


FIG. 5. *Flow around* 53 *fixed point obstacles.*

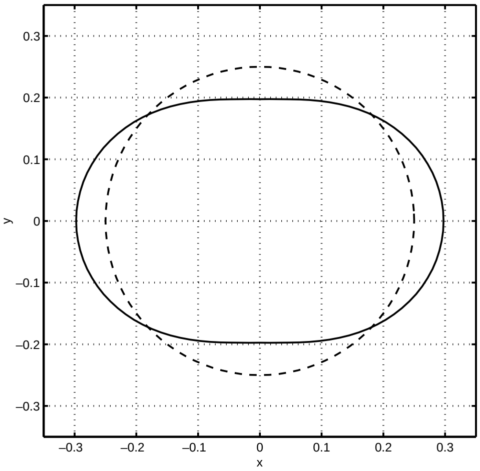


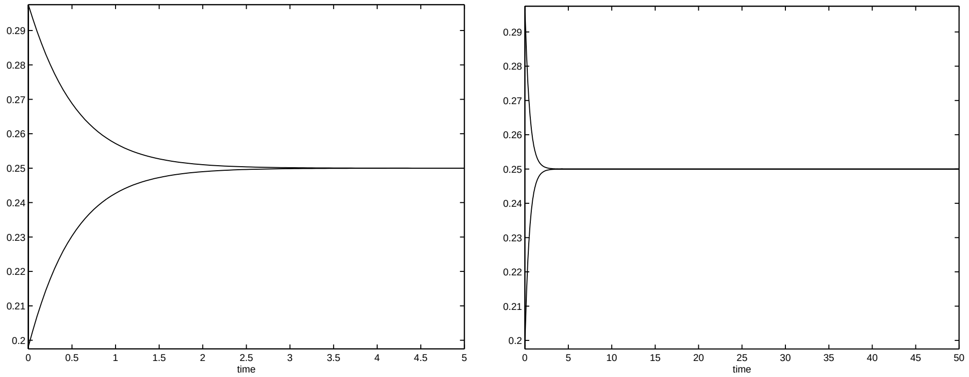
FIG. 6. *Initial shape (solid) and circle of radius* $r_o = 0.25$ *enclosing the same area.*

the shape of the final solution is to compute

$$r_{min}(t) = \min_{1 \le k \le N} \|\mathbf{x}_k(t)\| \quad \text{and} \quad r_{max}(t) = \max_{1 \le k \le N} \|\mathbf{x}_k(t)\|$$

and to check their convergence as $t \to \infty$. Figure 7 shows these results for a discretization corresponding to $N = 50$ points along the interface (equally spaced at $t = 0$). Figure 7(a) is for early times and Figure 7(b) shows the longtime behavior. The figure shows that both $r_{max}(t)$ and $r_{min}(t)$ converge to r_o. In addition, the membrane retains its circular shape indefinitely, indicating that the computation remains stable for very long times.

The interface will stop moving once the discretization errors dominate the force giving and indication of the accuracy of the method. Table 1 summarizes the results for discretizations corresponding to $N = 50$, 100, and 200. The second column of the table shows the discretization size for each run. In every case, the regularization parameter was set to $\epsilon = 1.2\Delta s$. We found this to give good results, although the



(a) (b)

FIG. 7. *Plot of $r_{max}(t)$ and $r_{min}(t)$ showing the distance from the origin to the interface as a function of time for $N = 50$ and $\epsilon = 1.2\Delta s$.* (a) *shows the plot for $0 \leq t \leq 5$ and* (b) *shows the plot for $0 \leq t \leq 50$.*

TABLE 1
Results showing r_{min}, r_{max}, area conservation A, and the magnitude of the forces for long times and various discretizations.

N	Δs	$r_{max} - r_{min}$	$\lvert A(t = 50) - \pi r_o^2\rvert$	$\lVert\mathbf{f}(t \geq 10)\rVert_\infty$
50	0.0323193	1.156836×10^{-6}	1.758022×10^{-6}	5.700275×10^{-5}
100	0.0161693	0.048211×10^{-6}	0.078004×10^{-6}	0.323780×10^{-5}
200	0.0080858	0.012735×10^{-6}	0.016602×10^{-6}	0.001307×10^{-5}

differences from other choices were not significant. The third column shows the convergence of the shape to a circle at $t = 50$; however, these numbers are essentially constant for $t \geq 10$. Note that the numbers indicate quadratic convergence in terms of Δs. The fourth column of Table 1 shows the error in the computed area inside the interface relative to the expected value πr_o^2. These numbers remained nearly constant throughout the runs. The areas were computed by discretizing a boundary integral using fourth-order finite differences. The errors in the table are within this discretization error. Finally, the last column of the table shows the maximum magnitude of the forces for $t \geq 10$ as a function of the discretization. They give an indication of the errors involved in the numerical method.

4. Modifications for closed, smooth boundaries. The case of a closed, smooth boundary is the most widely studied and one for which several numerical methods exist. This is because the situation is simplified by the absence of endpoints or by the presence of a boundary consisting of disconnected points. Techniques discussed in [34, 22, 28, 29, 30, 31, 37] are often used to compute accurate solutions of elliptic problems in general regions. In the context of fluid flow, the computation of the velocities and pressure far from the boundary or on the boundary are not typically problematic. The challenge is the computation of the solution at points very close to but off the boundary. This situation arises, for example, when the velocity is required in the entire computational domain for the advection of chemicals or solvents.

We present a modified regularization approach that leads to pointwise second-order accuracy in ϵ.

We consider a closed boundary parametrized by arclength $\mathbf{x}(s)$ for $0 \leq s \leq L$.

Our strategy will be to write the solution as a combination of integrals of the form

$$I_1(\mathbf{x}, f) = \int_0^L f(s)\ \mathbf{x}'(s) \times \nabla G(\mathbf{x} - \mathbf{x}(s))\ ds$$

and

$$I_2(\mathbf{x}, g) = \int_0^L g(s)\ \mathbf{x}'(s) \cdot \nabla G(\mathbf{x} - \mathbf{x}(s))\ ds$$

for some functions $f(s)$ and $g(s)$ and Green's function G. The reason is that Beale and Lai [2] have developed correction terms that can be added to the trapezoid rule applied to these integrals that guarantee second-order accuracy. To approximate these integrals, first we use the identities

$$\int_0^L \mathbf{x}'(s) \cdot \nabla G(\mathbf{x} - \mathbf{x}(s))\ ds = 0 \quad \text{for all } \mathbf{x} \text{ off the curve,}$$

$$\int_0^L \mathbf{x}'(s) \times \nabla G(\mathbf{x} - \mathbf{x}(s))\ ds = \begin{cases} 1 & \text{for } \mathbf{x} \text{ inside the boundary,} \\ 0 & \text{for } \mathbf{x} \text{ outside the boundary} \end{cases}$$

to rewrite them, for any value of s^*, as

$$I_2(\mathbf{x}, g) = \int_0^L [\mathbf{x}'(s) \cdot \nabla G(\mathbf{x} - \mathbf{x}(s))]\,[g(s) - g(s^*)]\ ds$$

and

$$I_1(\mathbf{x}, f) = \int_0^L [\mathbf{x}'(s) \times \nabla G(\mathbf{x} - \mathbf{x}(s))]\,[f(s) - f(s^*)]\ ds + \chi(\mathbf{x}) f(s^*),$$

where $\chi(\mathbf{x}) = 1$ if $\mathbf{x}$ is enclosed by the boundary and $\chi(\mathbf{x}) = 0$ otherwise. This formulation gives additional regularity to the integrands. The idea is to discretize the integrals with the trapezoid rule and to realize that when the evaluation point $\mathbf{x}$ is very near the boundary, i.e., $\|\mathbf{x} - \mathbf{x}(s)\| \le O(\Delta s)$ for some s, the accuracy of the quadrature is compromised. However, the leading-order error terms can be identified and subtracted from the quadrature in order to achieve second-order accuracy pointwise. Therefore,

$$\begin{aligned} I_1(\mathbf{x}, f) = \sum_{k=0}^{N} \mathbf{x}'(s_k) \times \nabla G_\epsilon(\mathbf{x} - \mathbf{x}(s_k))[f_k - f(s^*)]\Delta s \\ + \chi(\mathbf{x}) f(s^*) + T_1^{(n)}(\mathbf{x}) + T_2^{(n)}(\mathbf{x}) + O(\Delta s^2), \end{aligned}$$

where the $T^{(n)}$ terms are corrections associated with the use of G_ϵ in place of G and the discretization of the integral. Similarly,

$$I_2(\mathbf{x}, g) = \sum_{k=0}^{N} \mathbf{x}'(s_k) \cdot \nabla G_\epsilon(\mathbf{x} - \mathbf{x}(s_k))[g_k - g(s^*)]\Delta s + T_1^{(t)}(\mathbf{x}) + T_2^{(t)}(\mathbf{x}) + O(\Delta s^2).$$

For each point $\mathbf{x}$ where the sum is evaluated we require s^*, which is the value of the curve parameter that identifies the boundary point closest to $\mathbf{x}$. These formulas have been derived for the specific cutoff function

$$\phi_\epsilon(r) = \frac{1}{\pi \epsilon^2} e^{-r^2/\epsilon^2}.$$

Refer to [2] for more details.

In order to write formulas for p and $\mathbf{u}$ in terms of this type of integrals, it will be convenient to use the decomposition of the forces

$$\mathbf{f}(s) = f^{(n)}\hat{n}(s) + f^{(t)}\mathbf{x}'(s),$$

where the outward normal vector is $\hat{n}(s) = (y'(s), -x'(s))$. Based on (12) we have that

$$\begin{aligned} p(\mathbf{x}) &= \int_0^L \frac{\mathbf{f}(s)\cdot(\mathbf{x}-\mathbf{x}(s))}{2\pi|\mathbf{x}-\mathbf{x}(s)|^2}\,ds = \int_0^L \mathbf{f}(s)\cdot\nabla G(\mathbf{x}-\mathbf{x}(s))\,ds \\ &= \int_0^L -f^{(n)}(s)\ \mathbf{x}'(s)\times\nabla G(\mathbf{x}-\mathbf{x}(s))\,ds + \int_0^L f^{(t)}(s)\ \mathbf{x}'(s)\cdot\nabla G(\mathbf{x}-\mathbf{x}(s))\,ds \\ &= -I_1(\mathbf{x}, f^{(n)}) + I_2(\mathbf{x}, f^{(t)}). \end{aligned} \tag{15}$$

From (13) we have

$$\begin{aligned} \mathbf{u}(\mathbf{x}) &= \int_0^L \frac{-\mathbf{f}(s)}{4\pi\mu}\ \ln|\mathbf{x}-\mathbf{x}(s)| + [\mathbf{f}(s)\cdot(\mathbf{x}-\mathbf{x}(s))]\frac{(\mathbf{x}-\mathbf{x}(s))}{4\pi\mu|\mathbf{x}-\mathbf{x}(s)|^2}\,ds \\ &= \int_0^L \left(\frac{-\mathbf{f}(s)}{2\mu}\right) G(\mathbf{x}-\mathbf{x}(s))\,ds + \int_0^L (\mathbf{x}-\mathbf{x}(s))\frac{[\mathbf{f}(s)\cdot\nabla G(\mathbf{x}-\mathbf{x}(s))]}{2\mu}\,ds. \end{aligned}$$

By defining the vector

$$\mathbf{F}(s) = \int_0^s \frac{-\mathbf{f}(\alpha)}{2\mu}\,d\alpha,$$

we can write the first integral as

$$\int_0^L \mathbf{F}(s)\ \mathbf{x}'(s)\cdot\nabla G(\mathbf{x}-\mathbf{x}(s))\,ds = I_2(\mathbf{x}, \mathbf{F}). \tag{16}$$

The second integral becomes

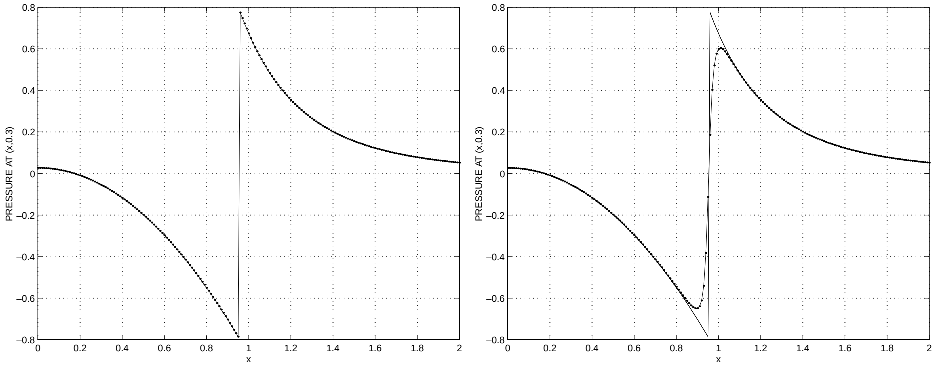
$$\begin{aligned} &\int_0^L (\mathbf{x}-\mathbf{x}(s))\frac{[\mathbf{f}(s)\cdot\nabla G(\mathbf{x}-\mathbf{x}(s))]}{2\mu}\,ds \\ &= \int_0^L \frac{(\mathbf{x}-\mathbf{x}(s))}{2\mu}[-f^{(n)}(s)\mathbf{x}'(s)\times\nabla G(\mathbf{x}-\mathbf{x}(s)) + f^{(t)}(s)\mathbf{x}'(s)\cdot\nabla G(\mathbf{x}-\mathbf{x}(s))]\,ds \\ &= I_1(\mathbf{x}, \mathbf{Q^a}) + I_2(\mathbf{x}, \mathbf{Q^b}), \end{aligned}$$

where

$$\mathbf{Q^a}(s) = -\frac{(\mathbf{x}-\mathbf{x}(s))f^{(n)}(s)}{2\mu}, \quad \mathbf{Q^b}(s) = \frac{(\mathbf{x}-\mathbf{x}(s))f^{(t)}(s)}{2\mu}.$$

Then,

$$\mathbf{u}(\mathbf{x}) = I_2(\mathbf{x}, \mathbf{F}) + I_1(\mathbf{x}, \mathbf{Q^a}) + I_2(\mathbf{x}, \mathbf{Q^b}). \tag{17}$$



(a) (b)

FIG. 8. (a) *Computed pressure p along the line $(x, 3/10)$ for $N = 100$ (dots) using the method of section* 4. *Also plotted is the exact (discontinuous) solution.* (b) *Pressure computed using* (10).

4.1. Example 4a: Normal forces on a circle. Consider a circular boundary of radius 1 parametrized by $\mathbf{x}(\theta) = (\cos(\theta), \sin(\theta))$ and define the forces on the boundary by

$$\mathbf{f}(\theta) = 2\sin(3\theta)\mathbf{x}(\theta)$$

so that the forces are *normal* to the boundary. The exact solution is given by

$$p(r,\theta) = \begin{cases} -r^3\sin(3\theta), & r < 1, \\ r^{-3}\ \sin(3\theta), & r > 1, \end{cases}$$

$$u(r,\theta) = \begin{cases} \frac{3}{8}r^2\sin(2\theta) + \frac{1}{16}r^4\sin(4\theta) - \frac{1}{4}r^4\sin(2\theta), & r < 1, \\ \\ \frac{1}{8}r^{-2}\sin(2\theta) - \frac{3}{16}r^{-4}\sin(4\theta) + \frac{1}{4}r^{-2}\sin(4\theta), & r \geq 1, \end{cases}$$

$$v(r,\theta) = \begin{cases} \frac{3}{8}r^2\cos(2\theta) - \frac{1}{16}r^4\cos(4\theta) - \frac{1}{4}r^4\cos(2\theta), & r < 1, \\ \\ \frac{1}{8}r^{-2}\cos(2\theta) + \frac{3}{16}r^{-4}\cos(4\theta) - \frac{1}{4}r^{-2}\cos(4\theta), & r \geq 1. \end{cases}$$

We note that p is discontinuous across the boundary, but $\partial p/\partial r$, u, ∇u, v, and ∇v are continuous.

In order to show representative results, we choose to compare the solution along the line $(x, 3/10)$ for $0 < x < 2$. The boundary was discretized with N equally spaced points, and the regularization parameter was set equal to $\epsilon = 2\pi/16N$. Figures 8 and 9 show results for $N = 100$.

Figure 8(a) displays the computed pressure using the method in section 4. For larger values of N, the solutions are nearly indistinguishable from one another. The formula based on (15) is able to capture the jump in the pressure in spite of the use of regularization. This is because the jump was explicitly included in the formula for $I_1(\mathbf{x}, f)$. If instead one used (10) to compute the pressure, the effect of the regularization would be to replace the discontinuity with a rapid but smooth transition over a region of size $O(\epsilon)$, as shown in Figure 8(b). This, of course, leads to $O(1)$ errors near the interface.

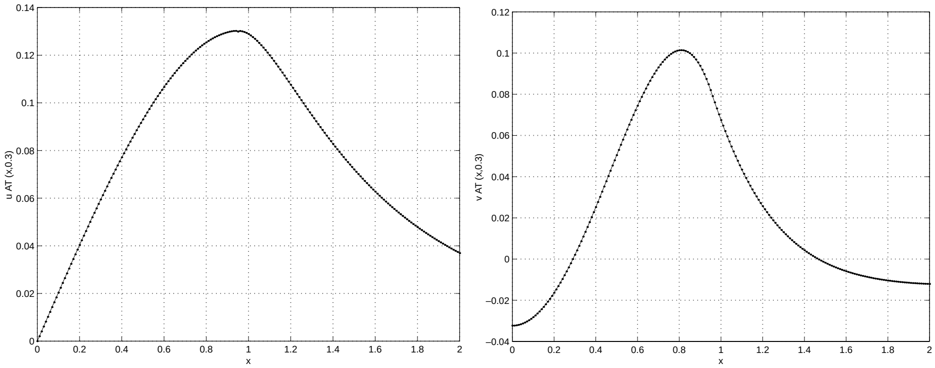


FIG. 9. *Velocity components u (left) and v (right) along the line* $(x, 3/10)$ *for* $N = 100$ *(dots). Also plotted is the exact solution (solid).*

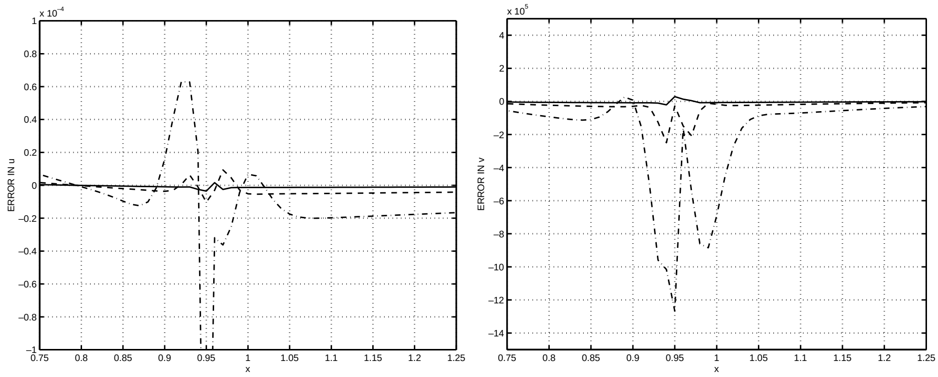


FIG. 10. *Error in the velocity components along the line* $(x, 3/10)$ *for* $N = 100$*,* 200*, and* 400.

The velocities are $C^1(\mathbb{R}^2)$ and this additional smoothness allows the numerical method to approximate the velocities very well everywhere, as seen in Figure 9. The largest discrepancy, however, is still near the boundary. To be more precise, we compute the error in the solution for the various discretizations. We recall that, in this example, ϵ is inversely proportional to N. Figure 10 shows the error in the velocities and Figure 11 the error in the pressure.

The largest velocity errors are $O(10^{-4})$, corresponding to the smallest value of N. More importantly, the L_2 norm of these errors, shown in Table 2, decreases by a factor of about 4 as the number of boundary points N is doubled, indicating that the pressure and velocity errors are $O(\epsilon^2)$ as $\epsilon \to 0$.

4.2. Example 4b: Tangential forces on a circle. Now consider the same circular boundary of radius 1 parametrized by $\mathbf{x}(\theta) = (\cos(\theta), \sin(\theta))$ but define the forces on the boundary by

$$\mathbf{f}(\theta) = 2\sin(3\theta)\mathbf{x}'(\theta)$$

so that the forces are now *tangential* to the boundary. The difference between this and the previous example is significant because of the character of the solution. Here, p and $\mathbf{u}$ are continuous everywhere, but their gradients are discontinuous across the

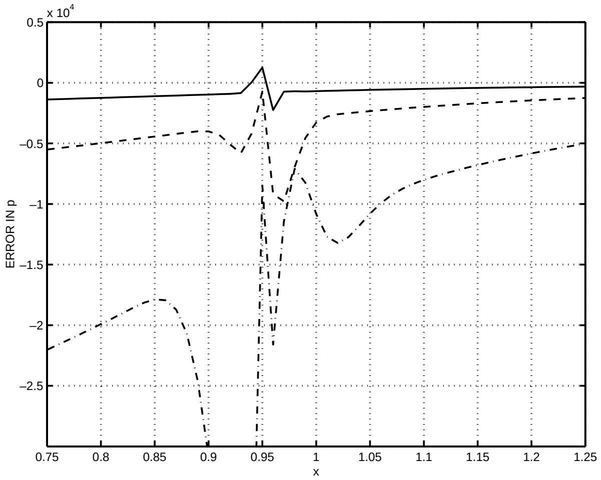


FIG. 11. *Error in the pressure along the line* $(x, 3/10)$ *for* $N = 100$, 200, *and* 400.

TABLE 2

L_2 *norm of errors shown in Figures* 10–11. *When the number of particles is doubled, the errors decrease by a factor of about* 4, *showing second order convergence.*

N	Error in p	Error in u	Error in v
100	25.35×10^{-4}	4.603×10^{-4}	4.656×10^{-4}
200	6.023×10^{-4}	0.781×10^{-4}	1.052×10^{-4}
400	1.468×10^{-4}	0.200×10^{-4}	0.248×10^{-4}

boundary. The exact solution is

$$p(r,\theta) = \begin{cases} -r^3\cos(3\theta), & r < 1, \\ -r^{-3}\ \cos(3\theta), & r > 1, \end{cases}$$

$$u(r,\theta) = \begin{cases} \frac{1}{8}r^2\cos(2\theta) + \frac{1}{16}r^4\cos(4\theta) - \frac{1}{4}r^4\cos(2\theta), & r < 1, \\ -\frac{1}{8}r^{-2}\cos(2\theta) + \frac{5}{16}r^{-4}\cos(4\theta) - \frac{1}{4}r^{-2}\cos(4\theta), & r \geq 1, \end{cases}$$

$$v(r,\theta) = \begin{cases} -\frac{1}{8}r^2\sin(2\theta) + \frac{1}{16}r^4\sin(4\theta) + \frac{1}{4}r^4\sin(2\theta), & r < 1, \\ \frac{1}{8}r^{-2}\sin(2\theta) + \frac{5}{16}r^{-4}\sin(4\theta) - \frac{1}{4}r^{-2}\sin(4\theta), & r \geq 1. \end{cases}$$

Again we compare the solution along the line $(x, 3/10)$ for $0 < x < 2$, which crosses the boundary at around $x = 0.954$. The boundary was discretized with N particles (initially equally spaced), and this time the regularization parameter was set to $\epsilon = \Delta s/4$. For other values of ϵ, the results were similar. Figures 12 and 13 show the results. We note that the corners in the solution (gradient discontinuities) are captured well with the method.

The errors using $N = 100$, 200, and 400 are shown in Figure 14. The magnitude of the errors is comparable to those in the previous example, and the convergence appears to be second order as well.

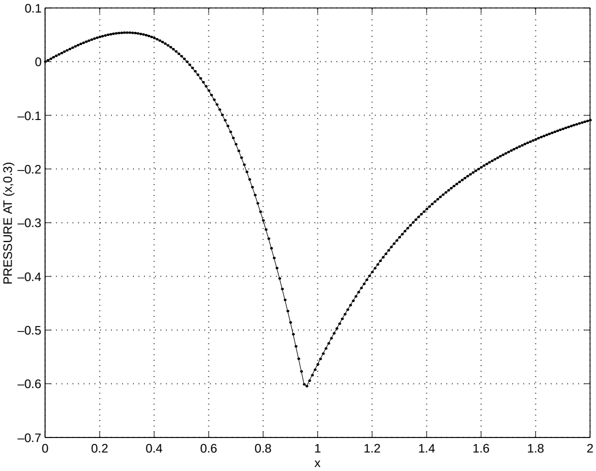


FIG. 12. *Computed pressure p along the line $(x, 3/10)$ for $N = 100$ (dots). Also plotted is the exact solution (solid).*

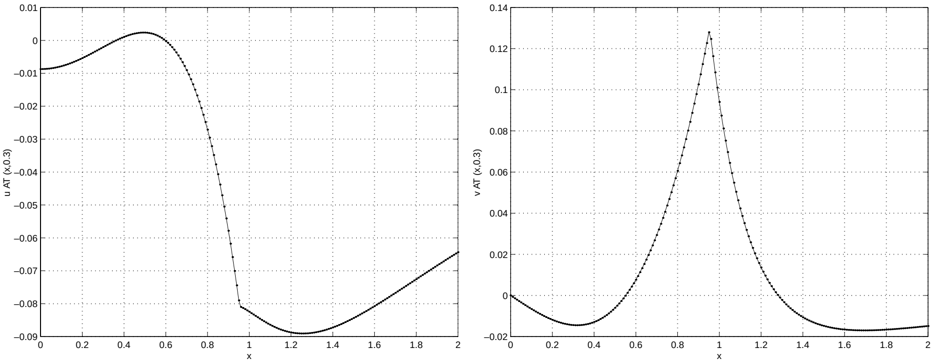


FIG. 13. *Computed velocity components u (left) and v (right) along the line $(x, 3/10)$ for $N = 100$ (dots). Also plotted is the exact solution (solid).*

We make one more comment about the choice of the regularization parameter ϵ in the present context. While it is usually taken as $\epsilon = O(\Delta s)$, the errors in computing integrals of the form

$$\int_0^L \hat{n}(s) \cdot \nabla G(\mathbf{x} - \mathbf{x}(s))[f(s) - f(s^*)]ds$$

or

$$\int_0^L \mathbf{x}'(s) \cdot \nabla G(\mathbf{x} - \mathbf{x}(s))[g(s) - g(s^*)]ds$$

are slightly different when the evaluation point $\mathbf{x}$ is near, but off, the curve. This is because

$$\nabla G(\mathbf{x} - \mathbf{x}(s^*)) = \frac{\mathbf{x} - \mathbf{x}(s^*)}{2\pi|\mathbf{x} - \mathbf{x}(s^*)|^2} = \frac{\hat{n}(s^*)}{2\pi|\mathbf{x} - \mathbf{x}(s^*)|},$$

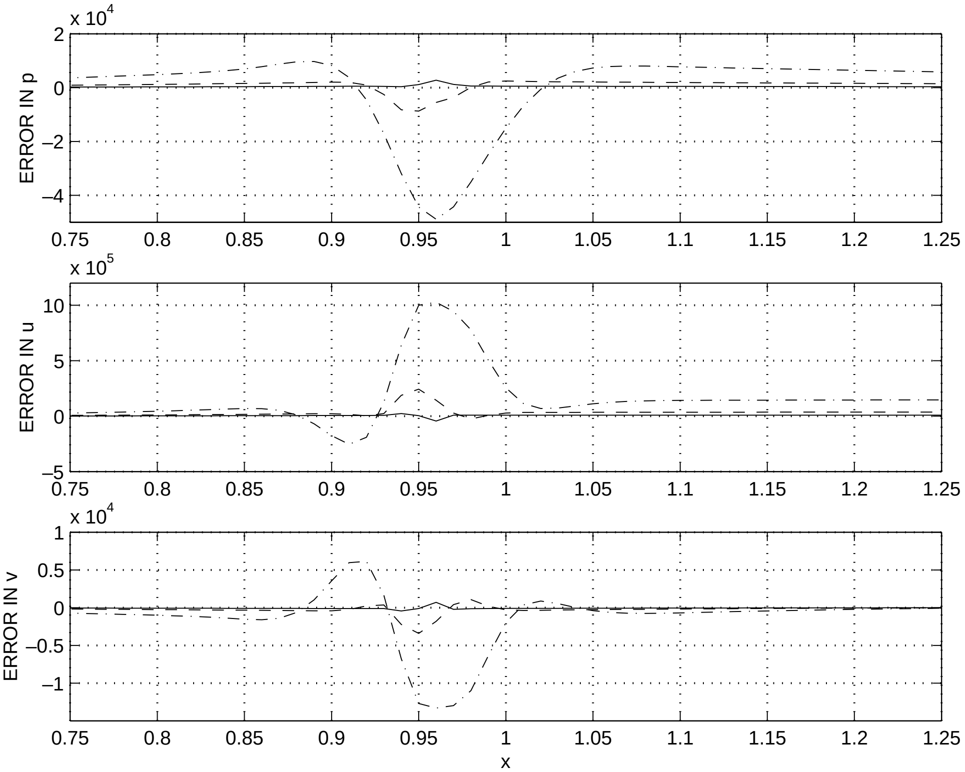


FIG. 14. *Error in the pressure and velocity components for Example* 4*b. The errors are measured along the line* $(x, 3/10)$ *for* $N = 100$, 200, *and* 400.

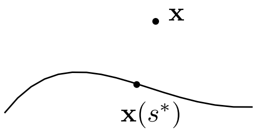


FIG. 15. *Schematic diagram.*

which implies that for $s \approx s^*$ (see Figure 15)

$$\hat{n}(s) \cdot \nabla G(\mathbf{x} - \mathbf{x}(s)) \sim \frac{O(1)}{|\mathbf{x} - \mathbf{x}(s^*)|},$$

while

$$\mathbf{x}'(s) \cdot \nabla G(\mathbf{x} - \mathbf{x}(s)) \sim \frac{O(s - s^*)}{2\pi|\mathbf{x} - \mathbf{x}(s^*)|},$$

so that the first integrand above is “more nearly singular” than the second. This somewhat affects the optimal size of ϵ relative to Δs.

4.3. Numerical example 5: Computing stresses. We consider a final example of flow in a channel obstructed by a semicircular protrusion (see Figure 16). This problem has been studied in [19] as a model for the adhesion of a cell to the wall of a channel and the stresses exerted on these cells by fluid flowing within the channel. Additional numerical studies of this problem are found in [39] and a variant of it

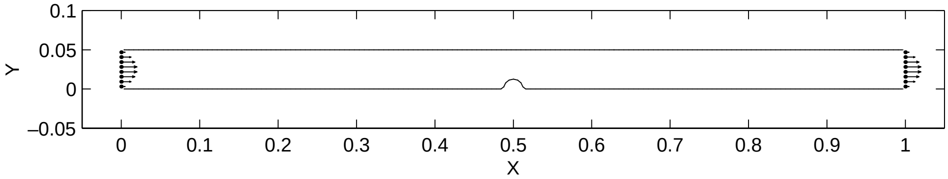


FIG. 16. *Geometry of the problem and velocity boundary conditions.*

in [18]. The objective is to determine the stresses along the channel walls, including the obstruction. It is assumed that the flow is driven by a constant pressure gradient and has reached a steady state. In our case, the corners where the semicircle meets the channel wall have not been smoothed with fillets. The velocity inflow and outflow conditions were assumed to be the parabolic Poiseuille flow, while the fluid velocity at the channel walls was set to zero.

Using the nondimensionalization in [19], the parameters used were $\Delta p = 1$, $\mu = H^2/8L^2$, $L = 1$, and $H = 0.05$, where L and H are the length and height of the channel. The radius of the obstruction was set to $R = H/4$. A particle separation of $\Delta s = 0.0025$ was used along the straight horizontal boundaries and approximately the same on the semicircle and the inflow/outflow ends. The regularization parameter was set to $\epsilon = \Delta s/4$. To simulate the flow induced by the constant pressure gradient, a constant force $(\Delta p, 0)$ was imposed on a grid covering the interior of the channel. The velocity due to this force was computed on the channel walls using (11). As a second step, the forces along the channel walls are computed with (14) so that the desired boundary conditions, including the parabolic inflow/outflow, are obtained. The resulting boundary forces were then used to compute the normal and shear stresses on the bottom part of the channel. This was done by decomposing the forces into the normal and tangential components along the channel walls. The results are shown in Figure 17.

The shear stress has been normalized by the shear stress on a channel without the obstruction. This allows one to see that away from the obstruction, the shear stress approaches that of the free channel. Also apparent from the graph is that the maximum shear stress is nearly three times that of the flat wall. On the other hand, the normal stress is nearly equal to the pressure which decreases linearly from $p(0) = 1$ to $p(L) = 0$. The obstruction merely slightly perturbs this profile. These results are nearly identical to those shown in Figure 5 of [19] with the exception of the values at the points where the semicircle meets the wall. In [19], this corner was smoothed with small fillets to minimize the oscillation observed here, where no smoothing was used.

5. Concluding remarks. We have introduced a numerical technique for the computation of force-driven Stokes flows. At the core of the method is the regularized Stokeslet, which gives the solution due to the regular force field in (3). The standard Stokeslet is recovered in the limit as the regularization parameter vanishes. Several examples were presented that demonstrate the use of the method when the forces are applied at random points, flexible or solid boundaries. The results show that the motion of the immersed boundaries is captured very well. The volume conservation is excellent due to the exact satisfaction of the incompressibility condition in the formula (7). The solution off the boundaries is also computed accurately with a slightly modified formulation in terms of single and double layer potentials. These

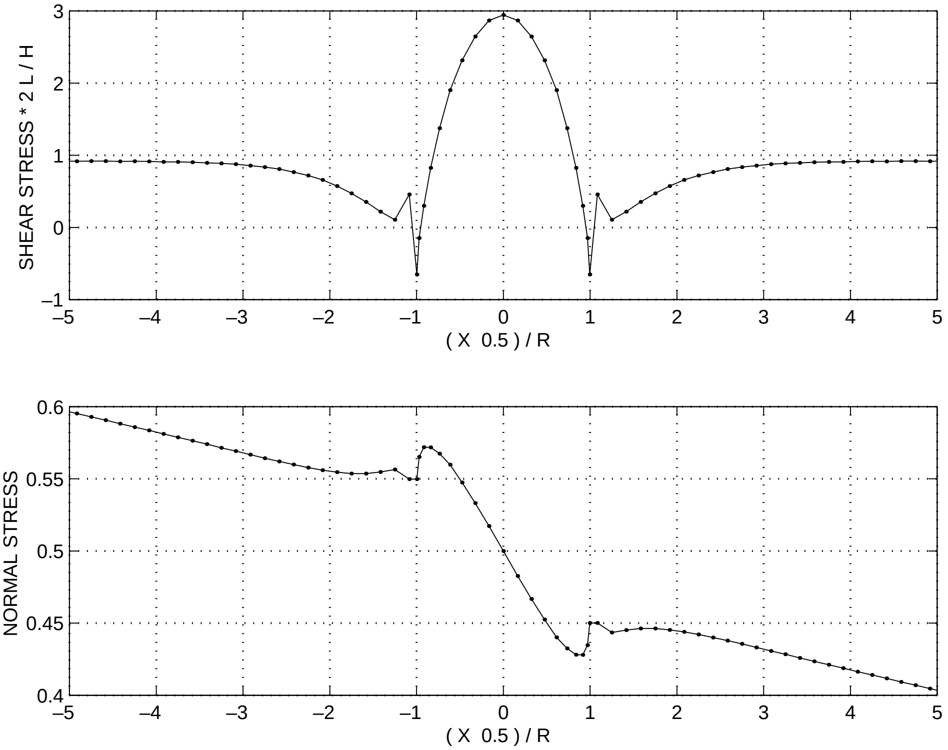


Fig. 17. *Shear and normal stresses along the bottom wall of the channel.*

results can be combined when the velocity in the entire domain is required in order to track other quantities advected by the flow.

Velocity boundary conditions can be satisfied to great precision by inverting the velocity equation and solving for the forces on the boundaries. Thus far, this has been done by the iterative procedure GMRES, although other techniques and associated preconditioners are being considered for future implementations. The resulting forces can also be used to determine stresses along walls, producing profiles without excess noise.

There are other singular solutions of Stokes equations that are also important for many applications. Elements such as dipoles, rotlets, and stresslets can be derived by applying certain differential operators to the Stokeslet. One can derive the regularized versions of these elements from the regularized Stokeslet. Regularized dipoles are already used in other fluid flow applications [5, 10, 35].

Another future direction of research is the extension of this work to three dimensions. In principle, this is already done here, since the derivation of the formulas is identical. For the simulation of bubbles or waving surfaces, these can be combined with existing triangulation or other discretization techniques. We expect to report on this case in a future article.

REFERENCES

[1] G. K. Batchelor, *An Introduction to Fluid Dynamics*, Cambridge University Press, Cambridge, UK, 1967.

[2] J. T. Beale and M.-C. Lai, *A method for computing nearly singular integrals*, SIAM J. Numer.

Anal., 38 (2001), pp. 1902–1925.

[3] J. T. Beale and A. Majda, *High order accurate vortex methods with explicit velocity kernels*, J. Comput. Phys., 58 (1985), pp. 188–208.

[4] J. R. Blake, *A model for the microstructure in ciliated organisms*, J. Fluid Mech., 55 (1972), pp. 1–23.

[5] T. F. Buttke, *Velicity methods: Lagrangian numerical methods which preserve the Hamiltonian structure of incompressible fluid flow*, in Vortex Flows and Related Numerical Methods, J. T. Beale, G.-H. Cottet, and S. Huberson, eds., NATO Adv. Sci. Inst. Ser. C Math. Phys. Sci. 395, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1993, pp. 39–57.

[6] S. Childress, *Mechanics of Swimming and Flying*, Cambridge University Press, Cambridge, UK, 1981.

[7] A. J. Chorin, *Numerical study of slightly viscous flow*, J. Fluid Mech., 57 (1973), pp. 785–796.

[8] R. Cortez, *An impulse-based approximation of fluid motion due to boundary forces*, J. Comput. Phys., 123 (1996), pp. 341–353.

[9] R. Cortez, *Convergence of high-order deterministic particle methods for the convection-diffusion equation*, Comm. Pure Appl. Math., 50 (1997), pp. 1235–1260.

[10] R. Cortez and D. A. Varela, *The dynamics of an elastic membrane using the impulse method*, J. Comput. Phys., 138 (1997), pp. 224–247.

[11] P. Degond and S. Mas-Gallic, *The weighted particle method for convection-diffusion equations part* 1: *The case of an isotropic viscosity*, Math. Comp., 53 (1989), pp. 485–507.

[12] D. Fishelov, *A new vortex scheme for viscous flows*, J. Comput. Phys., 86 (1990), pp. 211–224.

[13] S. Gueron and N. Liron, *Ciliary motion modeling, and dynamic multicilia interactions*, Biophys. J., 63 (1992), pp. 1045–1058.

[14] O. H. Hald, *Convergence of vortex methods for Euler's equations* II, SIAM J. Numer. Anal., 16 (1979), pp. 726–755.

[15] H. Hasimoto, *On the periodic fundamental solutions of the Stokes equations and their applications to viscous flow past a cubic array of spheres*, J. Fluid Mech., 5 (1959), pp. 317–328.

[16] J. J. L. Higdon, *The generation of feeding currents by flagellar motions*, J. Fluid Mech., 94 (1979), pp. 305–330.

[17] J. J. L. Higdon, *A hydrodynamic analysis of flagellar propulsion*, J. Fluid Mech., 90 (1979), pp. 685–711.

[18] J. J. L. Higdon, *Stokes flow in arbitrary two-dimensional domains: Shear flow over ridges and cavities*, J. Fluid Mech., 159 (1985), pp. 195–226.

[19] D. P. Gaver III and S. M. Kute, *A theoretical model study of the influence of fluid stresses on a cell adhering to a microchannel wall*, Biophys. J., 75 (1998), pp. 721–733.

[20] J. B. Keller and S. I. Rubinow, *Slender-body theory for slow viscous flow*, J. Fluid Mech., 75 (1976), pp. 705–714.

[21] R. Krasny, *Desingularization of periodic vortex sheet roll-up*, J. Comput. Phys., 65 (1986), pp. 292–313.

[22] R. J. LeVeque and Z. Li, *Immersed interface methods for Stokes flow with elastic boundaries or surface tension*, SIAM J. Sci. Comput., 18 (1997), pp. 709–735.

[23] J. Lighthill, *Flagellar hydrodynamics*, SIAM Rev., 18 (1976), pp. 161–230.

[24] J. Lighthill, *Helical distributions of Stokeslets*, J. Engrg. Math., 30 (1996), pp. 35–78.

[25] J. Lighthill, *Mathematical Biofluiddynamics*, SIAM, Philadelphia, 1975.

[26] N. Liron and S. Mochon, *Stokes flow for a Stokeslet between two parallel flat plates*, J. Engrg. Math., 10 (1976), pp. 287–303.

[27] N. Liron and R. Shahar, *Stokes flow due to a Stokeslet in a pipe*, J. Fluid Mech., 86 (1978), pp. 727–744.

[28] A. Mayo, *The fast solution of Poisson's and the biharmonic equations on irregular regions*, SIAM J. Numer. Anal., 21 (1984), pp. 285–299.

[29] A. Mayo, *The rapid evaluation of volume integrals of potential theory on general regions*, J. Comput. Phys., 100 (1992), pp. 236–245.

[30] A. Mayo and C. S. Peskin, *An Implicit Numerical Method for Fluid Dynamics Problems with Immersed Elastic Boundaries*, Contemp. Math. 141, AMS, Providence, RI, 1993, pp. 261–277.

[31] A. McKenney, L. Greengard, and A. Mayo, *A fast Poisson solver for complex geometries*, J. Comput. Phys., 118 (1995), pp. 348–355.

[32] C. Pozrikidis, *Boundary Integral and Singularity Methods for Linearized Viscous Flow*, Cambridge University Press, Cambridge, UK, 1992.

[33] C. Pozrikidis, *On the transient motion of ordered suspensions of liquid drops*, J. Fluid Mech., 246 (1993), pp. 301–320.

[34] C. Pozrikidis, *Introduction to Theoretical and Computational Fluid Dynamics*, Oxford Uni-

versity Press, Oxford, UK, 1997.

[35] M. C. Recchioni and G. Russo, *Hamilton-based numerical methods for a fluid-membrane interaction in two and three dimensions*, SIAM J. Sci. Comput., 19 (1998), pp. 861–892.

[36] M. Shelley and T. Ueda, *The Stokesian hydrodynamics of flexing, stretching filaments*, Phys. D, 146 (2000), pp. 221–245.

[37] J. Strain, *Fast potential theory* II: *Layer potentials and discrete sums*, J. Comput. Phys., 99 (1992), pp. 251–270.

[38] C. Tu and C. S. Peskin, *Stability and instability in the computation of flows with moving immersed boundaries: A comparison of three methods*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 1361–1376.

[39] H. A. R. Williams, L. J. Fauci, and D. P. Gaver III, *Evaluation of interfacial fluid dynamical stresses using the immersed boundary method*, SIAM J. Sci. Comput., submitted.

SIAM J. Sci. Comput.
Vol. 23, No. 4, pp. 1226–1252

hp-ADAPTIVE DISCONTINUOUS GALERKIN FINITE ELEMENT METHODS FOR FIRST-ORDER HYPERBOLIC PROBLEMS*

PAUL HOUSTON† AND ENDRE SÜLI‡

Abstract. We consider the a posteriori error analysis of *hp*-discontinuous Galerkin finite element approximations to first-order hyperbolic problems. In particular, we discuss the question of error estimation for linear functionals, such as the outflow flux and the local average of the solution. Based on our a posteriori error bound we design and implement the corresponding adaptive algorithm to ensure reliable and efficient control of the error in the prescribed functional to within a given tolerance. This involves exploiting both local polynomial-degree variation and local mesh subdivision. The theoretical results are illustrated by a series of numerical experiments.

Key words. *hp*-finite element methods, a posteriori error analysis, hyperbolic problems

AMS subject classifications. 65N12, 65N15, 65N30

PII. S1064827500378799

1. Introduction.

Adaptive finite element methods that are capable of exploiting both local polynomial-degree variation (p-refinement) and local mesh subdivision (h-refinement) offer greater flexibility and improved efficiency than mesh refinement methods which incorporate only h-refinement or p-refinement in isolation. The aim of this paper is to develop the a posteriori error analysis of the *hp*-version of the discontinuous Galerkin finite element method; see [9], and the references cited therein, and [6] for earlier work in this area. In particular, we shall be concerned with the derivation of computable error bounds for linear functionals of the solution to scalar first-order hyperbolic problems. Relevant examples of linear functionals of the solution include the mean value of the field over the computational domain and the normal flux through the outflow boundary.

The paper is structured as follows. In section 2 we introduce the model problem and formulate its discontinuous Galerkin finite element approximation. Then, in section 3, we derive both a posteriori and a priori error bounds for linear functionals of the solution. Our a posteriori error bounds stem from a hyperbolic duality argument and include computable residual terms multiplied by local weights involving the dual solution; cf. [14, 19]. Guided by our a posteriori error analysis, in section 4 we design an *hp*-adaptive finite element algorithm to guarantee both reliable and efficient control of the error in the computed functional with respect to a fixed user-defined tolerance. A key question in *hp*-adaptive algorithms is how to automatically decide when to h-refine/derefine and when to p-refine/derefine. Here, stimulated by the work of Ainsworth and Senior [2], we develop a decision mechanism based on the estimation of local Sobolev regularities of the primal and dual solutions by means of the a priori error bounds from section 3 (see also [23]). The performance of the resulting *hp*-refinement strategy is then studied in section 5 through a series of numerical ex-

*Received by the editors September 29, 2000; accepted for publication June 11, 2001; published electronically December 4, 2001.

http://www.siam.org/journals/sisc/23-4/37879.html

†Department of Mathematics and Computer Science, University of Leicester, Leicester LE1 7RH, UK (Paul.Houston@mcs.le.ac.uk). This author was supported by Engineering and Physical Sciences Research Council grant GR/N24230.

‡University of Oxford, Computing Laboratory, Wolfson Building, Parks Road, Oxford OX1 3QD, UK (Endre.Suli@comlab.ox.ac.uk).

periments. In particular, we demonstrate the superiority of using *hp*-adaptive mesh refinement over the traditional *h*-refinement method, where the degree of the approximating polynomial is kept fixed at some low value. Finally, in section 6 we summarize the work presented in this paper and draw some conclusions.

2. Model problem and discretization.

Let Ω be a bounded open polyhedral domain in $\mathbb{R}^d$, $d \geq 2$, and let Γ denote the union of open faces of Ω. Suppose that $\mathbf{b} = (b_1, \ldots, b_d)$ is a d-component vector function defined on $\bar{\Omega}$ with $b_i \in W^1_\infty(\Omega)$, $i = 1, \ldots, d$, and consider the following subsets of Γ:

$$\Gamma_- = \{x \in \Gamma : \mathbf{b}(x) \cdot \mathbf{n}(x) < 0\}, \qquad \Gamma_+ = \{x \in \Gamma : \mathbf{b}(x) \cdot \mathbf{n}(x) > 0\};$$

here, $\mathbf{n}(x)$ denotes the unit outward normal vector to Γ at $x \in \Gamma$. The sets Γ_- and Γ_+ are referred to as the inflow and outflow boundary, respectively. We shall suppose that Γ is a.e. noncharacteristic in the sense that the set $\Gamma \setminus (\Gamma_- \cup \Gamma_+)$ has $(d-1)$-dimensional measure zero. Given $c \in L_\infty(\Omega)$, $f \in L_2(\Omega)$, and $g \in L_2(\Gamma_-)$, we consider the following boundary value problem: find $u \in H(\mathcal{L}, \Omega)$ such that

$$\begin{aligned} \mathcal{L}u \equiv \mathbf{b} \cdot \nabla u + cu = f, &\quad x \in \Omega\ , \\ u = g, &\quad x \in \Gamma_-\ , \end{aligned} \tag{2.1}$$

where $H(\mathcal{L}, \Omega) = \{v \in L_2(\Omega) : \mathcal{L}v \in L_2(\Omega)\}$ denotes the *graph space* of the partial differential operator $\mathcal{L}$ in $L_2(\Omega)$. In addition, we adopt the following (standard) hypothesis: there exists a vector $\xi \in \mathbb{R}^d$ and a positive real number δ such that

$$c - \frac{1}{2}\nabla \cdot \mathbf{b} + \mathbf{b} \cdot \xi \geq \delta^2 \ \text{ for a.e. } x \in \Omega\ . \tag{2.2}$$

We note that assumption (2.2) ensures the uniqueness of a solution $u \in H(\mathcal{L}, \Omega)$ to (2.1); cf., for example, [15]. General results concerning the existence and uniqueness of solutions to boundary value problems for first-order hyperbolic equations are given in [4], and [10, pp. 215–262]. For the sake of simplicity of presentation, we assume that (2.2) is satisfied with $\xi = 0$, and we define $c_0 \in L_\infty(\Omega)$ by

$$c_0^2(x) = c(x) - \frac{1}{2}\nabla \cdot \mathbf{b}(x), \qquad x \in \Omega\ . \tag{2.3}$$

Clearly, $c_0(x) \geq \delta > 0$ for almost every $x \in \Omega$.

2.1. Finite element spaces.

Suppose that $\mathcal{T}$ is a regular or 1-irregular subdivision of Ω into disjoint open element domains κ such that $\bar{\Omega} = \cup_{\kappa \in \mathcal{T}} \bar{\kappa}$. Thus a $(d-1)$-dimensional face of each element κ in $\mathcal{T}$ is allowed to contain at most one hanging (irregular) node—typically the barycenter of the face. It will be assumed that $\mathcal{T}$ respects the decomposition of Γ into Γ_- and Γ_+ in the following sense: if a $(d-1)$-dimensional open face e of an element κ lies on Γ, then e is a subset of either Γ_- or Γ_+. We shall suppose that the family of subdivisions $\mathcal{T}$ is shape-regular (cf. pp. 61 and 113 and Remark 2.2 on p. 114 in [8], for example) and that each $\kappa \in \mathcal{T}$ is a smooth bijective image of a fixed master element $\hat{\kappa}$; that is, $\kappa = F_\kappa(\hat{\kappa})$ for all $\kappa \in \mathcal{T}$, where $\hat{\kappa}$ is either the open unit simplex or the open unit hypercube in $\mathbb{R}^d$. On the reference element $\hat{\kappa}$ we define spaces of polynomials of degree $p \geq 1$ as follows:

$$\mathcal{Q}_p = \operatorname{span}\{\hat{x}^\alpha : 0 \leq \alpha_i \leq p,\ 1 \leq i \leq d\}, \qquad \mathcal{P}_p = \operatorname{span}\{\hat{x}^\alpha : 0 \leq |\alpha| \leq p\}.$$

To each $\kappa \in \mathcal{T}$ we assign an integer $p_\kappa \geq 1$; collecting the p_κ and F_κ in the vectors $\mathbf{p} = \{p_\kappa : \kappa \in \mathcal{T}\}$ and $\mathbf{F} = \{F_\kappa : \kappa \in \mathcal{T}\}$, respectively, we introduce the finite element space

$$\begin{aligned} S^{\mathbf{p}}(\Omega, \mathcal{T}, \mathbf{F}) = \{u \in L_2(\Omega) \,:\, & u|_\kappa \circ F_\kappa \in \mathcal{Q}_{p_\kappa} \text{ if } F_\kappa^{-1}(\kappa) \text{ is the open unit hypercube,} \\ \text{and} \quad & u|_\kappa \circ F_\kappa \in \mathcal{P}_{p_\kappa} \text{ if } F_\kappa^{-1}(\kappa) \text{ is the open unit simplex; } \kappa \in \mathcal{T}\} \,. \end{aligned}$$

Assuming that $\mathcal{T}$ is a subdivision of Ω, we consider the broken Sobolev space $H^{\mathbf{s}}(\Omega, \mathcal{T})$ of composite index $\mathbf{s}$ with nonnegative components s_κ, $\kappa \in \mathcal{T}$, defined by

$$H^{\mathbf{s}}(\Omega, \mathcal{T}) = \{u \in L_2(\Omega) : \; u|_\kappa \in H^{s_\kappa}(\kappa) \quad \forall \kappa \in \mathcal{T}\} \,.$$

If $s_\kappa = s \geq 0$ for all $\kappa \in \mathcal{T}$, we shall simply write $H^s(\Omega, \mathcal{T})$.

In the next section, we formulate the *hp*-version of the discontinuous Galerkin finite element method (*hp*-DGFEM, for short) for the numerical solution of (2.1).

2.2. The *hp*-discontinuous Galerkin method. Given that κ is an element in the subdivision $\mathcal{T}$, we denote by $\partial\kappa$ the union of $(d-1)$-dimensional open faces of κ. This is nonstandard notation in that $\partial\kappa$ is a subset of the boundary of κ. Let $x \in \partial\kappa$ and suppose that $\mathbf{n}_\kappa(x)$ denotes the unit outward normal vector to $\partial\kappa$ at x. With these conventions, we define the inflow and outflow parts of $\partial\kappa$, respectively, by

$$\partial_-\kappa = \{x \in \partial\kappa : \; \mathbf{b}(x) \cdot \mathbf{n}_\kappa(x) < 0\} \,, \quad \partial_+\kappa = \{x \in \partial\kappa : \; \mathbf{b}(x) \cdot \mathbf{n}_\kappa(x) \geq 0\} \,. \tag{2.4}$$

For each $\kappa \in \mathcal{T}$ and any $v \in H^1(\kappa)$ we denote by v_κ^+ the interior trace of v on $\partial\kappa$ (the trace taken from within κ). Now consider an element κ such that the set $\partial_-\kappa\backslash\Gamma_-$ is nonempty; then, for each $x \in \partial_-\kappa\backslash\Gamma_-$ (with the exception of a set of $(d-1)$-dimensional measure zero), there exists a unique element κ', depending on the choice of x, such that $x \in \partial_+\kappa'$. Suppose that $v \in H^1(\Omega, \mathcal{T})$. If $\partial_-\kappa\backslash\Gamma_-$ is nonempty for some $\kappa \in \mathcal{T}$, then we define the outer trace v_κ^- of v on $\partial_-\kappa\backslash\Gamma_-$ relative to κ as the inner trace $v_{\kappa'}^+$ relative to those elements κ' for which $\partial_+\kappa'$ has intersection with $\partial_-\kappa\backslash\Gamma_-$ of positive $(d-1)$-dimensional measure. We also introduce the jump of v across $\partial_-\kappa\backslash\Gamma_-$: $[\,v\,]_\kappa = v_\kappa^+ - v_\kappa^-$. Since below it will always be clear from the context which element κ in the subdivision $\mathcal{T}$ the quantities $\mathbf{n}_\kappa$, v_κ^+, v_κ^- and $[v]_\kappa$ correspond to, for the sake of notational simplicity we shall suppress the letter κ in the subscript and instead write $\mathbf{n}$, v^+, v^-, and $[v]$, respectively.

For $v, w \in H^1(\Omega, \mathcal{T})$, we define the bilinear form

$$\begin{aligned} B_{\mathrm{DG}}(w, v) = \sum_{\kappa \in \mathcal{T}} \int_\kappa \mathcal{L}w\, v \;\mathrm{d}x - \sum_{\kappa \in \mathcal{T}} \int_{\partial_-\kappa\backslash\Gamma_-} (\mathbf{b} \cdot \mathbf{n})[w]\, v^+ \;\mathrm{d}s \\ - \sum_{\kappa \in \mathcal{T}} \int_{\partial_-\kappa\cap\Gamma_-} (\mathbf{b} \cdot \mathbf{n}) w^+\, v^+ \;\mathrm{d}s, \end{aligned}$$

and, for $v \in H^1(\Omega, \mathcal{T})$, we consider the linear functional

$$\ell_{\mathrm{DG}}(v) = \sum_{\kappa \in \mathcal{T}} \int_\kappa f v \;\mathrm{d}x - \sum_{\kappa \in \mathcal{T}} \int_{\partial_-\kappa\cap\Gamma_-} (\mathbf{b} \cdot \mathbf{n})\, g v^+ \;\mathrm{d}s \,.$$

The *hp*-DGFEM for (2.1) is defined as follows: find $u_{\mathrm{DG}} \in S^{\mathbf{p}}(\Omega, \mathcal{T}, \mathbf{F})$ such that

$$B_{\mathrm{DG}}(u_{\mathrm{DG}}, v) = \ell_{\mathrm{DG}}(v) \quad \forall v \in S^{\mathbf{p}}(\Omega, \mathcal{T}, \mathbf{F}) \,. \tag{2.5}$$

We note that

$$\begin{aligned}B_{\mathrm{DG}}(v,v) &= \sum_\kappa \int_\kappa c_0^2(x)|v|^2\,\mathrm{d}x + \frac{1}{2}\sum_\kappa \int_{\partial_-\kappa\cap\Gamma_-} |\mathbf{b}\cdot\mathbf{n}|\,|v^+|^2\,\mathrm{d}s \\ &+\frac{1}{2}\sum_\kappa \int_{\partial_-\kappa\setminus\Gamma_-} |\mathbf{b}\cdot\mathbf{n}|\,|[v]|^2\,\mathrm{d}s + \frac{1}{2}\sum_\kappa \int_{\partial_+\kappa\cap\Gamma_+} |\mathbf{b}\cdot\mathbf{n}|\,|v^+|^2\,\mathrm{d}s\end{aligned}$$

for all $v \in H^1(\Omega,\mathcal{T})$ (see, for example, (2.31) in [16]), and therefore $B_{\mathrm{DG}}(v,v) > 0$ for all $v \in H^1(\Omega,\mathcal{T}) \setminus \{0\}$; thus, recalling that (2.5) is a linear problem on the finite-dimensional linear space $S^{\mathbf{p}}(\Omega,\mathcal{T},\mathbf{F})$, we deduce the existence of a unique solution u_{DG} in $S^{\mathbf{p}}(\Omega,\mathcal{T},\mathbf{F})$ to (2.5). Further, for each $v \in H^1(\Omega,\mathcal{T})$,

$$\begin{aligned}|\ell_{\mathrm{DG}}(v)| &\le \left(\sum_{\kappa\in\mathcal{T}} \int_\kappa c_0^{-2}(x)|f|^2\,\mathrm{d}x + 2\sum_{\kappa\in\mathcal{T}} \int_{\partial_-\kappa\cap\Gamma_-} |\mathbf{b}\cdot\mathbf{n}|\,|g|^2\,\mathrm{d}s\right)^{1/2} \\ &\times \left(\sum_{\kappa\in\mathcal{T}} \int_\kappa c_0^2(x)|v|^2\,\mathrm{d}x + \frac{1}{2}\sum_{\kappa\in\mathcal{T}} \int_{\partial_-\kappa\cap\Gamma_-} |\mathbf{b}\cdot\mathbf{n}|\,|v^+|^2\,\mathrm{d}s\right)^{1/2}.\end{aligned}$$

Hence, letting $|||v|||_{\mathrm{DG}} = [B_{\mathrm{DG}}(v,v)]^{1/2}$, we deduce the stability of the method in the sense that

$$|||u_{\mathrm{DG}}|||_{\mathrm{DG}} \le \left(\sum_{\kappa\in\mathcal{T}} \int_\kappa c_0^{-2}(x)|f(x)|^2\,\mathrm{d}x + 2\sum_{\kappa\in\mathcal{T}} \int_{\partial_-\kappa\cap\Gamma_-} |\mathbf{b}(x)\cdot\mathbf{n}(x)|\,|g(s)|^2\,\mathrm{d}s\right)^{1/2}.$$

For the a priori error analysis of (2.5) in the DG-norm $|||\cdot|||_{\mathrm{DG}}$ we refer to the paper [15]; cf. Lemma 3.9 below. The analysis of the scheme (2.5) with the addition of streamline-diffusion stabilization has been considered in [16]; see also Bey and Oden [6]. Here we shall concentrate on the situation when there is no streamline-diffusion stabilization in the method.

3. A posteriori and a priori error analysis.

In many problems of physical importance the quantity of interest is a linear functional $J(\cdot)$ of the solution. Relevant examples include the lift and drag coefficients for a body immersed into an inviscid fluid, the local mean value of the field, or its flux through the outflow boundary of the computational domain.

Suppose that we wish to control the discretization error in some linear functional $J(\cdot)$ defined on a linear space which contains $H(\mathcal{L},\Omega) + S^{\mathbf{p}}(\Omega,\mathcal{T},\mathbf{F})$. Following the argument presented in [14] for stabilized continuous finite element approximations, we do so by deriving an a posteriori bound on the error between $J(u)$ and $J(u_{\mathrm{DG}})$. We begin our analysis by considering the following *dual* or *adjoint* problem: find z in $H(\mathcal{L}^*,\Omega)$ such that

$$B_{\mathrm{DG}}(w,z) = J(w) \quad \forall w \in H(\mathcal{L},\Omega)\,, \tag{3.1}$$

where $H(\mathcal{L}^*,\Omega)$ denotes the graph space of the adjoint operator $\mathcal{L}^*$ in $L_2(\Omega)$. Let us assume that (3.1) possesses a unique solution. Clearly, the validity of this assumption depends on the choice of the linear functional under consideration. Below, we present some important examples which are covered by our hypotheses.

Example 1: *Mean flow.* Consider approximating the (weighted) mean value $J(\cdot) \equiv M_\psi(\cdot)$ defined by

$$J(w) \equiv M_\psi(w) = \int_\Omega w\psi\,\mathrm{d}x\,, \tag{3.2}$$

where $\psi \in L_2(\Omega)$ is a given weight function. In this case the dual problem takes the following form: find z in $H(\mathcal{L}^*, \Omega)$ such that

$$\begin{aligned} \mathcal{L}^* z \equiv -\nabla \cdot (\mathbf{b} z) + cz = \psi, \quad & x \in \Omega\,, \\ z = 0, \quad & x \in \Gamma_+\,. \end{aligned} \tag{3.3}$$

Example 2: *Point value.* Under the assumption that the analytical solution u is a continuous function in the neighborhood of a given point x_c in Ω, the point value $u(x_c)$ may also be approximated. However, now $J(w) = w(x_c)$, and when this is inserted as the right-hand side into the dual problem (3.1), the resulting weak solution z is a measure rather than a regular distribution; in particular, z does not belong to $L_2(\Omega)$. Thus, to avoid technical complications, we mollify the functional J by considering a nonnegative function φ in $L_{1,\mathrm{loc}}(\mathbb{R}^d)$ whose support is contained in the unit ball $B(0,1)$ centered at $x = 0$ and such that the integral of φ over $B(0,1)$ is equal to 1. Writing $\psi(x) = \varphi_\varepsilon(x) \equiv \varepsilon^{-d}\varphi((x - x_c)/\varepsilon)$, $M_\psi(u)$ converges to $u(x_c)$ as $\varepsilon \to 0$. Further, setting $J(w) = M_\psi(w)$ into (3.1) as the right-hand side, for $0 \ll \varepsilon < 1$ fixed, now results in a unique solution z in $H(\mathcal{L}^*, \Omega)$.

Example 3: *Outflow normal flux.* As a final example, consider the (weighted) normal flux $J(\cdot) \equiv N_\psi(\cdot)$ through the outflow boundary Γ_+, defined, for $w \in H(\mathcal{L}, \Omega)$, by

$$J(w) \equiv N_\psi(w) = \int_{\Gamma_+} (\mathbf{b} \cdot \mathbf{n})\, w\psi \,\mathrm{d}s\,, \tag{3.4}$$

where ψ is a given "weight" function in $L_2(\Gamma_+)$. A simple calculation based on the divergence theorem shows that z is the (unique) solution to the following boundary value problem: find z in $H(\mathcal{L}^*, \Omega)$ such that

$$\begin{aligned} \mathcal{L}^* z \equiv -\nabla \cdot (\mathbf{b} z) + cz = 0, \quad & x \in \Omega\,, \\ z = \psi, \quad & x \in \Gamma_+\,. \end{aligned}$$

For a given linear functional $J(\cdot)$ the proceeding a posteriori error bound will be expressed in terms of the finite element residual $r_{h,p}$ defined on $\kappa \in \mathcal{T}$ by

$$r_{h,p}|_\kappa = (f - \mathcal{L} u_{\mathrm{DG}})|_\kappa,$$

which measures the extent to which u_{DG} fails to satisfy the differential equation on the union of the elements κ in the mesh $\mathcal{T}$; thus we refer to $r_{h,p}$ as the *internal residual.* Also, since u_{DG} satisfies only the boundary conditions approximately, the difference $g - u_{\mathrm{DG}}$ is not necessarily zero on Γ_-; thus, for each element κ with $\partial_-\kappa \cap \Gamma_-$ of positive $(d-1)$-dimensional measure, we define the *boundary residual* $r_{h,p-}$ by

$$r_{h,p-}|_{\partial_-\kappa\cap\Gamma_-} = (g - u_{\mathrm{DG}}^+)|_{\partial_-\kappa\cap\Gamma_-}\,.$$

With these definitions we observe from (2.5) the following relationship between the internal and boundary residual:

$$\begin{aligned} B_{\mathrm{DG}}(u - u_{\mathrm{DG}}, v) \equiv & \sum_{\kappa\in\mathcal{T}} (r_{h,p}, v)_\kappa + \sum_{\kappa\in\mathcal{T}} ((\mathbf{b}\cdot\mathbf{n})[u_{\mathrm{DG}}], v^+)_{\partial_-\kappa\setminus\Gamma_-} \\ & - \sum_{\kappa\in\mathcal{T}} ((\mathbf{b}\cdot\mathbf{n}) r_{h,p-}, v^+)_{\partial_-\kappa\cap\Gamma_-} = 0 \end{aligned} \tag{3.5}$$

for all v in $S^{\mathbf{p}}(\Omega, \mathcal{T}, \mathbf{F})$. The identity (3.5) is referred to as the *Galerkin orthogonality* property of the *hp*-DGFEM. We remark that the second term on the right-hand side of (3.5) reflects the fact that while the normal flux $(\mathbf{b}u) \cdot \mathbf{n} = (\mathbf{b} \cdot \mathbf{n})u$ of the analytical solution u is continuous across element interfaces (even if the analytical solution u is only piecewise continuous on Ω), the normal flux of the numerical solution u_{DG} is *not*, due to the choice of the finite element space $S^{\mathbf{p}}(\Omega, \mathcal{T}, \mathbf{F})$; for finite element approximations to (2.1) based on continuous piecewise polynomials, this term is, of course, equal to zero.

3.1. Type I a posteriori error bound. The starting point of the a posteriori error analysis is the following general result.

THEOREM 3.1. *Let u and u_{DG} denote the solutions of* (2.1) *and* (2.5)*, respectively, and suppose that the dual solution z is defined by* (3.1)*. Then, the following error representation formula holds:*

$$\begin{aligned} J(u) - J(u_{\mathrm{DG}}) = {} & \sum_{\kappa \in \mathcal{T}} (r_{h,p}, z - z_{h,p})_\kappa + \sum_{\kappa \in \mathcal{T}} ((\mathbf{b} \cdot \mathbf{n})[u_{\mathrm{DG}}], (z - z_{h,p})^+)_{\partial_-\kappa \setminus \Gamma_-} \\ & - \sum_{\kappa \in \mathcal{T}} ((\mathbf{b} \cdot \mathbf{n}) r_{h,p-}, (z - z_{h,p})^+)_{\partial_-\kappa \cap \Gamma_-} \\ \equiv {} & \mathcal{E}_\Omega(u_{\mathrm{DG}}, h, p, z - z_{h,p}) \end{aligned} \tag{3.6}$$

for all $z_{h,p}$ in $S^{\mathbf{p}}(\Omega, \mathcal{T}, \mathbf{F})$.

Proof. On choosing $w = u - u_{\mathrm{DG}}$ in (3.1) and recalling the linearity of $J(\cdot)$ and the Galerkin orthogonality property (3.5), we deduce that

$$\begin{aligned} J(u) - J(u_{\mathrm{DG}}) &= J(u - u_{\mathrm{DG}}) = B_{\mathrm{DG}}(u - u_{\mathrm{DG}}, z) = B_{\mathrm{DG}}(u - u_{\mathrm{DG}}, z - z_{h,p}) \\ &= \sum_{\kappa \in \mathcal{T}} (r_{h,p}, z - z_{h,p})_\kappa + \sum_{\kappa \in \mathcal{T}} ((\mathbf{b} \cdot \mathbf{n})[u_{\mathrm{DG}}], (z - z_{h,p})^+)_{\partial_-\kappa \setminus \Gamma_-} \\ &\quad - \sum_{\kappa \in \mathcal{T}} ((\mathbf{b} \cdot \mathbf{n}) r_{h,p-}, (z - z_{h,p})^+)_{\partial_-\kappa \cap \Gamma_-} \, , \end{aligned}$$

and hence (3.6). □

Given a linear functional $J(\cdot)$ and a positive tolerance TOL, the aim of the computation is to calculate u_{DG} such that

$$|J(u) - J(u_{\mathrm{DG}})| \leq \mathtt{TOL} \, . \tag{3.7}$$

According to (3.6), a necessary and sufficient condition for this to hold is that the *stopping criterion*

$$|\mathcal{E}_\Omega(u_{\mathrm{DG}}, h, p, z - z_{h,p})| \leq \mathtt{TOL} \tag{3.8}$$

is satisfied. If (3.8) holds, then $J(u_{\mathrm{DG}})$ is accepted as an accurate representation of $J(u)$; otherwise, u_{DG} is discarded and a new, improved approximation is computed on a refined subdivision. In order to ensure that the subdivision is refined only where necessary, a decision has to be made on each element κ as to whether the local mesh size h_κ and the local polynomial degree p_κ are acceptable in relation to TOL. A convenient approach to obtaining a local refinement criterion which relates the local discretization parameters h_κ and p_κ to TOL is to *localize* $\mathcal{E}_\Omega(u_{\mathrm{DG}}, h, p, z - z_{h,p})$. More precisely, $|\mathcal{E}_\Omega(u_{\mathrm{DG}}, h, p, z - z_{h,p})|$ is further bounded above by noting that

$\mathcal{E}_\Omega(u_{\rm DG}, h, p, z - z_{h,p})$ can be decomposed as $\sum_\kappa \chi_\kappa$, with

$$\chi_\kappa = (r_{h,p}, z - z_{h,p})_\kappa + ((\mathbf{b}\cdot\mathbf{n})[u_{\rm DG}], (z - z_{h,p})^+)_{\partial_-\kappa\backslash\Gamma_-} \\ +((\mathbf{b}\cdot\mathbf{n})r_{h,p-}, (z - z_{h,p})^+)_{\partial_-\kappa\cap\Gamma_-} \,,$$

and applying the inequality $|\sum_\kappa \chi_\kappa| \leq \sum_\kappa |\chi_\kappa|$. Thus we arrive at the following result.

COROLLARY 3.2. *Under the assumptions of Theorem* 3.1, *the following a posteriori error bound holds:*

$$|J(u) - J(u_{\rm DG})| \leq \mathcal{E}(u_{\rm DG}, h, p, z - z_{h,p}) \,, \tag{3.9}$$

where

$$\mathcal{E}(u_{\rm DG}, h, p, z - z_{h,p}) \equiv \sum_{\kappa\in\mathcal{T}} \eta_\kappa \,, \tag{3.10}$$

and

$$\eta_\kappa = |(r_{h,p}, z - z_{h,p})_\kappa + ((\mathbf{b}\cdot\mathbf{n})[u_{\rm DG}], (z - z_{h,p})^+)_{\partial_-\kappa\backslash\Gamma_-} \\ -((\mathbf{b}\cdot\mathbf{n})r_{h,p-}, (z - z_{h,p})^+)_{\partial_-\kappa\cap\Gamma_-}| \,. \tag{3.11}$$

Now, a possible local refinement criterion might, for example, consist of checking whether, on each element κ in the subdivision $\mathcal{T}$, the inequality

$$\eta_\kappa \leq \frac{\texttt{TOL}}{N} \tag{3.12}$$

holds, where N is the number of elements in $\mathcal{T}$. If (3.12) is valid on each element κ in $\mathcal{T}$, then, according to (3.9), the stopping criterion (3.8) has been reached and the required error control (3.7) has been achieved. The a posteriori error bound (3.9)–(3.11), where the difference between the dual solution z and its discrete representation (e.g. interpolant, quasi-interpolant, or projection) $z_{h,p}$ defined on the primal subdivision $\mathcal{T}$ enters into the error estimate as local weight function, will be referred to as an a posteriori error bound of Type I.

It may seem natural to further bound η_κ in (3.11) by writing

$$\eta_\kappa \leq |(r_{h,p}, z - z_{h,p})_\kappa| + |((\mathbf{b}\cdot\mathbf{n})[u_{\rm DG}], (z - z_{h,p})^+)_{\partial_-\kappa\backslash\Gamma_-}| \\ +|((\mathbf{b}\cdot\mathbf{n})r_{h,p-}, (z - z_{h,p})^+)_{\partial_-\kappa\cap\Gamma_-}| \equiv \zeta_\kappa \,.$$

We may then consider the a posteriori error bound

$$|J(u) - J(u_{\rm DG})| \leq \sum_{\kappa\in\mathcal{T}} \zeta_\kappa \,. \tag{3.13}$$

Indeed, we could proceed even further, to eliminate $z_{h,p}$ from the a posteriori error estimate, by bounding each term in ζ_κ via the Cauchy–Schwarz inequality and standard results from approximation theory to estimate the expressions $\|z - z_{h,p}\|_{L_2(\kappa)}$ and $\|(z - z_{h,p})^+\|_{L_2(\partial_-\kappa)}$ in terms of the discretization parameters h_κ and p_κ and Sobolev seminorms of z. For then, finally, the dual solution z may also be eliminated from the a posteriori error estimate by bounding norms of z by suitable norms of the data for the dual problem via hyperbolic well-posedness results. The resulting a posteriori error bound will then, in the spirit of Erikson et al. [11], involve only the residual terms

r_{hp}, $(\mathbf{b}\cdot\mathbf{n})[u_{\mathrm{DG}}]$ and $r_{h,p-}$, the discretization parameters, the interpolation constants, and the stability factor of the dual problem; such an estimate will be referred to here as a Type II a posteriori error bound. We note that the residual-based weighted a posteriori error bounds of Becker and Rannacher [5] are intermediate between Type I and Type II bounds, although, due to the locality of the weight, closer in spirit to those of Type I. For earlier work on Type I and Type II error bounds for finite element and finite volume approximations to linear hyperbolic problems, we refer to [14, 22].

The seemingly harmless transition from the Type I error bound (3.9)–(3.13) and the subsequent elimination of the dual solution z *en route* to a Type II bound can be detrimental: due to loss of global cancellations the resulting Type II bound may, under mesh refinement, exhibit a rate of convergence inferior to that of $|J(u)-J(u_{\mathrm{DG}})|$, resulting in uneconomical meshes and an inefficient adaptive algorithm (see, for example, [14]). For this reason we refrain from bounding the terms η_κ further; in particular, we shall not attempt to eliminate the dual solution z from our a posteriori error bound by exploiting standard approximation results and the strong stability of the dual problem. Instead, in the practical implementation of (3.9) we replace z in the bound (3.9) by an approximation $\tilde{z}_{\mathrm{DG}}$, computed by an *hp*-adaptive discontinuous Galerkin finite element method on a sequence of auxiliary (dual) subdivisions $\tilde{\mathcal{T}}$ of the computational domain Ω; see section 4.1 below. We then define $z_{h,p}$ as the $L_2(\Omega)$ projection of $\tilde{z}_{\mathrm{DG}}$ onto the "primal" finite element space $S^{\mathbf{p}}(\Omega,\mathcal{T},\mathbf{F})$ over the "primal" subdivision $\mathcal{T}$. Thus, after decomposing the error representation formula derived in Theorem 3.1 into terms which are computable, i.e., those involving the numerical approximation $\tilde{z}_{\mathrm{DG}}$ to the dual problem, and those that involve the analytical dual solution z, we arrive at the following result.

COROLLARY 3.3. *Under the assumptions of Theorem* 3.1*, the following a posteriori error bound holds:*

$$\begin{aligned}|J(u)-J(u_{\mathrm{DG}})| &\leq \mathcal{E}(u_{\mathrm{DG}},h,p,\tilde{z}_{\mathrm{DG}}-z_{h,p}) + |\mathcal{E}_\Omega(u_{\mathrm{DG}},h,p,z-\tilde{z}_{\mathrm{DG}})| \\ &\equiv \mathcal{E}_{\mathrm{P}}+\mathcal{E}_{\mathrm{D}}\,,\end{aligned} \tag{3.14}$$

where $\mathcal{E}$ and $\mathcal{E}_\Omega$ are defined in (3.10) *and* (3.6)*, respectively.*

We emphasize here that the key difference between the terms $\mathcal{E}(u_{\mathrm{DG}},h,p,\cdot)$ and $|\mathcal{E}_\Omega(u_{\mathrm{DG}},h,p,\cdot)|$ is that in the former the absolute value signs appear under the summation over the elements $\kappa\in\mathcal{T}$, while in the latter the absolute value sign is outside the sum. It will be shown through numerical experiments that the true weighting function $z-z_{h,p}$ appearing in the a posteriori error bound (3.9) may be accurately approximated by $\tilde{z}_{\mathrm{DG}}-z_{h,p}$; indeed, we shall show that $\mathcal{E}_{\mathrm{D}}$ is typically an order of magnitude smaller than $\mathcal{E}_{\mathrm{P}}$; cf. Table 5.1 and Figures 5.4 and 5.8, below, and [13, 24] in the case of nonlinear hyperbolic conservation laws. Therefore, $\mathcal{E}_{\mathrm{D}}$ can be safely absorbed into $\mathcal{E}_{\mathrm{P}}$ without compromising the reliability of the adaptive algorithm when the stopping criterion (3.8) is replaced by

$$\mathcal{E}_{\mathrm{P}} \leq \mathtt{TOL}\,.$$

By this we mean that the right-hand side of (3.14) remains an upper bound on the true error in the linear functional $J(\cdot)$, even when the term $\mathcal{E}_{\mathrm{D}}$ is neglected in the process of error control for the primal problem.

The *hp*-DGFEM approximation $\tilde{z}_{\mathrm{DG}}$ to the dual solution z appearing in $\mathcal{E}_{\mathrm{P}}$ will be computed by an *hp*-adaptive finite element algorithm, on a sequence of "dual meshes" $\tilde{\mathcal{T}}$, which, in general, differ from the "primal meshes" $\mathcal{T}$. The sequence of dual meshes will be generated by means of an a posteriori error bound on

$$\mathcal{E}_{\mathrm{D}} = \left|\mathcal{E}_\Omega(u_{\mathrm{DG}}, h, p, z) - \mathcal{E}_\Omega(u_{\mathrm{DG}}, h, p, \tilde{z}_{\mathrm{DG}})\right| .$$

Since $\mathcal{E}_\Omega(u_{\mathrm{DG}}, h, p, \cdot)$ is a linear functional, we can derive an a posteriori bound on $\mathcal{E}_{\mathrm{D}}$ by mimicking the process of error estimation for the primal problem described above with the primal problem replaced by the dual and the functional $J(\cdot)$ substituted by $\mathcal{E}_\Omega(u_{\mathrm{DG}}, h, p, \cdot)$. As the use of a Type I a posteriori error bound on $\mathcal{E}_{\mathrm{D}}$ would involve the solution of the dual to the dual problem, which would then in turn have to be solved numerically, we shall instead use a crude Type II a posteriori error bound on $\mathcal{E}_{\mathrm{D}}$ so as to terminate the potentially infinite succession of mutually dual problems that would otherwise arise. The crudeness of the Type II bound on $\mathcal{E}_{\mathrm{D}}$ will be of no concern: as noted above, numerical experiments indicate that $\mathcal{E}_{\mathrm{D}} \ll \mathcal{E}_{\mathrm{P}}$, so from the practical point of view there appears to be little advantage in performing reliable error control for $\mathcal{E}_{\mathrm{D}}$; our aim, when using the Type II bound on $\mathcal{E}_{\mathrm{D}}$, is merely to generate an adequate sequence of finite element approximations $\tilde{z}_{\mathrm{DG}}$ to the dual solution z which we can then use to compute $\mathcal{E}_{\mathrm{P}}$. The next section is, therefore, devoted to Type II a posteriori error bounds for the hp-DGFEM.

3.2. Type II a posteriori error bounds. For convenience, in the rest of this section we shall restrict ourselves to meshes consisting of affine equivalent d-parallelepiped elements. Generalizations to nonaffine elements may be treated by constructing meshes consisting of local patches as in [16].

Assuming that m is a positive integer, we define the negative Sobolev norm $\|\cdot\|_{H^{-m}(\Omega)}$ in the usual way:

$$\|w\|_{H^{-m}(\Omega)} = \sup_{v \in C_0^\infty(\Omega)} \frac{|(w, v)|}{\|v\|_{H^m(\Omega)}} .$$

Further, we write $\|v\|_\tau$, $\tau \subset \partial\kappa$, $\kappa \in \mathcal{T}$, to denote the (semi-) norm induced by the (semi-) inner product

$$\langle v, w\rangle_\tau = \int_\tau |\mathbf{b}\cdot\mathbf{n}|\, vw \,\mathrm{d}s,$$

for $v, w \in L_2(\tau)$. Finally, we recall from [16] and [21] the following approximation results for the finite element space $S^{\mathbf{p}}(\Omega, \mathcal{T}, \mathbf{F})$.

Lemma 3.4. *Suppose that $u|_\kappa \in H^{k_\kappa}(\kappa)$, $k_\kappa \geq 1$, for some κ in $\mathcal{T}$. Then, there exists $\Pi_{hp}u$ in the finite element space $S^{\mathbf{p}}(\Omega, \mathcal{T}, \mathbf{F})$, a constant C_{int} dependent only on d and the shape-regularity of $\mathcal{T}$, but independent of u, $h_\kappa = diam(\kappa)$, p_κ and k_κ, such that*

$$\|u - \Pi_{hp}u\|^2_{L_2(\kappa)} \leq C^2_{\mathrm{int}} h_\kappa^{2s_\kappa} \frac{1}{p_\kappa(p_\kappa+1)}\, \Phi_1(p_\kappa, s_\kappa)\, |u|^2_{H^{s_\kappa}(\kappa)} ,$$

and

$$\|\nabla(u - \Pi_{hp}u)\|^2_{L_2(\kappa)} \leq C^2_{\mathrm{int}} h_\kappa^{2s_\kappa-2}\, \Phi_1(p_\kappa, s_\kappa)\, |u|^2_{H^{s_\kappa}(\kappa)} ,$$

for $1 \leq s_\kappa \leq \min(k_\kappa, p_\kappa+1)$. Here,

$$\Phi_1(p, s) := \frac{\Gamma(p-s+2)}{\Gamma(p+s)} + \frac{1}{p(p+1)}\frac{\Gamma(p-s+3)}{\Gamma(p+s-1)}, \qquad 1 \leq s \leq p+1 .$$

THEOREM 3.5. *Let u and u_{DG} denote the solutions of* (2.1) *and* (2.5)*, respectively. Suppose further that the dual problem* (3.1) *has a unique solution $z \in H(\mathcal{L}^*, \Omega)$ such that $z|_\kappa \in H^{t_\kappa}(\kappa)$ for each $\kappa \in \mathcal{T}$ with $1 \le t_\kappa \le p_\kappa + 1$. Then,*

$$|J(u) - J(u_{\mathrm{DG}})| \le C_0 C_{\mathrm{int}} |z|_{H^{\mathbf{t}}(\Omega,\mathcal{T})} \left[\left(\sum_{\kappa \in \mathcal{T}} \frac{h_\kappa^{2t_\kappa}}{p_\kappa(p_\kappa+1)} \, \Phi_1(p_\kappa, t_\kappa) \, \|r_{h,p}\|^2_{L_2(\kappa)} \right)^{1/2} \right.$$
$$+ \left(\sum_{\kappa \in \mathcal{T}} \frac{h_\kappa^{2t_\kappa - 1}}{p_\kappa} \, \Phi_1(p_\kappa, t_\kappa) \, \|[u_{\mathrm{DG}}]\|^2_{\partial_-\kappa \setminus \Gamma_-} \right)^{1/2}$$
$$\left. + \left(\sum_{\kappa \in \mathcal{T}} \frac{h_\kappa^{2t_\kappa - 1}}{p_\kappa} \, \Phi_1(p_\kappa, t_\kappa) \, \|r_{h,p-}\|^2_{\partial_-\kappa \cap \Gamma_-} \right)^{1/2} \right],$$

where C_0 depends only on $\|\mathbf{b}\|_{L_\infty(\Omega)}$, the dimension d, and the shape-regularity of $\mathcal{T}$.

Proof. Applying the Cauchy–Schwarz inequality to the right-hand side of (3.9) in Corollary 3.2 gives

$$\begin{aligned} |J(u) - J(u_{\mathrm{DG}})| &\le \sum_{\kappa \in \mathcal{T}} \|r_{h,p}\|_{L_2(\kappa)} \|z - z_{h,p}\|_{L_2(\kappa)} \\ &\quad + \sum_{\kappa \in \mathcal{T}} \|[u_{\mathrm{DG}}]\|_{\partial_-\kappa \setminus \Gamma_-} \|(z - z_{h,p})^+\|_{\partial_-\kappa \setminus \Gamma_-} \\ &\quad + \sum_{\kappa \in \mathcal{T}} \|r_{h,p-}\|_{\partial_-\kappa \cap \Gamma_-} \|(z - z_{h,p})^+\|_{\partial_-\kappa \cap \Gamma_-} \\ &\equiv \mathrm{I} + \mathrm{II} + \mathrm{III}\,. \end{aligned} \tag{3.15}$$

Exploiting the approximation result stated in Lemma 3.4, together with the Cauchy–Schwarz inequality, we get

$$\mathrm{I} \le C_{\mathrm{int}} \left(\sum_{\kappa \in \mathcal{T}} \frac{h_\kappa^{2t_\kappa}}{p_\kappa(p_\kappa+1)} \, \Phi_1(p_\kappa, t_\kappa) \, \|r_{h,p}\|^2_{L_2(\kappa)} \right)^{1/2} \left(\sum_{\kappa \in \mathcal{T}} |z|^2_{H^{t_\kappa}(\kappa)} \right)^{1/2}$$

for $1 \le t_\kappa \le p_\kappa + 1$ and κ in $\mathcal{T}$. To bound the approximation error $z - z_{h,p}$ on a subset τ of the boundary of a given element κ in $\mathcal{T}$, we use the following trace inequality:

$$\|v\|^2_\tau \le \frac{1}{2} C_0^2 (\|\nabla v\|_{L_2(\kappa)} \|v\|_{L_2(\kappa)} + h_\kappa^{-1} \|v\|^2_{L_2(\kappa)}); \tag{3.16}$$

here, C_0 depends only on $\|\mathbf{b}\|_{L_\infty(\Omega)}$, d, and the shape-regularity of $\mathcal{T}$.

Exploiting (3.16), together with Lemma 3.4, gives the following bounds on terms II and III:

$$\mathrm{II} \le C_0 C_{\mathrm{int}} \left(\sum_{\kappa \in \mathcal{T}} \frac{h_\kappa^{2t_\kappa - 1}}{p_\kappa} \, \Phi_1(p_\kappa, t_\kappa) \, \|[u_{\mathrm{DG}}]\|^2_{\partial_-\kappa \setminus \Gamma_-} \right)^{1/2} \left(\sum_{\kappa \in \mathcal{T}} |z|^2_{H^{t_\kappa}(\kappa)} \right)^{1/2},$$
$$\mathrm{III} \le C_0 C_{\mathrm{int}} \left(\sum_{\kappa \in \mathcal{T}} \frac{h_\kappa^{2t_\kappa - 1}}{p_\kappa} \, \Phi_1(p_\kappa, t_\kappa) \, \|r_{h,p-}\|^2_{\partial_-\kappa \cap \Gamma_-} \right)^{1/2} \left(\sum_{\kappa \in \mathcal{T}} |z|^2_{H^{t_\kappa}(\kappa)} \right)^{1/2}$$

for $1 \le t_\kappa \le p_\kappa + 1$ and κ in $\mathcal{T}$. Upon inserting the estimates on I, II, and III into (3.15) the result follows. □

REMARK 3.6. *As an application of Theorem* 3.5*, let us consider the problem of a posteriori error estimation for the hp-DGFEM approximation of the weighted mean value of the solution u to the model problem* (2.1)*. Choosing $J(u) = M_\psi(u)$ in Theorem* 3.5 *with the weight function $\psi \in C_0^\infty(\Omega)$ gives*

$$\begin{aligned} |M_\psi(u) - M_\psi(u_{\mathrm{DG}})| &\equiv |(u - u_{\mathrm{DG}}, \psi)| \\ &\le C_0 C_{\mathrm{int}} \left[\left(\sum_{\kappa\in\mathcal{T}} \frac{h_\kappa^{2t_\kappa}}{p_\kappa(p_\kappa+1)} \, \Phi_1(p_\kappa, t_\kappa) \, \|r_{h,p}\|^2_{L_2(\kappa)} \right)^{1/2} \right. \\ &\quad + \left(\sum_{\kappa\in\mathcal{T}} \frac{h_\kappa^{2t_\kappa-1}}{p_\kappa} \, \Phi_1(p_\kappa, t_\kappa) \, \|[u_{\mathrm{DG}}]\|^2_{\partial_-\kappa\setminus\Gamma_-} \right)^{1/2} \\ &\quad \left. + \left(\sum_{\kappa\in\mathcal{T}} \frac{h_\kappa^{2t_\kappa-1}}{p_\kappa} \, \Phi_1(p_\kappa, t_\kappa) \, \|r_{h,p-}\|^2_{\partial_-\kappa\cap\Gamma_-} \right)^{1/2} \right] |z|_{H^{\mathbf{t}}(\Omega)} . \end{aligned} \tag{3.17}$$

Letting $m = \max_{\kappa\in\mathcal{T}} t_\kappa$, it follows by the differentiability theorem of Rauch [20] *applied to* (3.3) *that*

$$\|z\|_{H^m(\Omega)} \le C_{\mathrm{stab}} \|\psi\|_{H^m(\Omega)} , \tag{3.18}$$

where C_{stab} is a positive constant, independent of ψ, called the stability factor of the dual problem. Denoting by $\mathcal{B}(h, p, u_{\mathrm{DG}})$ the expression in the square bracket in (3.17)*, we deduce the following Type* II *a posteriori error bound:*

$$|M_\psi(u) - M_\psi(u_{\mathrm{DG}})| \le C_0 C_{\mathrm{int}} C_{\mathrm{stab}} \|\psi\|_{H^m(\Omega)} \mathcal{B}(h, p, u_{\mathrm{DG}}) .$$

In fact, upon dividing both sides of the last inequality by $\|\psi\|_{H^m(\Omega)}$ and taking the supremum over all $\psi \in C_0^\infty(\Omega)$) yields the following Type II *a posteriori error bound in a negative Sobolev norm:*

$$\|u - u_{\mathrm{DG}}\|_{H^{-m}(\Omega)} \le C_0 C_{\mathrm{int}} C_{\mathrm{stab}} \mathcal{B}(h, p, u_{\mathrm{DG}}) .$$

We note that a bound identical to (3.17) *holds for the weighted normal flux $N_\psi(u)$ of u through the outflow boundary Γ_+ with ψ a sufficiently smooth weight function defined on Γ_+.*

3.3. A priori error bounds. As indicated in the introduction, the *hp*-adaptive algorithms for the primal and dual problems will be driven by a posteriori error bounds: a Type I bound for the primal problem and a Type II bound for the dual problem. The decision about whether a local h-refinement/derefinement or a local p-refinement/derefinement is to be performed in the course of mesh adaptation will be based on assessing the local regularity of the primal and the dual solution. The estimation of local Sobolev indices which measure local regularity will, in turn, rely on an a priori bound on the error in the computed functional in terms of Sobolev norms of the analytical primal and dual solutions u and z, respectively; this will indicate the expected rate of convergence for $|J(u) - J(u_{\mathrm{DG}})|$ as the finite element space is enriched, i.e., as h tends to 0 and p tends to infinity, assuming that the primal and dual solutions have certain Sobolev regularities. The present section is devoted to the derivation of this a priori error bound.

We assume for the moment that the data for the problem (2.1) satisfy the following assumptions:

$$\begin{aligned} \mathbf{b}\cdot\nabla_{\mathcal{T}} v_h \in S^{\mathbf{p}}(\Omega,\mathcal{T},\mathbf{F}) \quad &\forall v_h \in S^{\mathbf{p}}(\Omega,\mathcal{T},\mathbf{F})\,, \\ c \in S^{\mathbf{0}}(\Omega,\mathcal{T},\mathbf{F})\,, \qquad &f \in S^{\mathbf{p}}(\Omega,\mathcal{T},\mathbf{F})\,. \end{aligned} \tag{3.19}$$

Here, $\nabla_{\mathcal{T}} v$, $v \in H^1(\Omega,\mathcal{T})$, denotes the broken gradient of v defined by $(\nabla_{\mathcal{T}} v)|_\kappa = \nabla(v|_\kappa)$, $\kappa \in \mathcal{T}$. To ensure that (2.1) is then meaningful (i.e., that the characteristic curves of the differential operator $\mathcal{L}$ are correctly defined), we still assume that $\mathbf{b} \in \left[W^1_\infty(\Omega)\right]^d$.

REMARK 3.7. *The condition placed on* $\mathbf{b}$ *is required in the proof of the a priori error bound stated below (Lemma* 3.9*), cf.* [15]*; the conditions on the reaction/absorption term* c *and the forcing function* f *are required to ensure that the internal residual* $r_{h,p}$ *belongs to the finite element space* $S^{\mathbf{p}}(\Omega,\mathcal{T},\mathbf{F})$*, cf. Lemma* 3.10*. However, we shall see in section* 5 *that the restrictions* (3.19) *are not essential in practice for our a priori error bounds to hold.*

Before embarking on the a priori error analysis, we quote from Schwab [21, Theorem 4.76, p. 208] the following inverse inequality for the finite element space $S^{\mathbf{p}}(\Omega,\mathcal{T},\mathbf{F})$.

LEMMA 3.8. *Given* v *in* $S^{\mathbf{p}}(\Omega,\mathcal{T},\mathbf{F})$*, there exists a positive constant* C*, dependent only on* d *and the shape-regularity of* $\mathcal{T}$*, such that*

$$|v|_{H^1(\kappa)} \le C\,\frac{p_\kappa^2}{h_\kappa}\,\|v\|_{L_2(\kappa)}$$

for all κ *in* $\mathcal{T}$*.*

In the following lemma (cf. [15]) we formulate an a priori bound on the error $u - u_{\mathrm{DG}}$ in terms of the DG–norm $|||\cdot|||_{\mathrm{DG}}$ defined in section 2.2. We recall that

$$\begin{aligned} |||v|||^2_{\mathrm{DG}} := \sum_{\kappa\in\mathcal{T}} \Big\{ &\|c_0 v\|^2_{L_2(\kappa)} + \frac{1}{2}\|v^+\|_{\partial_-\kappa\cap\Gamma_-} + \frac{1}{2}\|v^+\|^2_{\partial_+\kappa\cap\Gamma_+} \\ &+ \frac{1}{2}\|v^+ - v^-\|^2_{\partial_-\kappa\setminus\Gamma_-} \Big\}\,, \end{aligned} \tag{3.20}$$

where c_0 is the (positive) function defined in (2.3).

LEMMA 3.9. *Let* u *and* u_{DG} *denote the solutions of* (2.1) *and* (2.5)*, respectively. Assuming that* (3.19) *holds and* $u|_\kappa \in H^{k_\kappa}(\kappa)$*,* $k_\kappa \ge 1$*, for all* κ *in* $\mathcal{T}$*, we have*

$$|||u - u_{\mathrm{DG}}|||^2_{\mathrm{DG}} \le C \sum_{\kappa\in\mathcal{T}} h_\kappa^{2s_\kappa - 1}\,\Phi_2(p_\kappa, s_\kappa)\,|u|^2_{H^{s_\kappa}(\kappa)}$$

for any integers $1 \le s_\kappa \le \min(p_\kappa + 1, k_\kappa)$*,* $\kappa \in \mathcal{T}$*. Here,*

$$\begin{aligned} \Phi_2(s,p) = \frac{1}{2p+1}\,&\left(\frac{\Gamma(p+2-s)}{\Gamma(p+s)} + \frac{\Gamma(p+3-s)}{\Gamma(p+1+s)}\right) \\ &+ \left(\frac{\Gamma(p+2-s)}{\Gamma(p+2+s)}\right)^{\frac{1}{2}} \left(\frac{\Gamma(p+3-s)}{\Gamma(p+1+s)}\right)^{\frac{1}{2}} + \frac{\Gamma(p+2-s)}{\Gamma(p+2+s)}\,, \end{aligned}$$

and C is a positive constant depending only on d, $\|\mathbf{b}\|_{L_\infty(\Omega)}$, $\|c\|_{L_\infty(\Omega)}$, and the shape-regularity of $\mathcal{T}$.

Furthermore, we need the following bound on the internal residual $r_{h,p}$.

LEMMA 3.10. *Assuming that the conditions on the data* (3.19) *hold, there exists a positive constant C, dependent only on d and the shape-regularity of $\mathcal{T}$, such that*

$$\sum_{\kappa\in\mathcal{T}} \frac{h_\kappa}{p_\kappa^2+1}\|r_{h,p}\|^2_{L_2(\kappa)} \leq C|||u-u_{\mathrm{DG}}|||^2_{\mathrm{DG}}\,.$$

Proof. From the Galerkin orthogonality property (3.5) we have

$$\sum_{\kappa\in\mathcal{T}}(r_{h,p},v)_\kappa = -\sum_{\kappa\in\mathcal{T}}((\mathbf{b}\cdot\mathbf{n})[u_{\mathrm{DG}}],v^+)_{\partial_-\kappa\backslash\Gamma_-} + \sum_{\kappa\in\mathcal{T}}((\mathbf{b}\cdot\mathbf{n})r_{h,p-},v^+)_{\partial_-\kappa\cap\Gamma_-} \tag{3.21}$$

for any v in $S^{\mathbf{p}}(\Omega,\mathcal{T},\mathbf{F})$. Under the conditions (3.19), the internal residual $r_{h,p}$ belongs to the finite element space $S^{\mathbf{p}}(\Omega,\mathcal{T},\mathbf{F})$; thereby, choosing $v=\theta r_{h,p}$ in (3.21), where θ is in $S^{\mathbf{0}}(\Omega,\mathcal{T},\mathbf{F})$ such that $\theta|_\kappa=\theta_\kappa>0$, and applying the Cauchy–Schwarz inequality gives

$$\begin{aligned}\sum_{\kappa\in\mathcal{T}}\theta_\kappa\|r_{h,p}\|^2_{L_2(\kappa)} \leq{}& \sum_{\kappa\in\mathcal{T}}\theta_\kappa\|[u_{\mathrm{DG}}]\|_{\partial_-\kappa\backslash\Gamma_-}\|r^+_{h,p}\|_{\partial_-\kappa\backslash\Gamma_-}\\ &+\sum_{\kappa\in\mathcal{T}}\theta_\kappa\|r_{h,p-}\|_{\partial_-\kappa\cap\Gamma_-}\|r^+_{h,p}\|_{\partial_-\kappa\cap\Gamma_-}\,.\end{aligned}$$

Exploiting the trace inequality (3.16), together with the inverse inequality stated in Lemma 3.8, we deduce that

$$\sum_{\kappa\in\mathcal{T}}\theta_\kappa\|r_{h,p}\|^2_{L_2(\kappa)} \leq C\sum_{\kappa\in\mathcal{T}}\theta_\kappa\frac{p_\kappa^2+1}{h_\kappa}\left(\|[u_{\mathrm{DG}}]\|^2_{\partial_-\kappa\backslash\Gamma_-}+\frac{1}{2}\|r_{h,p-}\|^2_{\partial_-\kappa\cap\Gamma_-}\right).$$

Choosing $\theta_\kappa = h_\kappa/(p_\kappa^2+1)$ together with the definition of the DG-norm (3.20) gives the desired result. □

Equipped with Lemmas 3.9 and 3.10, together with the approximation results stated in Lemma 3.4, we are now in a position to prove the following hp-bound on the error in the computed functional $J(\cdot)$.

THEOREM 3.11. *Let u and u_{DG} denote the solutions of* (2.1) *and* (2.5), *respectively. Given that $u|_\kappa\in H^{k_\kappa}(\kappa)$, $k_\kappa\geq 1$, and $z|_\kappa\in H^{l_\kappa}(\kappa)$, $l_\kappa\geq 1$, for all κ in $\mathcal{T}$, we have*

$$\begin{aligned}|J(u)-J(u_{\mathrm{DG}})|^2 \leq C&\sum_{\kappa\in\mathcal{T}}h_\kappa^{2s_\kappa-1}\,\Phi_2(p_\kappa,s_\kappa)\,|u|^2_{H^{s_\kappa}(\kappa)}\\ &\times\sum_{\kappa\in\mathcal{T}}h_\kappa^{2t_\kappa-1}\,\Phi_1(p_\kappa,t_\kappa)\,|z|^2_{H^{t_\kappa}(\kappa)}\end{aligned} \tag{3.22}$$

for any $1\leq s_\kappa\leq\min(p_\kappa+1,k_\kappa)$, $1\leq t_\kappa\leq\min(p_\kappa+1,l_\kappa)$, $\kappa\in\mathcal{T}$. Here, C is a positive constant depending only on d, $\|\mathbf{b}\|_{L_\infty(\Omega)}$, $\|c\|_{L_\infty(\Omega)}$, and the shape-regularity of $\mathcal{T}$.

Proof. With terms I, II, and III defined as in the proof of Theorem 3.5 (cf. (3.15)), we deduce from Lemma 3.4 and (3.16) the following bounds:

$$\mathrm{I} \le C \left(\sum_{\kappa \in \mathcal{T}} \frac{h_\kappa}{p_\kappa^2 + 1} \, \|r_{h,p}\|^2_{L_2(\kappa)} \right)^{1/2}$$

$$\times \left(\sum_{\kappa \in \mathcal{T}} h_\kappa^{2t_\kappa - 1} \, \frac{p_\kappa^2 + 1}{p_\kappa(p_\kappa + 1)} \, \Phi_1(p_\kappa, t_\kappa) \, |z|^2_{H^{t_\kappa}(\kappa)} \right)^{1/2} , \tag{3.23}$$

$$\mathrm{II} \le C \left(\sum_{\kappa \in \mathcal{T}} \|[u_{\mathrm{DG}}]\|^2_{\partial_-\kappa \setminus \Gamma_-} \right)^{1/2} \left(\sum_{\kappa \in \mathcal{T}} \frac{h_\kappa^{2t_\kappa - 1}}{p_\kappa} \Phi_1(p_\kappa, t_\kappa) |z|^2_{H^{t_\kappa}(\kappa)} \right)^{1/2} , \tag{3.24}$$

$$\mathrm{III} \le C \left(\sum_{\kappa \in \mathcal{T}} \|r_{h,p-}\|^2_{\partial_-\kappa \cap \Gamma_-} \right)^{1/2} \left(\sum_{\kappa \in \mathcal{T}} \frac{h_\kappa^{2t_\kappa - 1}}{p_\kappa} \Phi_1(p_\kappa, t_\kappa) |z|^2_{H^{t_\kappa}(\kappa)} \right)^{1/2} , \tag{3.25}$$

respectively. Collecting the bounds (3.23), (3.24), and (3.25), and exploiting Lemma 3.10, we get

$$|J(u) - J(u_{\mathrm{DG}})|^2 \le C |\!|\!| u - u_{\mathrm{DG}} |\!|\!|^2_{\mathrm{DG}} \sum_{\kappa \in \mathcal{T}} h_\kappa^{2t_\kappa - 1} \Phi_1(p_\kappa, t_\kappa) |z|^2_{H^{t_\kappa}(\kappa)} .$$

Finally, employing Lemma 3.9 gives the desired result. □

Let us now discuss some special cases of the general error bound derived in Theorem 3.11. We first note that, for fixed s, Stirling's formula implies

$$\Phi_1(p, s) \le C(s) p^{-2s+2} , \qquad \Phi_2(p, s) \le C(s) p^{-2s+1} , \tag{3.26}$$

as $p \to \infty$. Thereby, for uniform orders $p_\kappa = p$, $s_\kappa = s$, $t_\kappa = t$, $k_\kappa = k$, $l_\kappa = l$, s, t, k and l integers, and $h_\kappa = h$ for all κ in $\mathcal{T}$, we get the bound

$$|J(u) - J(u_{\mathrm{DG}})| \le C \left(\frac{h}{p} \right)^{s+t-1} p^{1/2} \, |u|_{H^s(\Omega)} |z|_{H^t(\Omega)} , \tag{3.27}$$

where $1 \le s \le \min(p+1, k)$ and $1 \le t \le \min(p+1, l)$. Hence, we deduce that

$$|J(u) - J(u_{\mathrm{DG}})| \le C \frac{h^{s+t-1}}{p^{k+l-1}} p^{1/2} \, \|u\|_{H^k(\Omega)} \|z\|_{H^l(\Omega)} , \tag{3.28}$$

where $1 \le s \le \min(p+1, k)$ and $1 \le t \le \min(p+1, l)$; cf. [23]. We note that in the transition from (3.27) to (3.28) the generic constant C is increased by the factor $(k-1)^{k-1}(l-1)^{l-1}$. Here, the bounds (3.27) and (3.28) are optimal in h and suboptimal in p by $p^{1/2}$; in the case of fixed p, (3.27), (3.28) reduce to the optimal h-convergence error bound proved in [14] for a stabilized continuous approximation to u. From (3.28) we may deduce the following a priori error bound:

$$\|u - u_{\mathrm{DG}}\|_{H^{-m}(\Omega)} \le C \frac{h^{s+\theta-1}}{p^{k+m-1}} p^{1/2} \, \|u\|_{H^k(\Omega)}, \tag{3.29}$$

where $1 \le s \le \min(p+1, k)$ and $1 \le \theta \le \min(p+1, m)$. In the presence of streamline-diffusion stabilization, with stabilization of size $\delta = h/p$, the bounds (3.27), (3.28), and (3.29) can be sharpened to ones that are simultaneously optimal in both h and p.

The explicit dependence of the error bound stated in Theorem 3.11 on the local regularity of the primal and dual solutions u and z, respectively, allows us to deduce that the error in the functional $J(\cdot)$ is exponentially convergent as $p_\kappa \to \infty$ for each κ in $\mathcal{T}$. To this end, let us assume that z is elementwise analytic in the sense that, for each $\kappa \in \mathcal{T}$, $z|_\kappa$ has analytic extension to an open set, independent of h_κ, containing $\bar{\kappa}$. Then,

$$\forall \kappa \in \mathcal{T} \quad \exists d_\kappa > 1 \quad \exists C(z) > 0 \quad \forall s > 0 : \ |z|_{H^s(\kappa)} \le C(z)(d_\kappa)^s s!\, [\mathrm{meas}(\kappa)]^{1/2} .$$

In order to emphasize the dependence of d_κ on the particular function z under consideration, we write $d_\kappa(z)$ in lieu of d_κ. Thereby, assuming that z is elementwise analytic and $h_\kappa > 0$ is fixed for all κ in $\mathcal{T}$, on setting $t_\kappa = \alpha_\kappa p_\kappa + 1$, where $0 < \alpha_\kappa = (1 + (d_\kappa(z))^2)^{-1/2} < 1$ for all $\kappa \in \mathcal{T}$, it can be shown that

$$\sum_{\kappa \in \mathcal{T}} h_\kappa^{2t_\kappa - 1}\, \Phi_1(p_\kappa, t_\kappa)\, |z|^2_{H^{t_\kappa}(\kappa)} \le C(z) \sum_{\kappa \in \mathcal{T}} h_\kappa^{2t_\kappa - 1} p_\kappa^3\, \mathrm{e}^{-2\lambda_\kappa p_\kappa} \mathrm{meas}(\kappa) , \tag{3.30}$$

where λ_κ is a positive constant on each element κ in the mesh $\mathcal{T}$; namely,

$$\lambda_\kappa = \frac{1}{2} |\log F(\alpha_\kappa, d_\kappa(z))| , \qquad \text{where} \qquad F(\alpha, d) = \frac{(1-\alpha)^{1-\alpha}}{(1+\alpha)^{1+\alpha}}\, (\alpha d)^{2\alpha};$$

see [16] for details. Similarly, assuming that u is elementwise analytic, setting $s_\kappa = \beta_\kappa p_\kappa + 1$, where $0 < \beta_\kappa = (1 + (d_\kappa(u))^2)^{-1/2} < 1$ for all $\kappa \in \mathcal{T}$, we have that

$$\sum_{\kappa \in \mathcal{T}} h_\kappa^{2s_\kappa - 1}\, \Phi_2(p_\kappa, s_\kappa)\, |u|^2_{H^{s_\kappa}(\kappa)} \le C(u) \sum_{\kappa \in \mathcal{T}} h_\kappa^{2s_\kappa - 1} p_\kappa^2\, \mathrm{e}^{-2\mu_\kappa p_\kappa} \mathrm{meas}(\kappa) , \tag{3.31}$$

where $\mu_\kappa = (1/2)|\log F(\beta_\kappa, d_\kappa(u))|$. Thereby, combining (3.30) and (3.31), we deduce the exponential convergence estimate

$$\begin{aligned} |J(u) - J(u_{\mathrm{DG}})|^2 \le K &\sum_{\kappa \in \mathcal{T}} h_\kappa^{2s_\kappa - 1} p_\kappa^2\, \mathrm{e}^{-2\mu_\kappa p_\kappa} \mathrm{meas}(\kappa) \\ &\times \sum_{\kappa \in \mathcal{T}} h_\kappa^{2t_\kappa - 1} p_\kappa^3\, \mathrm{e}^{-2\lambda_\kappa p_\kappa} \mathrm{meas}(\kappa) , \end{aligned} \tag{3.32}$$

where $K = C(u)C(z)$.

REMARK 3.12. *The bound* (3.32) *indicates that in order to ensure that the error in the functional $J(\cdot)$ decays exponentially as the degree of the approximating polynomial is increased, it is only necessary to assume that either u or z is elementwise analytic; this will be demonstrated numerically in section* 5.

4. Implementational issues.

4.1. Numerical approximation of the dual solution. In this section we formulate the discontinuous Galerkin finite element approximation of the dual problem (3.1). As stated in section 3, the particular form of the dual problem is dependent on the functional under consideration. For generality, let us suppose that z is the (unique) solution to the following problem: find $z \in H(\mathcal{L}^*, \Omega)$ such that

$$\begin{aligned} \mathcal{L}^* z \equiv -\nabla \cdot (\mathbf{b} z) + cz = \varphi, &\quad x \in \Omega , \\ z = \chi, &\quad x \in \Gamma_+ . \end{aligned} \tag{4.1}$$

Clearly, (4.1) covers both the case when the functional $J(\cdot)$ under consideration represents the local mean value of the solution u and when $J(\cdot)$ is the outflow normal flux of u; cf. section 3.

As in section 2.1, we define $\tilde{S}^{\tilde{\mathbf{p}}}(\Omega, \tilde{\mathcal{T}}, \tilde{\mathbf{F}})$ to be the finite element space consisting of piecewise polynomials of degree $\tilde{\mathbf{p}}|_{\tilde{\kappa}} = \tilde{p}_{\tilde{\kappa}}$ on a mesh $\tilde{\mathcal{T}}$ consisting of shape-regular elements $\tilde{\kappa}$ of size $\tilde{h}_{\tilde{\kappa}}$. With $\partial_+ \tilde{\kappa}$ defined as in (2.4), we introduce the bilinear form

$$\begin{aligned}\tilde{B}_{\mathrm{DG}}(w, v) = \sum_{\tilde{\kappa} \in \tilde{\mathcal{T}}} \int_{\tilde{\kappa}} \mathcal{L}^* w \, v \, \mathrm{d}x + \sum_{\tilde{\kappa} \in \tilde{\mathcal{T}}} \int_{\partial_+ \tilde{\kappa} \setminus \Gamma_+} (\mathbf{b} \cdot \mathbf{n}) [w] \, v^+ \, \mathrm{d}s \\ + \sum_{\tilde{\kappa} \in \tilde{\mathcal{T}}} \int_{\partial_+ \tilde{\kappa} \cap \Gamma_+} (\mathbf{b} \cdot \mathbf{n}) w^+ \, v^+ \, \mathrm{d}s\end{aligned}$$

and linear functional

$$\tilde{\ell}_{\mathrm{DG}}(v) = \sum_{\tilde{\kappa} \in \tilde{\mathcal{T}}} \int_{\tilde{\kappa}} \varphi v \, \mathrm{d}x + \sum_{\tilde{\kappa} \in \tilde{\mathcal{T}}} \int_{\partial_+ \tilde{\kappa} \cap \Gamma_+} (\mathbf{b} \cdot \mathbf{n}) \, \chi v^+ \, \mathrm{d}s$$

associated with the *hp*-DGFEM approximation of the dual problem (4.1). Now the *hp*-DGFEM for (4.1) is defined as follows: find $\tilde{z}_{\mathrm{DG}} \in \tilde{S}^{\tilde{\mathbf{p}}}(\Omega, \tilde{\mathcal{T}}, \tilde{\mathbf{F}})$ such that

$$\tilde{B}_{\mathrm{DG}}(\tilde{z}_{\mathrm{DG}}, v) = \tilde{\ell}_{\mathrm{DG}}(v) \quad \forall v \in \tilde{S}^{\tilde{\mathbf{p}}}(\Omega, \tilde{\mathcal{T}}, \tilde{\mathbf{F}}) \, . \tag{4.2}$$

4.2. Adaptive algorithm. For a user-defined tolerance TOL, we now consider the problem of designing the *hp*-finite element space $S^{\mathbf{p}}(\Omega, \mathcal{T}, \mathbf{F})$ such that

$$|J(u) - J(u_{\mathrm{DG}})| \le \mathtt{TOL} \, , \tag{4.3}$$

subject to the constraint that the total number of degrees of freedom in $S^{\mathbf{p}}(\Omega, \mathcal{T}, \mathbf{F})$ is minimized. Following the discussion presented in section 3, we exploit the a posteriori error bound (3.14) to construct $S^{\mathbf{p}}(\Omega, \mathcal{T}, \mathbf{F})$ such that

$$\mathcal{E}_{\mathrm{P}} \le \mathtt{TOL} \, . \tag{4.4}$$

The stopping criterion (4.4) is enforced by equidistributing $\mathcal{E}_{\mathrm{P}}|_{\kappa} \equiv \tilde{\eta}_{\kappa}$ over the elements κ in the primal mesh $\mathcal{T}$, where $\tilde{\eta}_{\kappa}$ is defined in a similar manner to η_{κ} with z replaced by $\tilde{z}_{\mathrm{DG}}$ in (3.11). Thus, we insist that

$$\tilde{\eta}_{\kappa} \approx \frac{\mathtt{TOL}}{N} \tag{4.5}$$

holds for each κ in $\mathcal{T}$; here, N denotes the number of elements in the mesh $\mathcal{T}$.

Thereby, each of the elements in the primal mesh is flagged for either refinement or derefinement to ensure that the equidistribution principle (4.5) holds. Once an element κ has been flagged a decision must be made whether the local mesh size h_{κ} or the local degree p_{κ} of the approximating polynomial should be adjusted accordingly. Let us first deal with refinement, i.e., when the local error estimator $\tilde{\eta}_{\kappa}$ is larger than the "localized-tolerance" $\mathtt{TOL}/N$. Clearly, if the primal or dual solutions u and z, respectively, are locally "smooth," then p-enrichment will be more effective than h-refinement, since the error will be expected to decay quickly within the current element κ as p_{κ} is increased. However, if u and z have low regularity within the element κ, then h-refinement will be performed. Thus, regions in the computational domain where the primal and dual solution are locally nonsmooth are isolated from smooth regions, thereby reducing the influence of singularities/discontinuities as well as making p-enrichment more effective.

To ensure that the desired level of accuracy is achieved efficiently, an automatic procedure for deciding when to h- or p-refine must be implemented. To this end, we first compute the local error indicator $\tilde{\eta}_\kappa$ on each element κ in the mesh $\mathcal{T}$ using both a p_κ and a $p_\kappa - 1$ representation for u_{DG}; we denote the corresponding values of $\tilde{\eta}_\kappa$ by $\tilde{\eta}_\kappa(p_\kappa)$ and $\tilde{\eta}_\kappa(p_\kappa - 1)$, respectively. Assuming that $\tilde{\eta}_\kappa(p_\kappa - 1) \neq 0$, the perceived smoothness of the primal and dual solutions may be estimated using the ratio

$$\rho_\kappa = \tilde{\eta}_\kappa(p_\kappa)/\tilde{\eta}_\kappa(p_\kappa - 1); \tag{4.6}$$

cf. Adjerid, Aiffa, and Flaherty [1] and Gui and Babuška [12], for example. If $\rho_\kappa \leq \gamma$, $0 < \gamma < 1$, the error is decreasing as the polynomial degree is increased, indicating that p-enrichment should be performed. On the other hand, $\rho_\kappa > \gamma$ means that the element κ should be locally subdivided. The number γ is referred to as the *type-parameter* [12]. Clearly, the choice of γ is critical to the success of this algorithm and will depend on the asymptotic behavior of the quantity of interest. Instead of assigning an *ad hoc* value to the type parameter γ, we use ρ_κ, together with the a priori error bound (3.22), to directly estimate the local regularities k_κ and l_κ of the primal and dual solutions, respectively, on each element κ in $\mathcal{T}$. More precisely, motivated by (3.28), we assume that on a given element κ in $\mathcal{T}$

$$\tilde{\eta}_\kappa = \mathcal{E}_{\mathrm{P}}|_\kappa \approx C_\kappa\, p_\kappa^{-k_\kappa - l_\kappa + 1}.$$

Thus, we have that

$$k_\kappa + l_\kappa = \log(\rho_\kappa)/\log((p_\kappa - 1)/p_\kappa) + 1.$$

Ideally, we would like to know k_κ and l_κ individually. To do so, we compute an estimate of l_κ. The dual regularity l_κ may be estimated by calculating the $L^2(\kappa)$ norm of the error between the projection of $\tilde{z}_{\mathrm{DG}}$ in $\tilde{S}^{\tilde{\mathbf{p}}}(\Omega, \tilde{\mathcal{T}}, \tilde{\mathbf{F}})$ onto the finite element spaces $S^{\mathbf{p}}(\Omega, \mathcal{T}, \mathbf{F})$ and $S^{\mathbf{p-1}}(\Omega, \mathcal{T}, \mathbf{F})$, together with the approximation results derived in [3]. Once both k_κ and l_κ have been determined on element κ, then κ is p-enriched if either k_κ or l_κ is larger than $p_\kappa + 1$; otherwise, the element is subdivided. For computational simplicity, only one hanging node is allowed on each side of a given element κ; additionally, we restrict the variation in the polynomial degree vector $\mathbf{p}$ to be at most one between neighboring elements. We note that this approach has been developed by Ainsworth and Senior [2] in the context of norm control for second-order elliptic problems.

On the other hand, if an element has been flagged for derefinement, then the strategy implemented here is to coarsen the mesh in low-error regions where either the primal or dual solutions u and z, respectively, are smooth and decrease the degree of the approximating polynomial in low-error regions when both u and z are not sufficiently regular; cf. [1]. To this end, we again compute the local regularities k_κ and l_κ of the primal and dual solutions, respectively, on each element κ in $\mathcal{T}$ as described above. The element κ is then coarsened if either k_κ or l_κ is larger than $p_\kappa + 1$; otherwise, the degree p_κ is reduced by one.

The finite element space $\tilde{S}^{\tilde{\mathbf{p}}}(\Omega, \tilde{\mathcal{T}}, \tilde{\mathbf{F}})$ and the finite element approximation $\tilde{z}_{\mathrm{DG}} \in \tilde{S}^{\tilde{\mathbf{p}}}(\Omega, \tilde{\mathcal{T}}, \tilde{\mathbf{F}})$ of the dual solution z will be constructed adaptively at the same time as $S^{\mathbf{p}}(\Omega, \mathcal{T}, \mathbf{F})$. To this end, we note that $\mathcal{E}_\Omega(u_{\mathrm{DG}}, h, p, \cdot)$ is a linear functional on

$H(\mathcal{L}^*, \Omega)$, and we define the error indicator

$$\eta_{\tilde\kappa} = \frac{\tilde h_{\tilde\kappa}}{\tilde p_{\tilde\kappa}} \|\varphi - \mathcal{L}^* \tilde z_{\mathrm{DG}}\|_{L_2(\tilde\kappa)} + \left(\frac{\tilde h_{\tilde\kappa}}{\tilde p_{\tilde\kappa}}\right)^{1/2} \left(\|[\tilde z_{\mathrm{DG}}]\|_{\partial_+\tilde\kappa\backslash\Gamma_+} + \|\chi - \tilde z_{\mathrm{DG}}\|_{\partial_+\tilde\kappa\cap\Gamma_+}\right) \tag{4.7}$$

for

$$\mathcal{E}_{\mathrm{D}} \equiv |\mathcal{E}_\Omega(u_{\mathrm{DG}}, h, p, z) - \mathcal{E}_\Omega(u_{\mathrm{DG}}, h, p, \tilde z_{\mathrm{DG}})|.$$

The error indicator (4.7) arises from a Type II a posteriori error bound on $\mathcal{E}_{\mathrm{D}}$ upon setting all constants in the bound (such as the stability factor C_{stab} of the dual-dual problem and the interpolation constant C_{int}) to unity; cf. Theorem 3.5 and the subsequent Remark 3.6 with $m = 1$, and note that $\Phi_1(p_{\tilde\kappa}, 1)$ is $\mathcal{O}(1)$ by (3.26). The *hp*-adaptive algorithm for the dual problem will be based on the *fixed fraction strategy* outlined in [19]. Consequently, the absolute size of $\eta_{\tilde\kappa}(\tilde p_{\tilde\kappa})$ is insignificant: only the relative sizes of these quantities matter; in particular, this justifies setting all constants to unity in the dual error indicator (4.7). Once the elements have been flagged for refinement/derefinement, $\tilde h_{\tilde\kappa}$ and $\tilde p_{\tilde\kappa}$ are altered accordingly by estimating the local regularity $\tilde l_{\tilde\kappa}$ of the dual solution on the mesh $\tilde{\mathcal{T}}$ as above by calculating $\eta_{\tilde\kappa}$ using a $\tilde p_{\tilde\kappa}$ and $\tilde p_{\tilde\kappa} - 1$ representation of $\tilde z_{\mathrm{DG}}$, together with the a priori bound (3.29).

For related work concerning the design of automatic *hp*-adaptive mesh refinement algorithms, we refer, for example, to the articles by Bey, Oden, and Patra [7], Mavripilis [17], and Rachowicz, Oden, and Demkowicz [18].

5. Numerical experiments. In this section we present a number of experiments to numerically verify the a priori error bounds derived in section 3, as well as to demonstrate the performance of the *hp*-adaptive algorithm outlined in section 4.2.

5.1. Example 1. In this example we let $\Omega = (-1, 1)^2$, $\mathbf{b} = (2 - y^2, 2 - x)$, $c = 1 + (1 + x)(1 + y)^2$, and f is chosen so that the analytical solution to (2.1) is

$$u(x, y) = 1 + \sin(\pi(1 + x)(1 + y)^2/8); \tag{5.1}$$

cf. [15]. Furthermore, we choose the functional of interest $J(\cdot)$ to represent the mean flow of u over Ω, i.e., $J(\cdot) \equiv M_\psi(\cdot)$, where $M_\psi(\cdot)$ is given by (3.2); here, we define the weight function ψ so that the solution of the corresponding dual problem (3.3) is

$$z = 4\sin(\pi(1 + x)/2)\sin(\pi(1 + y)/2)\,\mathrm{e}^{-(2+x+y)^2/2}.$$

Thus, the true value of the mean flow of u over Ω is $M_\psi(u) = 3.9381$.

Here, we investigate the asymptotic behavior of the *hp*-DGFEM on a sequence of successively finer square and quadrilateral meshes for different p. In each case the quadrilateral mesh is constructed from a uniform $N \times N$ square mesh by randomly perturbing each of the *interior* nodes by up to 10% of the local mesh size; cf. [16].

In Figure 5.1 we present a comparison of the error in the functional $|J(u) - J(u_{\mathrm{DG}})|$ with the mesh size h for $p = 1, 2, 3$. Here, we observe that $|J(u) - J(u_{\mathrm{DG}})|$ converges to zero at the rate $\mathcal{O}(h^{2p+1})$ as the mesh is refined for each fixed p, thereby confirming Theorem 3.11 in the case when the assumptions on the data (3.19) are violated; cf. Remark 3.7. Finally, we investigate the convergence of the *hp*-DGFEM with p-enrichment for fixed h. Since the true solution (5.1) is a (real) analytic function, we expect to observe exponential rates of convergence; cf. section 3. Indeed, Figure 5.2 clearly illustrates this behavior: on the linear-log scale, the convergence plots for each

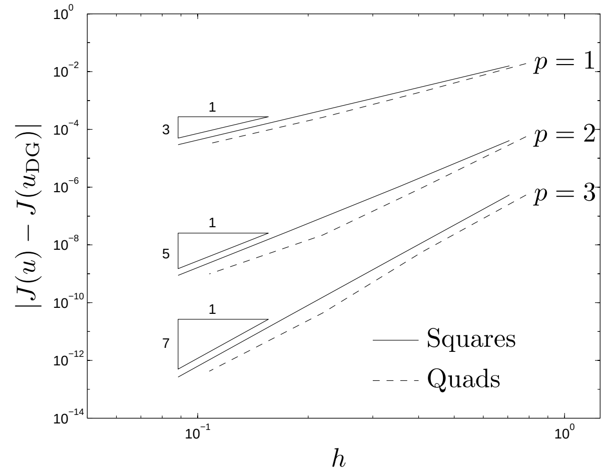


FIG. 5.1. *Example* 1. *Convergence of the hp-DGFEM with h-refinement.*

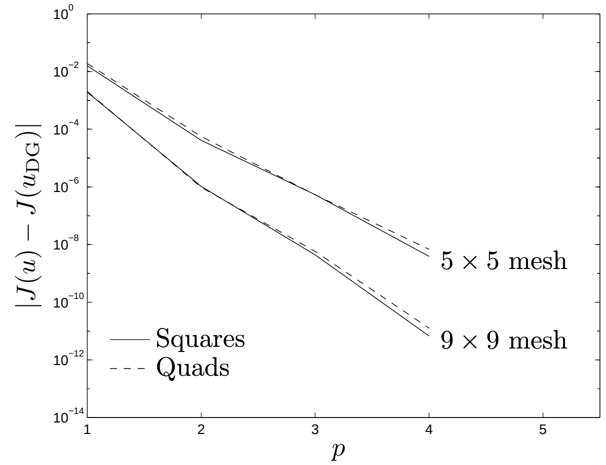


FIG. 5.2. *Example* 1. *Convergence of the hp-DGFEM with p-refinement.*

p become straight lines as the degree of the approximating polynomial is increased. Furthermore, we observe from Figures 5.1 and 5.2 that the h- and p-convergence, respectively, of the hp-DGFEM is robust with respect to mesh distortion.

5.2. Example 2. Here we consider a compressible hyperbolic problem subject to discontinuous inflow boundary condition with $\mathbf{b} = (2y^2 - 4x + 1, 1 + y)$, $c = 0$, and $f = 0$. The characteristics enter the computational domain Ω from three sides of Γ, namely, from $x = 0$, $y = 0$, and $x = 1$, and exit Ω through $y = 1$. Thus, we may prescribe

$$u(x,y) = \begin{cases} 0 & \text{for } x = 0,\ 0.5 < y \leq 1, \\ 1 & \text{for } x = 0,\ 0 \leq y \leq 0.5, \\ 1 & \text{for } 0 \leq x \leq 0.75,\ y = 0, \\ 0 & \text{for } 0.75 < x \leq 1,\ y = 0, \\ \sin^2(\pi y) & \text{for } x = 1,\ 0 \leq y \leq 1. \end{cases}$$

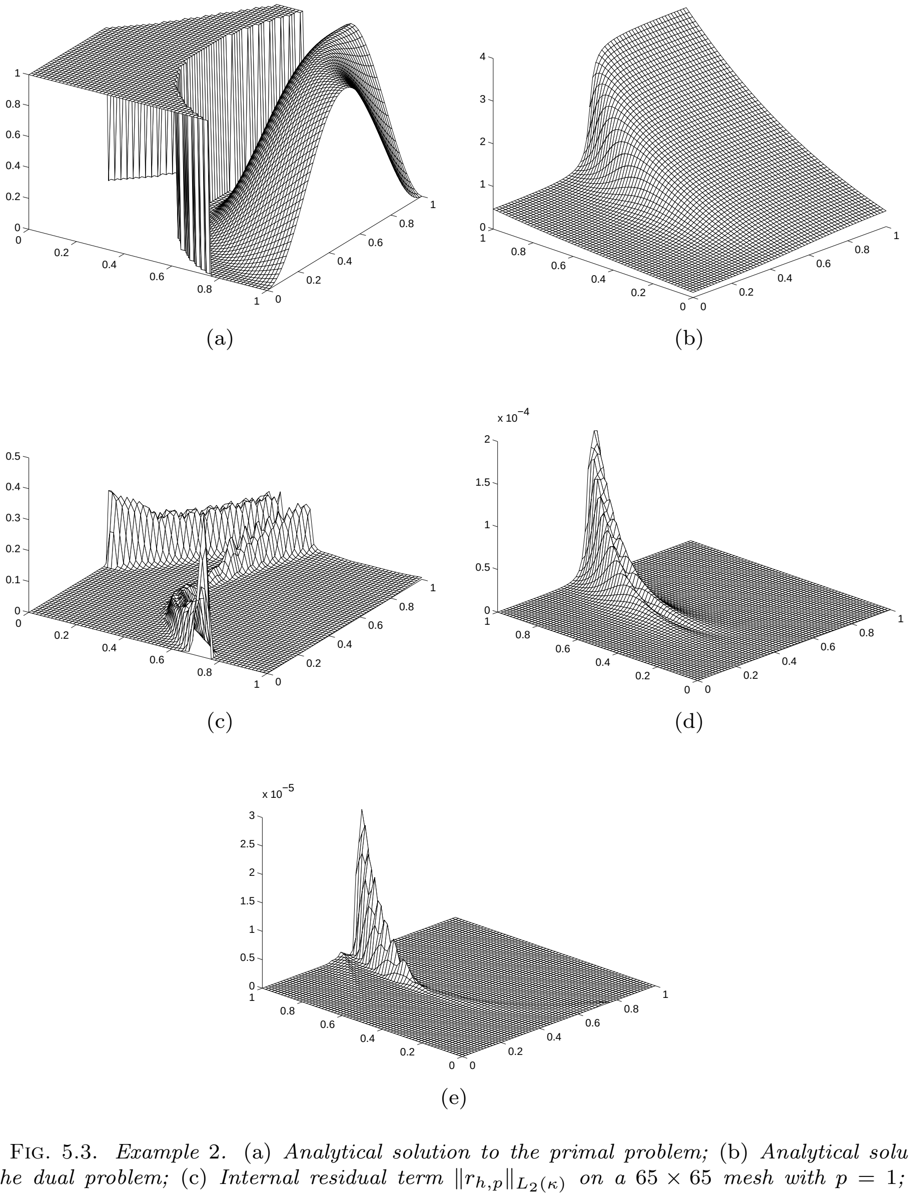


FIG. 5.3. *Example* 2. (a) *Analytical solution to the primal problem;* (b) *Analytical solution to the dual problem;* (c) *Internal residual term* $\|r_{h,p}\|_{L_2(\kappa)}$ *on a* 65×65 *mesh with* $p = 1$*;* (d) *Weighting term* $\|z - z_{h,p}\|_{L_2(\kappa)}$ *on a* 65×65 *mesh with* $p = 1$*;* (e) *Product of* (c) *and* (d)*, i.e.,* $\|r_{h,p}\|_{L_2(\kappa)}\,\|z - z_{h,p}\|_{L_2(\kappa)}$ *on a* 65×65 *mesh with* $p = 1$.

In this example we choose the functional of interest $J(\cdot)$ to represent the normal flux through the outflow boundary Γ_+, i.e., $J(\cdot) \equiv N_\psi(\cdot)$, where $N_\psi(\cdot)$ is given by (3.4); here, we define the weight function ψ by

$$\psi = 2 + \arctan((x - 1/2)/\varepsilon) \quad \text{for } 0 \le x \le 1,\ y = 1,$$

where $\varepsilon = 0.02$. Thereby, the true value of the outward normal flux is $N_\psi(u) = 2.0203$. The analytical solutions to both the primal and dual problems are shown in Figures 5.3(a) and 5.3(b), respectively. Furthermore, to understand how the terms

Table 5.1
Example 2. hp-mesh refinement algorithm.

Mesh	DOF	$\lvert N_\psi(u - u_{\mathrm{DG}})\rvert$	$\mathcal{E}_{\mathrm{P}}$	$\mathcal{E}_{\mathrm{D}}$	θ	$\mathcal{E}_{\mathrm{P}}/\mathcal{E}_{\mathrm{D}}$
1	16	6.112×10^{-2}	2.964×10^{-2}	5.707×10^{-2}	0.48	0.52
2	36	8.763×10^{-2}	6.169×10^{-2}	2.807×10^{-2}	0.70	2.20
3	90	3.547×10^{-2}	3.201×10^{-2}	4.803×10^{-3}	0.90	6.66
4	280	6.036×10^{-3}	5.496×10^{-3}	1.603×10^{-3}	0.91	3.43
5	702	1.472×10^{-3}	1.811×10^{-3}	1.158×10^{-4}	1.23	15.64
6	1904	4.107×10^{-5}	2.482×10^{-4}	3.941×10^{-6}	6.04	62.98
7	3679	2.502×10^{-5}	3.506×10^{-5}	3.353×10^{-6}	1.40	10.46
8	6414	5.715×10^{-7}	2.675×10^{-6}	3.320×10^{-7}	4.68	8.06
9	10493	9.012×10^{-8}	4.524×10^{-7}	2.083×10^{-8}	5.02	21.72
10	14107	3.175×10^{-8}	1.092×10^{-7}	9.175×10^{-9}	3.43	11.90

in the a posteriori error bound (3.14) interact with each other, in Figures 5.3(c) and 5.3(d), we have plotted the $L_2(\kappa)$ norm of the internal residual $r_{h,p}$ and the exact weight function $z - z_{h,p}$ on a 65×65 mesh with $p = 1$. Here, we observe that $\|r_{h,p}\|_{L_2(\kappa)}$ is large in the vicinity of the discontinuities as we would expect, while the weight function $\|z - z_{h,p}\|_{L_2(\kappa)}$ is large in the region where the layer in ψ enters the computational domain through Γ_+. The product of these two quantities is shown in Figure 5.3(e); here, we observe that the discontinuity emanating from $(x, y) = (0, 0.5)$ will have very little effect on the error in the linear function $N_\psi(\cdot)$ (see below).

In Table 5.1 we show the performance of the adaptive algorithm presented in section 4.2 for $\mathtt{TOL} = \mathtt{10}^{-7}$; we note that this level of accuracy may be far beyond what is of practical importance but is chosen to illustrate that the true error and the bound $\mathcal{E}_{\mathrm{P}}$ exhibit the same asymptotic behavior as the finite element space $S^{\mathbf{p}}(\Omega, \mathcal{T}, \mathbf{F})$ is enriched. In Table 5.1 we show the mesh number, the number of degrees of freedom (DOF) in $S^{\mathbf{p}}(\Omega, \mathcal{T}, \mathbf{F})$, the true error in the functional $|N_\psi(u - u_{\mathrm{DG}})|$, the error bound $\mathcal{E}_{\mathrm{P}}$, the remaining error $\mathcal{E}_{\mathrm{D}}$ (which in general will be noncomputable), the effectivity index $\theta = \mathcal{E}_{\mathrm{P}}/|N_\psi(u - u_{\mathrm{DG}})|$, and the ratio of $\mathcal{E}_{\mathrm{P}}$ and $\mathcal{E}_{\mathrm{D}}$. Here, we see that initially on very coarse meshes $\mathcal{E}_{\mathrm{P}}$ slightly underestimates the true error in the functional; however, as the finite element space is enriched the error bound overestimates $|N_\psi(u) - N_\psi(u_{\mathrm{DG}})|$ by a consistent factor in the range 1–6. Furthermore, we see that the remaining error term $\mathcal{E}_{\mathrm{D}}$ is about an order of magnitude smaller than the computable part of the a posteriori error bound $\mathcal{E}_{\mathrm{P}}$; this numerically justifies neglecting this term in the construction of the stopping criterion (4.4) for the design of our adaptive algorithm for the primal problem.

In Figure 5.4 we plot the results shown in Table 5.1; in particular, we plot $|N_\psi(u) - N_\psi(u_{\mathrm{DG}})|$, $\mathcal{E}_{\mathrm{P}}$, and $\mathcal{E}_{\mathrm{D}}$ using *hp*-refinement against the square root of the number of DOF on a linear-log scale. We see that after the initial transient, the error in the computed functional using *hp*-refinement becomes a straight line, thereby indicating exponential convergence; cf. Remark 3.12. Furthermore, in Figure 5.4 we plot the true error in the linear functional using *h*-refinement; here, we clearly observe the superiority of the adaptive *hp*-refinement algorithm. Indeed, on the final mesh the true error in the linear functional using *hp*-refinement is almost three orders of magnitude smaller than the true error in $N_\psi(\cdot)$ when *h*-refinement is employed.

Finally, in Figures 5.5 and 5.6 we show the primal and dual meshes after six and nine adaptive mesh refinements, respectively. For clarity, in each case we show the *h*-mesh alone, as well as the corresponding distribution of the polynomial degree and the percentage of elements with that degree. From Figure 5.5, we see that the elements in the primal mesh have been refined along the first discontinuity emanating

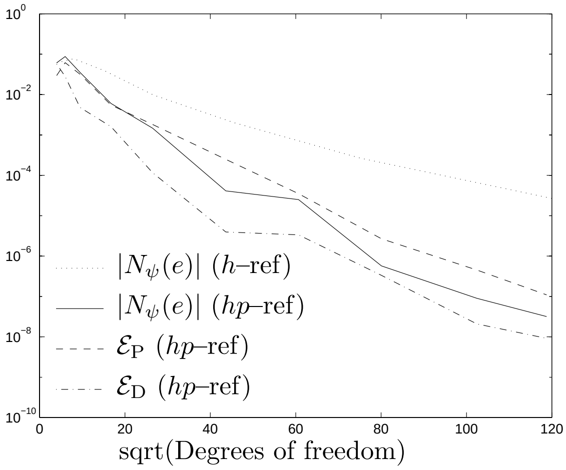


FIG. 5.4. *Example* 2. *Comparison between h- and hp-adaptive mesh refinement.*

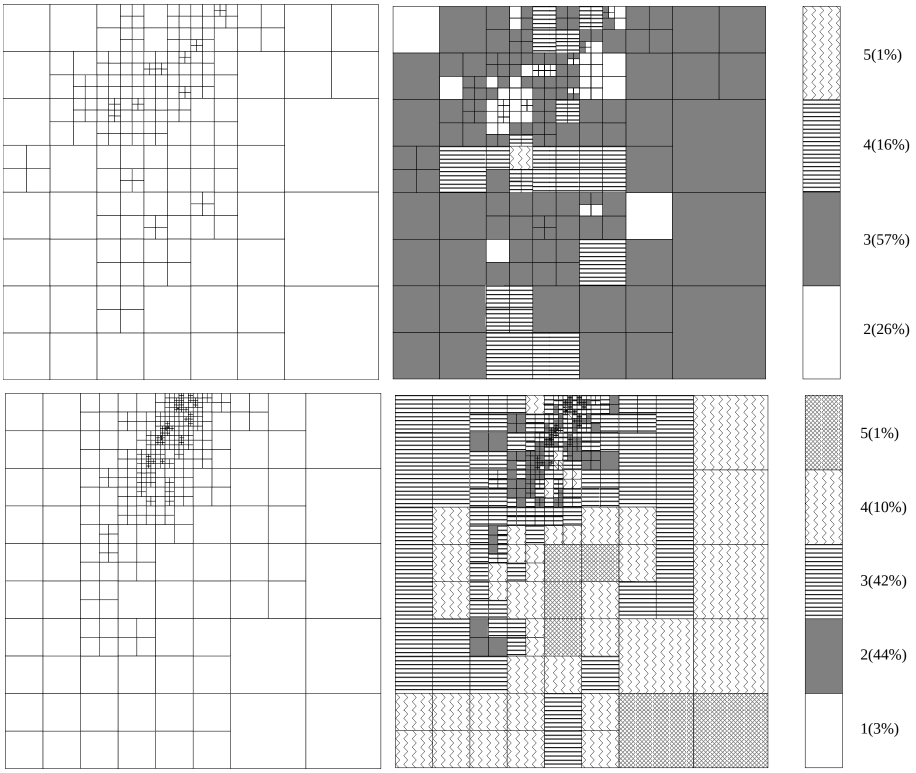


FIG. 5.5. *Example* 2. *Mesh* 7: *Primal (top:* 235 *elements,* 303 *nodes, and* 3679 *DOF) and dual (bottom:* 493 *elements,* 616 *nodes, and* 6781 *DOF) h- and hp-meshes. Here,* $N_\psi(u - u_{\rm DG}) = 2.502 \times 10^{-5}$, $\mathcal{E}_{\rm P} = 3.506 \times 10^{-5}$, *and* $\theta = 1.40$.

from $(x, y) = (0.75, 0)$, since the dual solution has a layer in this region as well. In contrast, elements lying on the second discontinuity in the primal problem, which emanates from $(x, y) = (0, 0.5)$, have been less refined, since the dual solution is smooth here, and hence the corresponding weights involving $\tilde{z}_{\rm DG} - z_{h,p}$ are inactive;

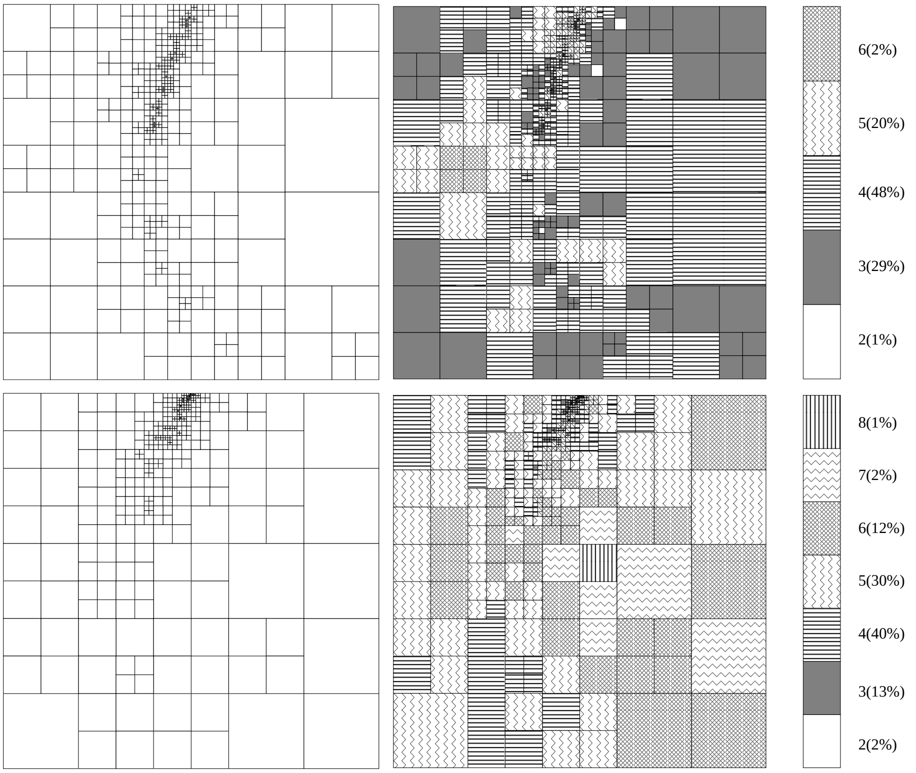


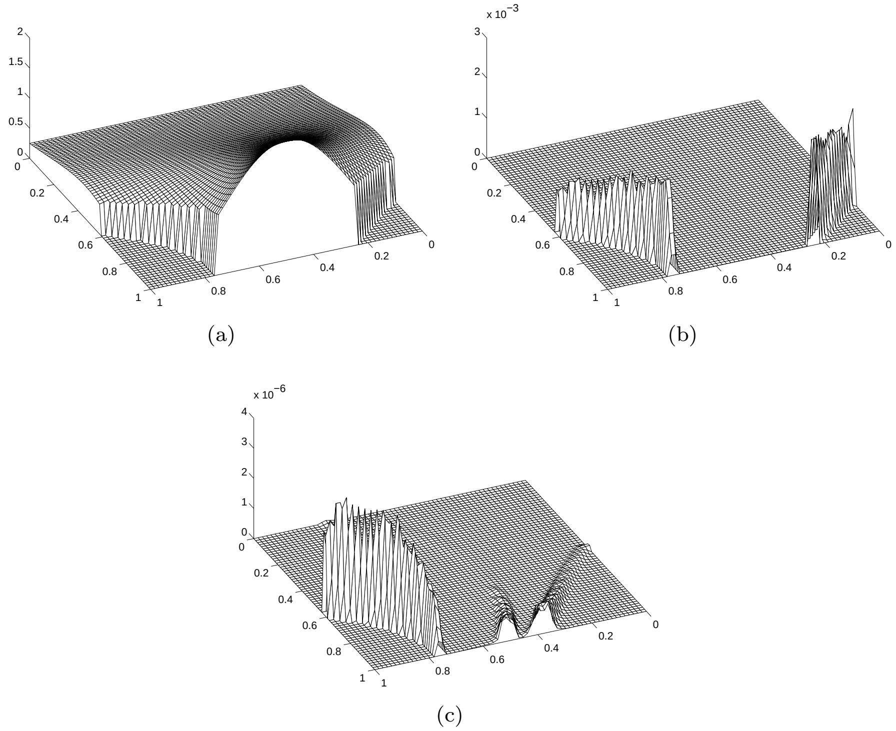
FIG. 5.6. *Example* 2. *Mesh* 10*: Primal (top:* 565 *elements,* 719 *nodes, and* 14107 *DOF) and dual (bottom:* 445 *elements,* 564 *nodes, and* 13629 *DOF) h- and hp-meshes. Here,* $N_\psi(u - u_{\mathrm{DG}}) = 3.175 \times 10^{-8}$, $\mathcal{E}_{\mathrm{P}} = 1.092 \times 10^{-7}$, *and* $\theta = 3.43$.

cf. Figure 5.3(e). Furthermore, the mesh for the dual solution is concentrated within the steep layer in the weight function ψ; the inherent smoothing in the dual problem introduced by the compressible nature of $\mathbf{b}$ leads to p-refinement in this layer as the flow moves away from Γ_+. The same behavior is observed in Figure 5.6 for the primal and dual solutions.

5.3. Example 3. In this final example, we consider the same primal problem presented in section 5.2. Furthermore, we again let $J(\cdot)$ denote the normal flux through the outflow boundary Γ_+, i.e., $J(\cdot) \equiv N_\psi(\cdot)$, where $N_\psi(\cdot)$ is given by (3.4); however, here we define the weight function ψ by

$$\psi = \begin{cases} 1 + \sin(2\pi(4x - 1)) & \text{for } 1/4 \leq x \leq 3/4, \\ 0 & \text{otherwise.} \end{cases}$$

In this case, the true value of the outward normal flux is $N_\psi(u) = 0.9175$. In Figures 5.7(a), 5.7(b), and 5.7(c) we plot the analytical to the dual problem, the weight function $\|z - z_{h,p}\|_{L_2(\kappa)}$, and the product of the weight function $\|z - z_{h,p}\|_{L_2(\kappa)}$ with the $L_2(\kappa)$ norm of the internal residual $r_{h,p}$ (see Figure 5.3(c) for the plot of $\|r_{h,p}\|_{L_2(\kappa)}$), respectively. From Figure 5.7(b), we see that the weighting term $\|z - z_{h,p}\|_{L_2(\kappa)}$ is large in the vicinity of the discontinuities emanating from Γ_+. By multiplying these terms by $\|r_{h,p}\|_{L_2(\kappa)}$ (cf. Figure 5.7(c)), we see that the discontinuity originating from $(0.75, 1)$ will dominate the error in the outward normal flux $N_\psi(\cdot)$; however, there



(a)

(b)

(c)

FIG. 5.7. *Example* 3. (a) *Analytical solution to the dual problem;* (b) *Weighting term* $\|z - z_{h,p}\|_{L_2(\kappa)}$ *on a* 65×65 *mesh with* $p = 1$*;* (c) *Product of* (b) *and Figure* 5.3(c)*, i.e.,* $\|r_{h,p}\|_{L_2(\kappa)} \|z - z_{h,p}\|_{L_2(\kappa)}$ *on a* 65×65 *mesh with* $p = 1$.

are visible peaks emanating from the top of the sin function as well as the second discontinuity located at $(0.25, 1)$.

In Figure 5.8 we show the performance of the adaptive algorithm for $\mathtt{TOL} = \mathtt{10}^{-6}$. We note that since both the primal and dual solutions are nonsmooth, we no longer expect to observe exponential convergence of the error in the functional $N_\psi(\cdot)$; thus, here we plot the same quantities as in Figure 5.4 against the number of degrees of freedom in $S^{\mathbf{p}}(\Omega, \mathcal{T}, \mathbf{F})$ on a log-log scale. We clearly observe that the error bound $\mathcal{E}_{\mathrm{P}}$ overestimates the true error by a consistent factor between 1–10. Furthermore, as in the previous example, the remaining error term $\mathcal{E}_{\mathrm{D}}$ is about an order of magnitude smaller than $\mathcal{E}_{\mathrm{P}}$, thereby justifying the absorption of $\mathcal{E}_{\mathrm{D}}$ into $\mathcal{E}_{\mathrm{P}}$. In addition, in Figure 5.8 we plot the true error in the linear functional using h-refinement; here, we observe that the true error in $N_\psi(\cdot)$ is (almost) always smaller if hp-refinement is employed. Indeed, on the final mesh the true error in $N_\psi(\cdot)$ is almost two orders of magnitude smaller if hp-refinement is employed as opposed to h-refinement only.

Finally, in Figures 5.9 and 5.10 we show the primal and dual meshes after six and nine adaptive mesh refinements, respectively. From Figure 5.9, we see that the primal mesh has only been h-refined in a small neighborhood of the two discontinuities in u as they exit the computational domain Ω. Furthermore, from Figure 5.10, we now see that the h-mesh has been refined in the vicinity of the discontinuity emanating from $(x, y) = (0, 0.5)$, while the h-mesh has in fact been coarsened in the region containing the second discontinuity originating from $(x, y) = (0.5, 0)$. In this latter region, the

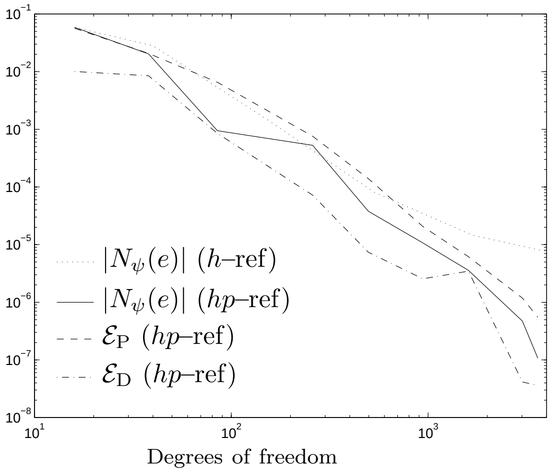


FIG. 5.8. *Example* 3. *Comparison between h- and hp-adaptive mesh refinement.*

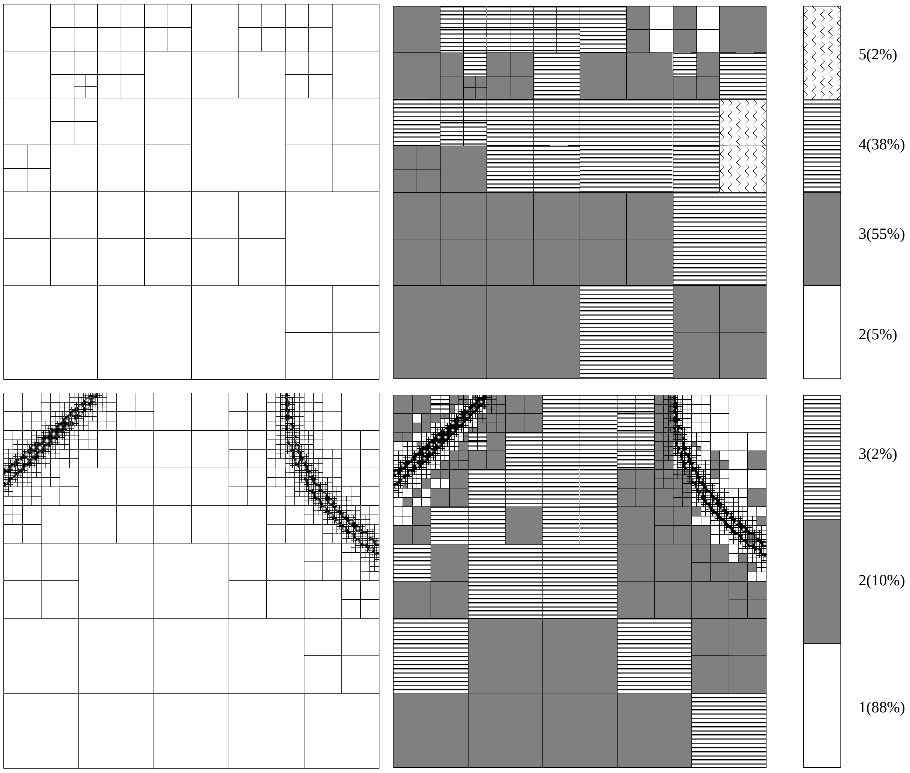


FIG. 5.9. *Example* 3. *Mesh* 7: *Primal (top:* 82 *elements,* 116 *nodes, and* 1603 *DOF) and dual (bottom:* 2629 *elements,* 3167 *nodes, and* 12316 *DOF) h- and hp-meshes. Here,* $N_\psi(u - u_{\rm DG}) = 3.568 \times 10^{-6}$, $\mathcal{E}_{\rm P} = 6.164 \times 10^{-6}$, *and* $\theta = 1.73$.

local polynomial degree has been enriched as the primal solution u is smooth here. Finally, Figures 5.9 and 5.10 show that the dual mesh has been extensively h-refined in the vicinity of both discontinuities with the degree p of the approximating polynomial increased as we move into the parts of the computational domain where the dual

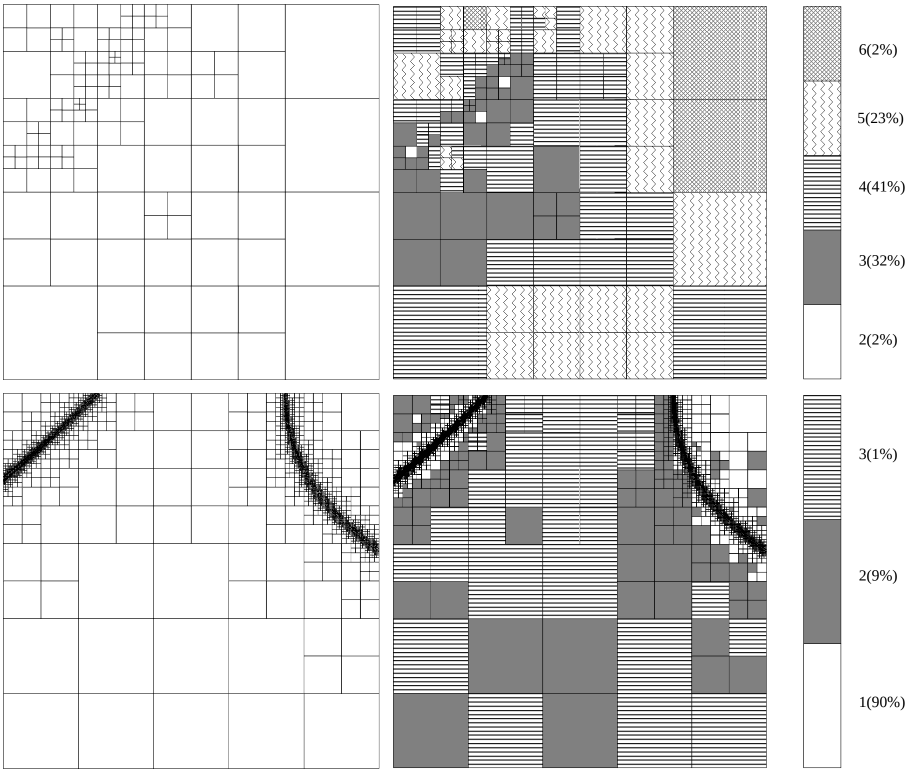


Fig. 5.10. *Example* 3. *Mesh* 10*: Primal (top:* 145 *elements,* 194 *nodes, and* 3609 *DOF) and dual (bottom:* 7630 *elements,* 9280 *nodes, and* 34451 *DOF) h- and hp-meshes. Here,* $N_\psi(u - u_{\rm DG}) = 1.072 \times 10^{-7}$, $\mathcal{E}_{\rm P} = 5.505 \times 10^{-7}$, *and* $\theta = 5.14$.

solution z is smooth.

6. Concluding remarks. In this article we have developed the a posteriori error analysis of the *hp*-version of the discontinuous Galerkin finite element method. In particular, by using a hyperbolic duality argument, we have derived computable error bounds for linear functionals of the solution, such as the mean flow of the field over the computational domain Ω and the normal flux through the outflow boundary Γ_+. Furthermore, based on our a posteriori error bound, we have designed and implemented a fully automatic adaptive algorithm that is capable of exploiting both local mesh subdivision and local polynomial-degree enrichment. Numerical experiments have been presented which clearly highlight the superiority of such a general adaptive strategy over the traditional *h*-refinement method, where the degree of the approximating polynomial p is kept fixed at some low value.

REFERENCES

[1] S. Adjerid, M. Aiffa, and J. E. Flaherty, *Computational methods for singularly perturbed systems*, in Singular Perturbation Concepts of Differential Equations, J. Cronin and R. E. O'Malley, eds., AMS, Providence, RI, 1998.

[2] M. Ainsworth and B. Senior, *An adaptive refinement strategy for hp–finite element computations*, Appl. Numer. Maths., 26 (1998), pp. 165–178.

[3] I. Babuška and M. Suri, *The hp-version of the finite element method with quasiuniform meshes*, RAIRO Modél. Math. Anal. Numér., 21 (1987), pp. 199–238.

[4] C. Bardos, *Problèmes aux limites pour les équations aux dérivées partielles du premier ordre à coefficients réels; théorèmes d'approximation; application à l'équation de transport*, Ann. Sci. École Norm. Sup. (4), 3 (1970), pp. 185–233.

[5] R. Becker and R. Rannacher, *Weighted A Posteriori Error Control in FE Methods*, Preprint 1, Interdisziplinäres Zentrum für Wissenschaftliches Rechnen, Universität Heidelberg, Heidelberg, Germany, 1996.

[6] K. S. Bey and J. T. Oden, *hp-version discontinuous Galerkin methods for hyperbolic conservation laws*, Comput. Methods Appl. Mech. Engrg., 133 (1996), pp. 259–286.

[7] K. S. Bey, J. T. Oden, and A. Patra, *A parallel hp-adaptive discontinuous Galerkin method for hyperbolic conservation laws*, Appl. Numer. Math., 20 (1996), pp. 321–336.

[8] D. Braess, *Finite Elements. Theory, Fast Solvers, and Applications in Solid Mechanics*, Cambridge University Press, Cambridge, UK, 1997.

[9] B. Cockburn, G. E. Karniadakis, and C.-W. Shu, *The development of discontinuous Galerkin methods*, in Discontinuous Galerkin Finite Element Methods, B. Cockburn, G. Karniadakis, and C.-W. Shu, eds., Lect. Notes Comput. Sci. Eng., 11, Springer-Verlag, Berlin, 2000, pp. 3–50.

[10] R. Dautray and J.-L. Lions, *Mathematical Analysis and Numerical Methods for Science and Technology, Vol.* 6: *Evolution Problems* II, Springer-Verlag, Berlin, 1992.

[11] K. Erikson, D. Estep, P. Hansbo, and C. Johnson, *Introduction to adaptive methods for differential equations*, Acta Numer., (1995), pp. 105–158.

[12] W. Gui and I. Babuška, *The h, p and h–p versions of the finite element method in* 1 *dimension. Part* III. *The adaptive h–p version*, Numer. Math., 49 (1986), pp. 659–683.

[13] R. Hartmann, *Adaptive FE-methods for conservation equations*, in Proceedings of the Eighth International Conference on Hyperbolic Problems, Theory, Numerics, Applications (HYP2000), G. Warnecke and H. Freistühler, eds., Birkhäuser, Basel, to appear.

[14] P. Houston, R. Rannacher and E. Süli, *A posteriori error analysis for stabilised finite element approximations of transport problems*, Comput. Methods Appl. Mech. Engrg., 190 (2000), pp. 1483–1508.

[15] P. Houston, Ch. Schwab, and E. Süli, *Discontinuous hp–finite element methods for advection–diffusion problems*, SIAM J. Numer. Anal., submitted.

[16] P. Houston, Ch. Schwab, and E. Süli, *Stabilized hp-finite element methods for first-order hyperbolic problems*, SIAM J. Numer. Anal., 37 (2000), pp. 1618–1643.

[17] C. Mavripilis, *Adaptive mesh strategies for the spectral element method*, Comput. Methods Appl. Mech. Engrg., 116 (1994), pp. 77–86.

[18] W. Rachowicz, J. T. Oden, and L. Demkowicz, *Toward a universal h-p adaptive finite element strategy.* III. *Design of h-p meshes*, Comput. Methods Appl. Mech. Engrg., 77 (1989), pp. 181–212.

[19] R. Rannacher, *Adaptive finite element methods*, in Error Control and Adaptivity in Scientific Computing, H. Bulgak and C. Zenger, eds., Kluwer Academic Publishers, Dordrecht, the Netherlands, 1998, pp. 247–278.

[20] J. Rauch, L_2 *is a continuable initial condition for Kreiss' mixed problems*, Comm. Pure Appl. Math., 25 (1972), pp. 265–285.

[21] Ch. Schwab, *p- and hp-Finite Element Methods. Theory and Applications to Solid and Fluid Mechanics*, Oxford University Press, Oxford, UK, 1998.

[22] E. Süli, *A posteriori error analysis and adaptivity for finite element approximations of hyperbolic problems*, in An Introduction to Recent Developments in Theory and Numerics for Conservation Laws, D. Kröner, M. Ohlberger, and C. Rohde, eds., Lect. Notes Comput. Sci. Eng. 5, Springer-Verlag, Berlin, 1998, pp. 123–194.

[23] E. Süli, P. Houston, and Ch. Schwab, *hp-Finite element methods for hyperbolic problems*, in The Mathematics of Finite Elements and Applications X, J. R. Whiteman, ed., Elsevier, New York, 2000, pp. 143–162.

[24] E. Süli, P. Houston, and B. Senior, *hp—discontinuous Galerkin finite element methods for hyperbolic problems: Error analysis and adaptivity*, in Numerical Methods for Fluid Dynamics VII, M. Baines, ed., ICFD, Oxford, UK, pp. 73–86.

SIAM J. SCI. COMPUT.
Vol. 23, No. 4, pp. 1253–1273

TIME RELAXED MONTE CARLO METHODS FOR THE BOLTZMANN EQUATION*

LORENZO PARESCHI† AND GIOVANNI RUSSO‡

Abstract. A new family of Monte Carlo schemes is introduced for the numerical solution of the Boltzmann equation of rarefied gas dynamics. The schemes are inspired by the Wild sum expansion of the solution of the Boltzmann equation for Maxwellian molecules and consist of a novel time discretization of the equation. In particular, high order terms in the expansion are replaced by the equilibrium Maxwellian distribution. The two main features of the schemes are high order accuracy in time and asymptotic preservation. The first property allows to recover accurate solutions with time steps larger than those required by direct simulation Monte Carlo (DSMC), while the latter guarantees that for the vanishing Knudsen number, the numerical solution relaxes to the local Maxwellian. Conservation of mass, momentum, and energy are preserved by the scheme. Numerical results on several space homogeneous problems show the improvement of the new schemes over standard DSMC. Applications to a one-dimensional shock wave problem are also presented.

Key words. Boltzmann equation, Monte Carlo methods, Wild sums, fluid-dynamic limit, Euler equations

AMS subject classifications. 65C05, 76P05, 82C80

PII. S1064827500375916

1. Introduction.

When a gas is near thermodynamical equilibrium, a detailed kinetic description is unnecessary, and the evolution of the moments can be described by Euler or Navier–Stokes equations. From a computational point of view, these equations are much less expensive than the full Boltzmann equation.

Far from thermodynamic equilibrium, the equation of gas dynamics do not give a satisfactory description of the physical system, and the full kinetic description is necessary.

The departure from local thermodynamic equilibrium can be measured by the so-called Knudsen number, which is the ratio between the collisional mean free path of the molecules and the characteristic length of variation of macroscopic variables. When the Knudsen number is small, then the distribution function is close to the local Maxwellian. Collisions occur at a fast rate, and a kinetic treatment of the system is very expensive, because of the large ratio of time scales between macroscopic and microscopic effects. One may say that the system is numerically *stiff* [10]. Because of the stiffness, standard explicit schemes for the numerical solution of the Boltzmann equation become extremely expensive in this case.

If the Knudsen number is uniformly small in the region one is interested in, then one should use a gas dynamic description of the system. There are several circumstances, however, where the Knudsen number varies over several orders of magnitude. In such cases, a standard kinetic treatment would be too expensive and a pure gas dynamic approach inappropriate. Domain decomposition methods have been pro-

*Received by the editors July 31, 2000; accepted for publication (in revised form) March 9, 2001; published electronically December 4, 2001. This work was supported by TMR project “Asymptotic Methods in Kinetic Theory,” contract ERB FMRX CT97 0157.

http://www.siam.org/journals/sisc/23-4/37591.html

†Universitá di Ferrara, Dipartimento di Matematica, Via Machiavelli 35, 44100 Ferrara, Italy (pareschi@dm.unife.it).

‡Universitá dell’Aquila, Dipartimento di Matematica, Via Vetoio, Loc. Coppito, 67010 L’Aquila, Italy (russo@univaq.it).

posed for this problem, in which the computational domain is divided into an *aerodynamic region* in which the system is treated by gas dynamic equations (Euler or Navier–Stokes), and a *Boltzmann region*, where a kinetic treatment is used. Suitable boundary conditions make the two regions coupled together [5]. Such schemes can be very effective in treating large scale problems, but they are quite complicated to implement, and they require a great care by the user.

In some recently developed Monte Carlo methods, a different approach is taken with the goal of constructing simple and efficient numerical methods for the solution of the Boltzmann equation in regions with a large variation in the mean free path [16, 15].

These algorithms are based on a suitable time discretization of the Boltzmann equation, first introduced in [9], which has the following properties: for the large Knudsen number, it is a time discretization of the Boltzmann equation of a given order, and when the Knudsen number vanishes, the numerical solution tends to the local Maxwellian. We will call this last property *asymptotic preservation.* In addition, the time discretization preserves mass, momentum, and energy.

Although the paper deals mainly with the space homogeneous case, the real motivation for the development of these Monte Carlo methods is their use for spatially dependent problems. In that case, standard splitting can be used, and the evolution of the system is obtained by alternating a collision step, which is space homogeneous in each space cell, and a convection step, which describes the free flow of the molecules. In the limit of the very small Knudsen number, the collision step replaces the distribution function by a local Maxwellian with the same moments. A Monte Carlo method based on such time discretization will behave as a sort of stochastic kinetic scheme for the underlying Euler equations of gas dynamics. Such a limit scheme was introduced by Pullin [18].

The plan of the paper is the following. In the next section, we give a short introduction on the Boltzmann equation and some of its relevant properties. In section 3, we introduce the time relaxed (TR) discretizations, starting from the Wild sum expansion, and making use of some of its properties. Section 4 is devoted to the derivation of time relaxed Monte Carlo (TRMC) methods. The section starts with a review of direct simulation Monte Carlo (DSMC), in particular of the Nanbu–Babovsky scheme [1], which is revisited and reinterpreted in a new light. A detailed description is given of TRMC algorithms. In the last section, we show some numerical tests performed with the new schemes. In particular, it is shown that it is possible to obtain the same accuracy of the Nanbu–Babovsky scheme with a lower computational cost. We conclude the paper with a plan of future extensions and improvements.

2. Time discretizations of the Boltzmann equation.

2.1. The Boltzmann equation. We consider the Boltzmann equation [7]

$$\frac{\partial f}{\partial t} + v \cdot \nabla_x f = \frac{1}{\varepsilon} Q(f, f), \tag{2.1}$$

supplemented with the initial condition

$$f(x, v, t = 0) = f_0(x, v), \tag{2.2}$$

where $f = f(x, v, t)$ is a nonnegative function describing the time evolution of the distribution of particles which move with velocity v in the position x at time $t > 0$. The parameter $\varepsilon > 0$ is the Knudsen number and is proportional to the mean free path between collisions. The bilinear collision operator $Q(f,f)$ describes the binary collisions of the particles and is given by

$$Q(f,f)(v) = \int_{\mathbb{R}^3}\int_{S^2} \sigma(|v-v_1|,\omega)[f(v')f(v'_*) - f(v)f(v_*)]\,d\omega\,dv_*. \tag{2.3}$$

In the above expression, ω is a unit vector of the sphere S^2 so that $d\omega$ is an element of area of the surface of the unit sphere S^2 in $\mathbb{R}^3$. Moreover, (v', v'_*) represent the postcollisional velocities associated with the precollisional velocities (v, v_*) and the collision parameter ω; i.e.,

$$v' = \frac{1}{2}(v + v_* + |v-v_*|\omega), \quad v'_* = \frac{1}{2}(v + v_* - |v-v_*|\omega). \tag{2.4}$$

The kernel σ is a nonnegative function which characterizes the details of the binary interactions. In the case of inverse kth power forces between particles, the kernel has the form

$$\sigma(|v-v_*|,\theta) = b_\alpha(\theta)|v-v_*|^\alpha, \tag{2.5}$$

where $\alpha = (k-5)/(k-1)$. For numerical purposes, a widely used model is the variable hard sphere (VHS) model [3], corresponding to take $b_\alpha(\theta) = C_\alpha$ where C_α, is a positive constant. The case $\alpha = 0$ is referred to as the Maxwellian gas, whereas the case $\alpha = 1$ yields the hard sphere gas.

2.2. Fluid-dynamical limit. During the evolution process, the collision operator preserves mass, momentum, and energy; i.e.,

$$\int_{\mathbb{R}^3} Q(f,f)\phi(v)\,dv = 0, \quad \phi(v) = 1, v, v^2, \tag{2.6}$$

and in addition it satisfies Boltzmann's well-known H-theorem

$$\int_{\mathbb{R}^3} Q(f,f)\log(f)dv \le 0. \tag{2.7}$$

From a physical point of view, Boltzmann's H-theorem implies that any equilibrium distribution function, i.e., any function f for which $Q(f,f) = 0$, has the form of a locally Maxwellian distribution

$$M(\rho,u,T)(v) = \frac{\rho}{(2\pi T)^{3/2}} \exp\left(-\frac{|u-v|^2}{2T}\right), \tag{2.8}$$

where ρ, u, and T are the density, mean velocity, and temperature of the gas defined by

$$\rho = \int_{\mathbf{R}^3} f\,dv, \qquad u = \frac{1}{\rho}\int_{\mathbf{R}^3} vf\,dv, \qquad T = \frac{1}{3\rho}\int_{\mathbf{R}^3}[v-u]^2 f\,dv. \tag{2.9}$$

As $\varepsilon \longrightarrow 0$, the distribution function approaches the local Maxwellian (2.8). In this case, the higher order moments of the distribution f can be computed as function of ρ, u, and T, by using (2.8), and we obtain to the leading order the closed system of compressible Euler equations of gas dynamics

$$\begin{aligned}
&\frac{\partial \rho}{\partial t} + \nabla_x \cdot (\rho u) = 0,\\
&\frac{\partial \rho u}{\partial t} + \nabla_x \cdot (\rho u \otimes u) + \nabla_x p = 0, \qquad (2.10)\\
&\frac{\partial E}{\partial t} + \nabla_x \cdot (Eu + pu) = 0,
\end{aligned}$$

$$p = \rho T, \quad E = \frac{3}{2}\rho T + \frac{1}{2}\rho u^2. \qquad (2.11)$$

2.3. Time discretizations. The time integration of the Boltzmann equation represents a challenging problem, since the nonlinear collision operator becomes highly stiff near the fluid regime ($\varepsilon \ll 1$). An additional limitation is given by the high computational cost required for evaluating the fivefold collisional integral. The starting point is the usual first order splitting in time of (2.1), which consists of solving separately a purely convective step (i.e., $Q \equiv 0$ in (2.1)) and a collision step characterized by a space homogeneous Boltzmann equation (i.e., $\nabla_x f \equiv 0$ in (2.1)). Clearly, after this splitting, almost all the main difficulties are contained in the collision step. For this reason, in what follows we will fix our attention on the time discretization of the homogeneous Boltzmann equation

$$\frac{\partial f}{\partial t} = \frac{1}{\varepsilon} Q(f,f). \qquad (2.12)$$

Let us consider the simple forward Euler scheme

$$f^{n+1} = f^n + \frac{\Delta t}{\varepsilon} Q(f^n, f^n), \qquad (2.13)$$

which has been widely used in simulation. It leads to a stability condition which requires the time step Δt to be of order ε. Thus for small values of ε the scheme is unusable for practical purposes.

On the other hand, the use of fully implicit schemes for (2.12), like the backward Euler method

$$f^{n+1} = f^n + \frac{\Delta t}{\varepsilon} Q(f^{n+1}, f^{n+1}), \qquad (2.14)$$

requires the solution of a large nonlinear system of algebraic equations, and this makes the method prohibitively expensive.

Alternatively, semi-implicit discretizations have been proposed [20] based on taking only the function $f(v)$ implicitly in (2.3), whereas $f(v')$, $f(v'_*)$, and $f(v_*)$ are evaluated explicitly. This gives rise to unconditionally stable schemes that have the same cost of an explicit scheme. Unfortunately, these schemes do not possess the correct fluid limit as $\varepsilon \to 0$, and, in addition, except for Maxwell molecules, the numerical solution does not preserve the conserved quantities.

High order extensions of the previous schemes can be developed. At present, however, the practical interest of high order schemes is strongly reduced by the high computational cost required by multiple evaluations of the fivefold integral (2.3).

In the next sections, following the approach presented in [9], we will show how it is possible to construct unconditionally stable schemes for the space homogeneous equation that are accurate for a wide range of Knudsen numbers, while avoiding the solution of nonlinear algebraic equations.

3. The TR schemes.

As proposed in [9], a general idea for deriving robust numerical schemes, that is, schemes that are unconditionally stable and preserve the asymptotic of the fluid-dynamic limit, for a nonlinear equation like (2.12), is to replace high order terms of a suitable well-posed power series expansion by the local equilibrium. We will call this class of schemes TR schemes. Here we will first derive the schemes as presented in [9], and then we will show how it is possible to generalize this approach.

3.1. Derivation.

Let us consider a differential system of the type

$$\frac{\partial f}{\partial t} = \frac{1}{\varepsilon}\left[P(f,f) - \mu f\right], \tag{3.1}$$

with the same initial condition (2.2), and where $\mu \neq 0$ is a constant and P a bilinear operator.

Let us replace the time variable t and the function $f = f(v,t)$ using the equations

$$\tau = (1 - e^{-\mu t/\varepsilon}), \qquad F(v,\tau) = f(v,t)e^{\mu t/\varepsilon}. \tag{3.2}$$

Then F is easily shown to satisfy

$$\frac{\partial F}{\partial \tau} = \frac{1}{\mu}P(F,F) \tag{3.3}$$

with $F(v,\tau = 0) = f_0(v)$.

Now, the solution to the Cauchy problem for (3.3) can be sought in the form of a power series

$$F(v,\tau) = \sum_{k=0}^{\infty} \tau^k f_k(v), \qquad f_{k=0}(v) = f_0(v), \tag{3.4}$$

where the functions f_k are given by the recurrence formula

$$f_{k+1}(v) = \frac{1}{k+1}\sum_{h=0}^{k} \frac{1}{\mu}P(f_h, f_{k-h}), \quad k = 0,1,\dots. \tag{3.5}$$

Making use of the original variables, we obtain the following formal representation of the solution to the Cauchy problem (2.12):

$$f(v,t) = e^{-\mu t/\varepsilon}\sum_{k=0}^{\infty}(1 - e^{-\mu t/\varepsilon})^k f_k(v). \tag{3.6}$$

Remark 3.1. The method was originally developed by Wild [22, 6] to solve the Boltzmann equation for Maxwellian molecules. Here we describe the method under a more general hypothesis on P, as derived in [9]. We emphasize that the representation (3.6) is not unique, and other well-posed power series expansions can be obtained in a similar way [9].

Finally, note that expansion (3.6) continues to hold also if μ is a function of v. Unfortunately, this choice leads to nonconservative schemes.

From this representation, a class of numerical schemes can be naturally derived.

In [9], the following class of numerical schemes, based on a suitable truncation for $m \geq 1$ of (3.6), has been constructed:

$$f^{n+1}(v) = e^{-\mu\Delta t/\varepsilon}\sum_{k=0}^{m}(1 - e^{-\mu\Delta t/\varepsilon})^k f_k^n(v) + (1 - e^{-\mu\Delta t/\varepsilon})^{m+1}M(v), \tag{3.7}$$

where $f^n = f(n\Delta t)$ and Δt is a small time interval. The quantity M (referred to as the local Maxwellian associated with f) is the asymptotic stationary solution of the equation.

It can be shown that the schemes obtained in this way are of order m in time. Furthermore, we have [9] the following proposition.

PROPOSITION 3.1. *The schemes defined by* (3.7) *satisfy the following:*

(i) Conservation.

If $P(f,g)$ is a nonnegative bilinear operator such that there exist some functions $\phi(v)$ with the following property,

$$\int_{R^3} P(f,f)\phi(v)\,dv = \mu \int_{R^3} f\phi(v)\,dv, \tag{3.8}$$

and the initial condition f^0 is a nonnegative function, then f^{n+1} is nonnegative for any $\mu\Delta t/\varepsilon$ and satisfies

$$\int_{R^3} f^{n+1}\phi(v)\,dv = \int_{R^3} f^n\phi(v)\,dv. \tag{3.9}$$

(ii) Asymptotic preservation (AP).

For any $m \geq 1$, we have

$$\lim_{\mu\Delta t/\varepsilon\to\infty} f^{n+1} = M(v). \tag{3.10}$$

3.2. Application to the Boltzmann equation. Now we apply the previous general approach to the Boltzmann equation and related kinetic models. To this aim, we will need a bound on the loss term, which is essential in most numerical methods.

Let us consider the space homogeneous problem

$$\frac{\partial f}{\partial t} = \frac{1}{\varepsilon}Q(f,f). \tag{3.11}$$

Problem (3.11) can be written in the form (3.1), taking

$$P(f,f) = Q^+(f,f) + f(v)\left(\mu - \int_{\mathbb{R}^3}\int_{S^2} \sigma(|v-v_*|,\omega)f(v_*)\,d\omega\,dv_*\right), \tag{3.12}$$

where

$$Q^+(f,f,) = \int_{\mathbb{R}^3}\int_{S^2} \sigma(|v-v_*|,\omega)f(v')f(v'_*)\,d\omega\,dv_*, \tag{3.13}$$

and the constant μ must satisfy

$$\mu \geq \int_{\mathbb{R}^3}\int_{S^2} \sigma(|v-v_*|,\omega)f(v_*)\,d\omega\,dv_*. \tag{3.14}$$

In general, the previous condition cannot be satisfied for all $v \in \mathbb{R}^3$. The usual way to overcome this difficulty is to replace the kernel σ with the following bounded kernel σ_b:

$$\sigma_b(|v-v_*|,\omega) = \min\{\sigma(|v-v_*|,\omega),\bar{\sigma}\}, \quad \bar{\sigma} > 0.$$

In this case, it is enough to choose $\mu = 4\pi\rho\bar{\sigma}$ in order to satisfy (3.14).

Note that, in a particle method, there is a natural bound $\bar{\sigma}$ on the cross section due to the finite number of velocities, and therefore it is always possible to satisfy (3.14).

It is a simple exercise to verify that Proposition 3.1 holds with $\phi(v) = 1, v, v^2$.

Let us consider the first order scheme, in which $f_k, k \geq 2$, is substituted by the local Maxwellian (2.8). With the notations of the previous sections, the scheme reads

$$f^{n+1}(v) = (1-\tau)f^n(v) + \tau(1-\tau)f_1^n(v) + \tau^2 M(v), \tag{3.15}$$

where $\tau = 1 - e^{-\mu\Delta t/\varepsilon}$. The results from the last two subsections show that the approximation defined by (3.15) is well defined independently of the Knudsen number and, by Proposition 3.1, has the correct moments and converges towards the correct fluid-dynamic limit.

Remark 3.2. In the case of Maxwell molecules, the method simplifies considerably. In fact, since the collision kernel does not depend on the relative velocity $\sigma_b = \sigma_b(\omega)$, we simply have $P(f,f) = Q^+_{\bar{\sigma}}(f,f)$ and $\mu = 4\pi\rho\bar{\sigma}$, where

$$\bar{\sigma} = \frac{1}{4\pi}\int_{S^2} \sigma_b(\omega)d\omega. \tag{3.16}$$

3.3. Generalized TR schemes. The approach we have seen in the previous sections can be generalized using different weight functions to combine the influence of the high order coefficients appearing in the Wild sum (3.6). In general, such schemes can be written as

$$f^{n+1} = \sum_{k=0}^{m} A_k f_k + A_{m+1}M, \tag{3.17}$$

where the coefficients f_k are given by (3.5).

The weights $A_k = A_k(\tau)$ are nonnegative functions that satisfy the following proposition.

PROPOSITION 3.2. *Let the weights $A_k = A_k(\tau)$ be nonnegative functions that satisfy the following.*

(i) Consistency.

$$\lim_{\tau\to 0} A_1(\tau)/\tau = 1, \quad \lim_{\tau\to 0} A_k(\tau)/\tau = 0, \quad k = 2,\ldots,m+1. \tag{3.18}$$

(ii) Conservation.

$$\sum_{k=0}^{m+1} A_k = 1, \quad \tau \in [0,1]. \tag{3.19}$$

(iii) Asymptotic preservation (AP).

$$\lim_{\tau\to 1} A_k(\tau) = 0, \quad k = 0,\ldots,m; \tag{3.20}$$

then (3.17) *is a consistent discretization of problem* (3.1) *that satisfies Proposition* 3.1.

The proof is a simple exercise, and we leave it to the reader.

A choice of functions which satisfies the previous requirements is given by

$$A_k = (1-\tau)\tau^k,\ k = 0,\dots,m, \quad A_{m+1} = \tau^{m+1}, \tag{3.21}$$

which correspond to the scheme (3.7). A better choice of parameters is [16]

$$A_k = (1-\tau)\tau^k,\ k = 0,\dots,m-1, \quad A_m = 1-\sum_{k=0}^{m} A_k - A_{m+1}, \quad A_{m+1} = \tau^{m+2}, \tag{3.22}$$

which corresponds to take $f_{m+1} = f_m$, $f_k = M$, $k \geq m+2$ in (3.6).

However, other choices are possible; and it is an open problem, the determination of the *optimal* set of functions A_k, that satisfies the previous requirements and guarantees the most accurate approximation.

3.4. Stability analysis of the TR schemes. In this paragraph, we report some results related to the stability of the TR schemes. In particular, we show that the first and second order TR schemes are L-stable [16].

Let us apply the TR scheme to the test equation

$$y' = \lambda y \tag{3.23}$$

with $y(0) = 1$ and $\lambda \in C^-$.

The equation can be written in the form

$$y' = P(y,y) - \mu y,$$

with $P(f,g) = \frac{1}{2}(\mu+\lambda)(f+g)$, and μ is a positive constant such that $|\lambda| < \mu$. The numerical solution after one time step is given by

$$y^1 = \sum_{k=0}^{m} A_k y_k. \tag{3.24}$$

The $m+1$ term vanishes because the equilibrium solution is zero. The term y_k can be explicitly computed by applying formula (3.5). A simple calculation yields

$$y_k = \frac{1}{k!}\prod_{j=1}^{k}(j+a) = \frac{\Gamma(k+1+a)}{\Gamma(k+1)\,\Gamma(1+a)}, \quad k = 0,\dots,m, \tag{3.25}$$

where $a = \lambda/\mu$ and Γ denotes the usual Γ-function. The numerical solution after one time step, called the *function of absolute stability*, is therefore a function of a real parameter τ and a complex parameter a such that

$$R_m(\tau,a) = \sum_{k=0}^{m} A_k(\tau) y_k(a),$$

where $\tau = 1-\exp(-\mu\Delta t)$ is the relaxed time, and a is a complex number $a \in C^-$, $|a| \leq 1$. Here we give definitions of A- and L-stability which are the natural generalization of the standard definition [10] to the case where the function of absolute stability depends on an additional real parameter.

DEFINITION 3.3.

(i) *We say that the scheme is A-stable if*

$$|R(\tau,a)| \le 1 \quad \forall \tau \in [0,1],\ a \in C^-, |a| \le 1.$$

(ii) *We say that the scheme is L-stable if it is A-stable and*

$$\lim_{\tau\to 1} R(\tau,a) = 0 \quad \forall a \in C^-, |a| \le 1.$$

It is clear that, because of the asymptotic preserving properties, if a TR scheme is A-stable it is also L-stable.

Here we recall the main result about the stability of first and second order TR schemes with weights given by (3.21).

The functions of absolute stability for these schemes are

$$R_1(\tau,a) = (1-\tau)[1+\tau(1+a)], \tag{3.26}$$

$$R_2(\tau,a) = (1-\tau)\left[1+\tau(1+a)+\frac{1}{2}\tau^2(1+a)(2+a)\right]. \tag{3.27}$$

We have the following theorem [16].

THEOREM 3.4. *The schemes defined by* (3.21) *for* $m=1$ *and* $m=2$ *are A-stable, i.e.,*

$$|R_1(\tau,a)| \le 1 \quad \forall \tau \in [0,1],\ a \in C^-, |a| \le 1,$$

$$|R_2(\tau,a)| \le 1 \quad \forall \tau \in [0,1],\ a \in C^-, |a| \le 1.$$

4. TRMC methods.

In this section, we describe the TRMC method for the evolution of the density function f. We shall include the description of the classical DSMC no-time counter algorithm, so that it will be easier to make a comparison between the new formulation and the previous one.

We develop the algorithms first in the simple case of constant cross sections (Maxwellian molecules) and then for the Boltzmann equation for VHS gas.

4.1. DSMC methods.

Here we will describe the classical DSMC method in the general framework we have introduced in the previous sections. More specifically, we consider the Nanbu–Babovsky algorithm [14, 1]. The convergence of this scheme has been proved by Babovsky and Illner [2].

The kinetic equations we are considering can be written in the form

$$\frac{\partial f}{\partial t} = \frac{1}{\epsilon}[P(f,f) - \mu f]. \tag{4.1}$$

In particular, for the Boltzmann equation for Maxwellian molecules, we have $P(f,f) = Q^+(f,f)$ (see *Remark* 3.2), whereas for general kernels with cut-off, $P(f,f)$ is given by (3.12).

We assume that f is a probability density; i.e.,

$$\rho = \int_{\mathbb{R}^3} f(v,t)\,dv = 1.$$

Let us discretize time and denote by $f^n(v)$ an approximation of $f(v,n\Delta t)$. The forward Euler scheme applied to (4.1) writes

$$f^{n+1} = \left(1 - \frac{\mu\Delta t}{\epsilon}\right) f^n + \frac{\mu\Delta t}{\epsilon}\frac{P(f^n,f^n)}{\mu}. \tag{4.2}$$

This equation has the following probabilistic interpretation: a particle with velocity v_i will not collide with probability $(1-\mu\Delta t/\epsilon)$, and it will collide with probability $\mu\Delta t/\epsilon$, according to the collision law described by $P(f^n, f^n)(v)$.

We remark here that the probabilistic interpretation of (4.2) breaks down if $\Delta t/\epsilon$ is too large because the coefficient of f^n on the right-hand side may become negative. This implies that the time step becomes extremely small when approaching the fluid-dynamic limit. Therefore, the classical DSMC method becomes almost unusable near the fluid regime.

For clarity of exposition, let us first consider kinetic equations for which $P(f,f) = Q^+(f,f)$; i.e., the collision kernel does not depend on the relative velocity of the particles.

An algorithm based on this probabilistic interpretation was proposed by Nanbu [14].

In its first version, Nanbu's algorithm was not conservative; i.e., energy and momentum were conserved only in the mean but not at each collision. A conservative version of the algorithm was introduced by Babovsky [1]. Instead of selecting single particles, independent particle pairs are selected, and conservation is maintained at each collision. The expected number of particles that collide in a small time step Δt is $N\mu\Delta t/\epsilon$, and the expected number of collision pairs is $N\mu\Delta t/(2\epsilon)$. The algorithm for evolving the density up to time $t = n_{\mathrm{TOT}}\Delta t$ is the following.

Algorithm 4.1 (DSMC for Maxwell molecules).

```
• compute the initial velocity of the particles, $\{v_i^0, i = 1, \dots, N\}$,
  by sampling them from the initial density $f_0(v)$
• for $n = 1$ to $n_{\mathrm{TOT}}$
      given $\{v_i^n, i = 1, \dots, N\}$,
          ○ compute $\{v_i^{n+1}\}$ as follows:
          ○ set $N_c = \mathrm{Iround}(\mu N\Delta t/(2\epsilon))$
          ○ select $N_c$ pairs $(i, j)$ uniformly among all possible pairs,
            and for those
              ○ perform the collision between $i$ and $j$, and
                compute $v_i'$ and $v_j'$ according to the collisional law
              ○ set $v_i^{n+1} = v_i'$, $v_j^{n+1} = v_j'$
          ○ set $v_i^{n+1} = v_i^n$ for all particles that have not been selected
  end for
```

During each step, all the other $N - 2N_c$ particle velocities remain unchanged. Here, by $\mathrm{Iround}(x)$, we denote a suitable integer rounding of a positive real number x. In our algorithm, we choose

$$\mathrm{Iround}(x) = \begin{cases} [x] & \text{with probability} \quad [x] + 1 - x, \\ [x] + 1 & \text{with probability} \quad x - [x], \end{cases}$$

where $[x]$ denotes the integer part of x.

The postcollisional velocities are computed through relations

$$v_i' = \frac{v_i + v_j}{2} + \frac{|v_i - v_j|}{2}\omega, \quad v_j' = \frac{v_i + v_j}{2} - \frac{|v_i - v_j|}{2}\omega, \tag{4.3}$$

where ω is chosen uniformly in the unit sphere, according to

$$\omega = \begin{pmatrix} \cos\phi\sin\theta \\ \sin\phi\sin\theta \\ \cos\theta \end{pmatrix}, \quad \theta = \arccos(2\xi_1 - 1), \quad \phi = 2\pi\xi_2, \tag{4.4}$$

and ξ_1, ξ_2 are uniformly distributed random variables in $[0, 1]$.

Note that this approach is equivalent to sampling the postcollisional velocity according to $P(f, f)/\mu$, where $\mu = 4\pi\sigma$ and σ is the constant cross section.

The above algorithm has to be modified when the scattering cross section is not constant. In this case, a simple algorithm is obtained by using an upper bound $\bar{\sigma}$ of the scattering cross section and the acceptance-rejection technique.

Algorithm 4.2 (DSMC for VHS molecules).

```
• compute the initial velocity of the particles, $\{v_i^0, i = 1, \dots, N\}$,
  by sampling them from the initial density $f_0(v)$
• for $n = 1$ to $n_{\mathrm{TOT}}$
      given $\{v_i^n, i = 1, \dots, N\}$,
          ◦ compute an upper bound $\bar{\sigma}$ for the cross section
          ◦ set $\mu = 4\pi\bar{\sigma}$
          ◦ set $N_c = \mathrm{Iround}(\mu N \Delta t/(2\epsilon))$
          ◦ select $N_c$ dummy collision pairs $(i, j)$ uniformly
            among all possible pairs, and for those
              ◦ compute the relative cross section $\sigma_{ij} = \sigma(|v_i - v_j|)$
              ◦ if $\bar{\sigma}\,\mathrm{Rand} < \sigma_{ij}$
                    ◦ perform the collision between $i$ and $j$, and
                      compute $v_i'$ and $v_j'$ according to the collisional law
                    ◦ set $v_i^{n+1} = v_i'$, $v_j^{n+1} = v_j'$
                else
                    ◦ set $v_i^{n+1} = v_i^n$, $v_j^{n+1} = v_j^n$
          ◦ set $v_i^{n+1} = v_i^n$ for the $N - 2N_c$ particles that have not been selected
  end for
```

The new velocities v_i' and v_j' are computed using (4.3) and (4.4).

The upper bound $\bar{\sigma}$ should be chosen as small as possible, to avoid inefficient rejection, and it should be computed fast.

An upper bound can be derived taking $\bar{\sigma}$ as

$$\sigma_{\max} = \max_{v_i, v_j} \sigma(|v_i - v_j|). \tag{4.5}$$

Classically, since this computation would require $O(N^2)$ operations, instead of (4.5) it is preferable to use an upper bound of $\sigma_{\max}$ given by

$$\sigma_{\max} \le \bar{\sigma} = \sigma(2\Delta v), \quad \Delta v = \max_i |v_i - \bar{v}|, \quad \bar{v} = \sum_i v_i/N. \tag{4.6}$$

For the general collisional kernel, the algorithm is slightly modified by introducing the angular dependence. In this case, the collision is always performed, and the new velocities are extracted according to the differential cross section. Then a rejection is used to decide whether the collision is accepted.

Remark 4.1. Note that if the time step is small enough, only a small fraction of particles, let us say N_c, will collide. The computational cost of the collisions is therefore $O(N_c)$. On the other hand, the cost of the computation of the upper bound $\bar{\sigma}$ is $O(N)$, which may be much larger than $O(N_c)$. A possible way to overcome this difficulty is to update at each time step the value of the upper bound $\bar{\sigma}$ only if it increases. This may be done as follows. During the computation of the collision between particles i and j, let $\tilde{v}_i$ and $\tilde{v}_j$ denote the new particle velocities. Then the quantity Δv is updated according to

$$\Delta v = \max(\Delta v, |\tilde{v}_i - \bar{v}|, |\tilde{v}_j - \bar{v}|). \tag{4.7}$$

At the end of the collision loop, the upper bound on the cross section is computed as

$$\bar{\sigma} = \sigma(2\Delta v).$$

In space nonhomogeneous calculations, assuming that there are several collisional time steps during a convection time step, the bound can be computed according to (4.6) the first time step and then updated as described above.

4.2. TRMC methods. The first order TRMC algorithm is based on the TR schemes

$$f^{n+1} = A_0 f^n + A_1 f_1 + A_2 M. \tag{4.8}$$

The probabilistic interpretation of the above equation is the following: a particle extracted from f^n will not collide with probability A_0; it will collide with another particle extracted from f^n with probability A_1; or it will be replaced by a particle sampled from a Maxwellian with probability A_2.

In this formulation, the probabilistic interpretation holds uniformly in $\mu\Delta t/\epsilon$, at variance with standard DSMC, which requires $\mu\Delta t/\epsilon < 1$. Furthermore, as $\mu\Delta t/\epsilon \to \infty$, the distribution at time $n+1$ is sampled from a Maxwellian. In this limit, the density f^{n+1} relaxes immediately to its equilibrium distribution. In a space nonhomogeneous case, this would be equivalent to the particle method for Euler equations proposed by Pullin [18].

The Monte Carlo schemes described above are conservative in the mean. It is possible to make it exactly conservative by selecting collision pairs uniformly, rather than individual particles, and by using a suitable algorithm for sampling a set of particles with prescribed momentum and energy from a Maxwellian [19].

The conservative version of the methods can be formalized in the following algorithm.

Algorithm 4.3 (first order TRMC for VHS molecules).

```
• compute the initial velocity of the particles, $\{v_i^0, i = 1, \dots, N\}$,
  by sampling them from the initial density $f_0(v)$
• for $n = 1$ to $n_{\mathrm{TOT}}$
      given $\{v_i^n, i = 1, \dots, N\}$,
          ○ compute an upper bound $\bar{\sigma}$ of the cross section
          ○ set $\tau = 1 - \exp(-\rho\bar{\sigma}\Delta t/\epsilon)$
          ○ compute $A_1(\tau)$, $A_2(\tau)$
          ○ set $N_c = \mathrm{Iround}(N A_1/2)$
          ○ perform $N_c$ dummy collisions, as in Algorithm 4.2
          ○ set $N_M = \mathrm{Iround}(N A_2)$
          ○ select $N_M$ particles among those that have not collided,
            and compute their mean momentum and energy
          ○ sample $N_M$ particles from the Maxwellian with the above
            momentum and energy, and replace the $N_M$ selected particles
            with the sampled ones
          ○ set $v_i^{n+1} = v_i^n$ for all the $N - 2N_c - N_M$ particles that
            have not been selected
  end for
```

A second order Monte Carlo scheme is obtained by the TR scheme

$$f^{n+1} = A_0 f^n + A_1 f_1 + A_2 f_2 + A_3 M \tag{4.9}$$

with

$$f_1 = \frac{P(f^n, f^n)}{\mu}, \quad f_2 = \frac{P(f^n, f_1)}{\mu}.$$

Given N particles distributed according to f^n, the probabilistic interpretation of scheme (4.9) is the following: NA_0 particles will not collide; NA_1 will be sampled from f_1 (as in the first order scheme); NA_2 will be sampled from f_2; i.e., $NA_2/2$ particles sampled from f^n will undergo dummy collisions with $NA_2/2$ particles sampled from f_1; and NA_3 particles will be sampled from a Maxwellian.

Once again, the methods can be made conservative using the same techniques adopted in the first order scheme. The various steps of the method can be summarized in the following algorithm.

Algorithm 4.4 (second order TRMC for VHS molecules).

```
• compute the initial velocity of the particles, $\{v_i^0, i = 1, \ldots, N\}$,
  by sampling them from the initial density $f_0(v)$
• for $n = 1$ to $n_{\mathrm{TOT}}$
      given $\{v_i^n, i = 1, \ldots, N\}$,
          ◦ compute an upper bound $\bar\sigma$ of the cross section
          ◦ set $\tau = 1 - \exp(-\rho\bar\sigma\Delta t/\epsilon)$
          ◦ compute $A_1(\tau)$, $A_2(\tau)$,$A_3(\tau)$
          ◦ set $N_1 = \mathrm{Iround}(NA_1/2)$ , $N_2 = \mathrm{Iround}(NA_2/4)$
          ◦ select $N_1 + N_2$ dummy collision pairs $(i, j)$ uniformly
            among all possible pairs
          ◦ for $N_1$ pairs
              ◦ compute the relative cross section $\sigma_{ij} = \sigma(|v_i - v_j|)$
              ◦ if $\bar\sigma\,\mathrm{Rand} < \sigma_{ij}$
                    ◦ perform the collision between $i$ and $j$, and
                      compute $v_i'$ and $v_j'$ according to the collisional law
                    ◦ set $v_i^{n+1} = v_i'$, $v_j^{n+1} = v_j'$
          ◦ for $N_2$ pairs
              ◦ compute the relative cross section $\sigma_{ij} = \sigma(|v_i - v_j|)$
              ◦ if $\bar\sigma\,\mathrm{Rand} < \sigma_{ij}$
                    ◦ perform the collision between $i$ and $j$,
                      compute $v_i'$ and $v_j'$ according to the collisional law and
                      store them
          ◦ select $2N_2$ particles from $f^n$
          ◦ perform the collision of these selected particles with the
            second set of $2N_2$ particles that have collided once
          ◦ update the velocity of the $4N_2$ particles with the outcome
            of the $2N_2$ collisions (of particles that have never collided
            before with particles that collided once)
          ◦ set $N_M = \mathrm{Iround}(NA_3)$
          ◦ replace $N_M$ particles with samples from Maxwellian, as in Algorithm 4.3
          ◦ set $v_i^{n+1} = v_i^n$ for all the $N - 2N_1 - 4N_2 - N_M$ particles that
            have not been selected
  end for
```

Similarly, higher order TRMC methods can be constructed. For example, a third order scheme is obtained from

$$f^{n+1} = A_0 f^n + A_1 f_1 + A_2 f_2 + A_3 f_3 + A_4 M \tag{4.10}$$

with

$$f_1 = \frac{P(f^n, f^n)}{\mu}, \quad f_2 = \frac{P(f^n, f_1)}{\mu}, \quad f_3 = \frac{1}{3\mu}[2P(f^n, f_2) + P(f_1, f_1)].$$

We omit for brevity the details of the resulting Monte Carlo algorithm.

We remark here that the TR scheme is a direct consequence of the time discretization of the Boltzmann equation. In this respect, the choice of a particular Monte Carlo scheme used to model the collision is not crucial. Here we considered a commonly used Monte Carlo scheme. If a different technique is used to model the collision, then the same technique could be used in conjunction with the TR discretization.

5. Numerical results.

In this section, we present some numerical results for the new TRMC method and compare them with the standard DSMC method.

5.1. Space homogeneous results.

First we consider homogeneous test problems for the Kac equation [12] and for the Boltzmann equation for Maxwell molecules. Exact solutions are known in these special situations, and so we can compare the accuracy of the different schemes. In our tests, we have performed a single run with a number of particles sufficiently large to control the effects of the fluctuations. We express the results as a function of the scaled time variable t/ϵ, which we denote again by t, in order to simplify the notations.

5.1.1. Test I: Kac equation.

In the first test, we compare the different Monte Carlo methods using an exact solution of the Kac equation [12]. This test is used to check the accuracy of the methods using different time steps.

The Kac equation is a simplified one-dimensional model of the Boltzmann equation for Maxwell molecules characterized by

$$Q(f,f)(v) = \frac{1}{2\pi}\int_0^{2\pi}\int_{\mathbb{R}}[f(v')f(v'_*) - f(v)f(v_*)]\,dv_*\,d\theta, \tag{5.1}$$

where

$$v' = v\cos\theta - v_*\sin\theta, \qquad v'_* = v\sin\theta + v_*\cos\theta. \tag{5.2}$$

An exact solution for this equation is [4, 13]

$$f(v,t) = \frac{1}{2C^{3/2}}\left[\frac{3}{2}(C-1) + (3-C)\frac{v^2}{C^2}\right]\exp\left(-v^2/C\right)$$

with $C(t) = 3 - 2\exp(-\sqrt{\pi}t/16)$.

The previous TR schemes can be applied directly to (5.1), taking

$$P(f,f) = \frac{1}{2\pi}\int_0^{2\pi}\int_{\mathbb{R}^3} f(v')f(v'_*)\,dv_*\,d\theta, \tag{5.3}$$

and $\mu = \rho$.

The density function f has been reconstructed on a regular grid, by convolving the particle distribution by a suitable mollifier [8]

$$f(V_I) = \frac{1}{N}\sum_{j=1}^{N} W_H(V_I - v_j), \tag{5.4}$$

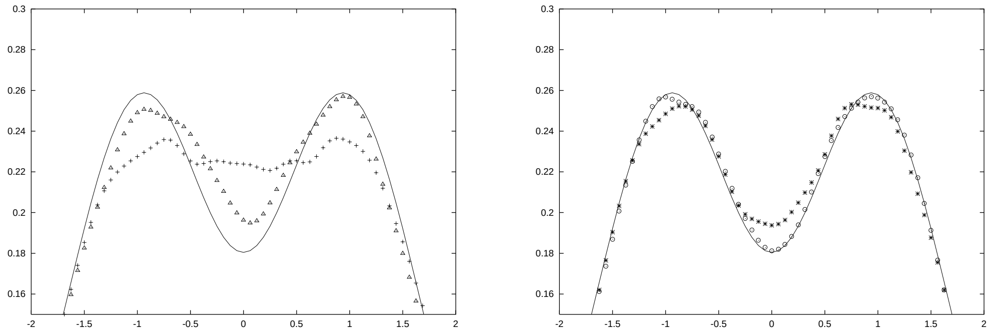


FIG. 1. *Test* I *(Kac equation). Details of the distribution function at time* $t = 2.0$ *for* $\Delta t = 1.0$. *Exact (line), DSMC (+), first order TRMC* ($\triangle$)*, second order TRMC* ($*$)*, third order TRMC* ($\circ$)*.*

where

$$\{V_I = V_{\min} + I\Delta V, I = 1, \dots, N_g\}.$$

The smoothing function W_{H} is given by

$$W_H(x) = \frac{1}{H} W\left(\frac{x}{H}\right), \quad W(x) = \left\{ \begin{array}{cl} 3/4 - x^2 & \text{if } |x| \le 0.5, \\ (x - 3/2)^2/2 & \text{if } 0.5 < |x| \le 1.5, \\ 0 & \text{otherwise.} \end{array} \right.$$

The value $H = 0.2$ has been selected as a good compromise between fluctuations and resolution. The simulations are performed for $t \in [0, 8]$ by starting with $N = 5 \times 10^4$ particles.

In Figure 1 we present the numerical results obtained at time $t = 2$ for the different schemes with the same time step $\Delta t = 1.0$, which represents the upper bound for the stability of DSMC.

It is evident that all TRMC schemes gives a better representation of the solution, especially near the local extrema.

Figure 2 shows the results of the L^2-norm of the error for the same test. At the bottom of the same figure, we show how it is possible to use larger time steps in TRMC schemes with respect to DSMC without any deterioration of the accuracy.

5.1.2. Test II: Maxwell molecules. Next we perform the same kind of accuracy test in the case of the Boltzmann equation for Maxwell molecules in the two-dimensional case. An exact solution to this equation [4, 13] is given by

$$f(v, t) = \frac{1}{2\pi C}\left[1 - \frac{(1 - C)}{C}\left(1 - \frac{v^2}{2C}\right)\right] \exp\left(-\frac{v^2}{2C}\right), \tag{5.5}$$

where $C(t) = 1 - (1/2)\exp(-t/8)$.

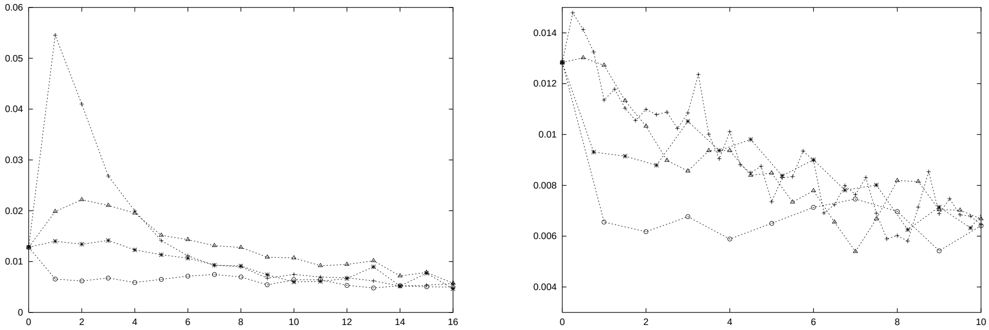


FIG. 2. *Test* I *(Kac equation).* L^2 *norm of the error vs time. DSMC (+), first order TRMC* $(\triangle)$*, second order TRMC* $(*)$*, third order TRMC* $(\circ)$*. Top: Time step* $\Delta t = 1.0$*. Bottom: DSMC with* $\Delta t = 0.25$*, first order TRMC with* $\Delta t = 0.5$*, second order TRMC with* $\Delta t = 0.75$*, third order TRMC*3 *with* $\Delta t = 1.0$*.*

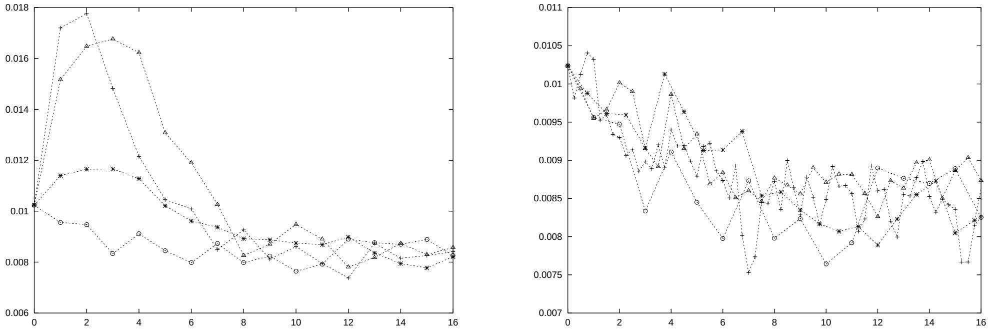


FIG. 3. *Test* II *(Maxwell molecules).* L^2 *norm of the error vs time. DSMC (+), first order TRMC* $(\triangle)$*, second order TRMC* $(*)$*, third order TRMC* $(\circ)$*. Top: Time step* $\Delta t = 1.0$*. Bottom: DSMC with* $\Delta t = 0.25$*, first order TRMC with* $\Delta t = 0.5$*, second order TRMC with* $\Delta t = 0.75$*, third order TRMC*3 *with* $\Delta t = 1.0$*.*

The comparison with the exact solution is obtained by reconstructing the function on a regular grid of spacing $\Delta v = 0.25$ by the "weighted area rule" [11].

All the simulations have been performed for $t \in [0, 16]$ by starting with $N = 10^5$ particles.

In Figure 3 we show the L^2 norm of the error in time for both DSMC and TRMC schemes. In the first picture, we report the results obtained with the same time step $\Delta t = 1.0$. The results confirm the gain of accuracy of the TRMC methods on the transient time scale. Next, the results obtained with different time steps, chosen in such a way that the different errors are roughly the same, are also reported.

5.2. Shock wave profiles. The last test problem deals with the numerical solution of the space nonhomogeneous Boltzmann equation for hard sphere molecules (VHS, for $\alpha = 1$) with $C_\alpha = 1$. We present some numerical results for one-dimensional stationary shock profiles. In particular, we have computed the structure of the shock for different Knudsen numbers, from the rarefied regime up to the fluid limit. In all our numerical tests, the gas is initially at the upstream equilibrium state in the left

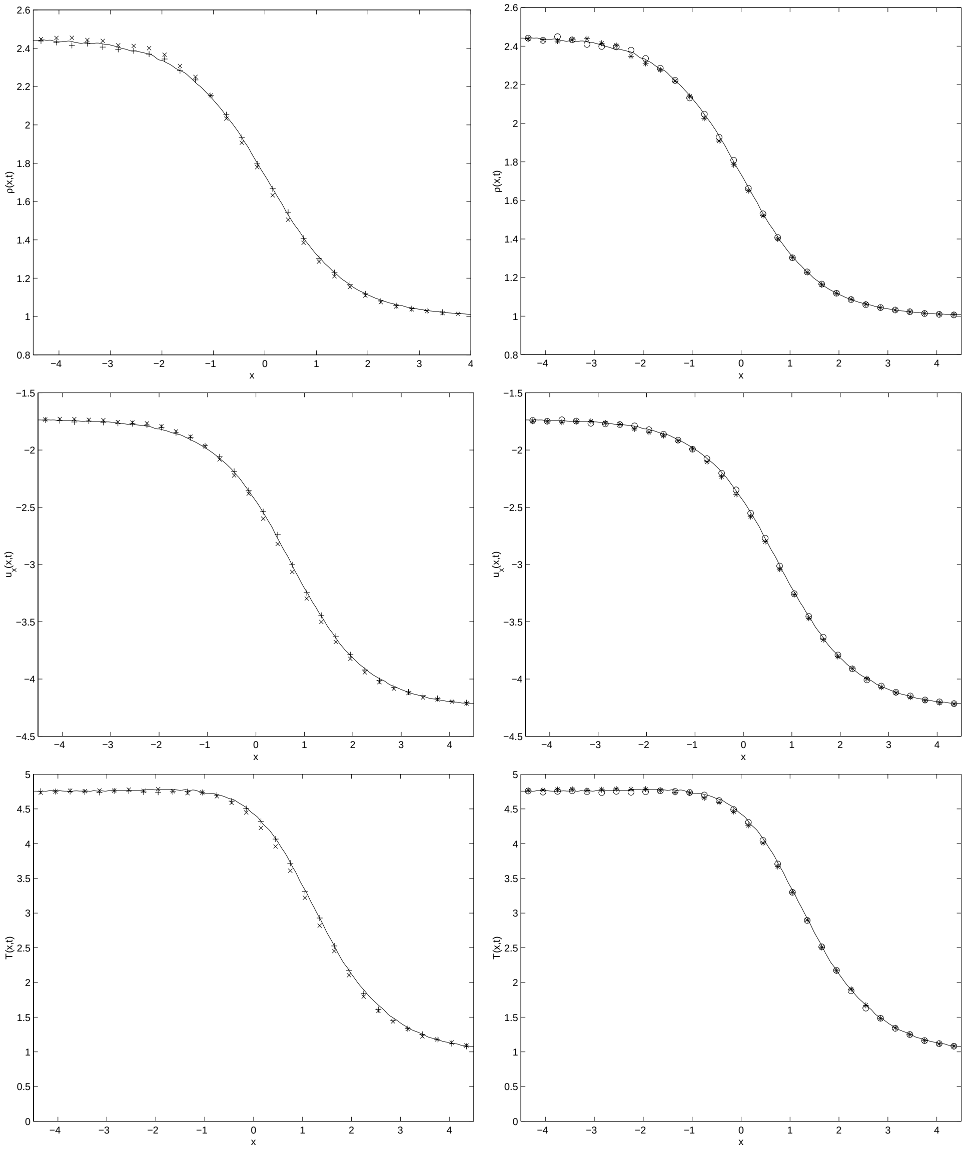


FIG. 4. *Shock wave profiles (rarefied regime): DSMC(+) and first order TRMC (×) (left column), second order (∗) and third order (○) TRMC (right column) for* $\epsilon = 1.0$ *and* $\Delta t = 0.025$. *From top to bottom:* ρ, u, T. *The line is the reference solution.*

half-space and in the downstream equilibrium state in the right half-space. The upstream state is determined from the downstream state using the Rankine–Hugoniot relations [21].

In the present calculations, the downstream state is characterized by

$$\rho = 1.0, \quad T = 1.0, \quad M = 3.0,$$

where M is the Mach number of the shock. The downstream mean velocity is then

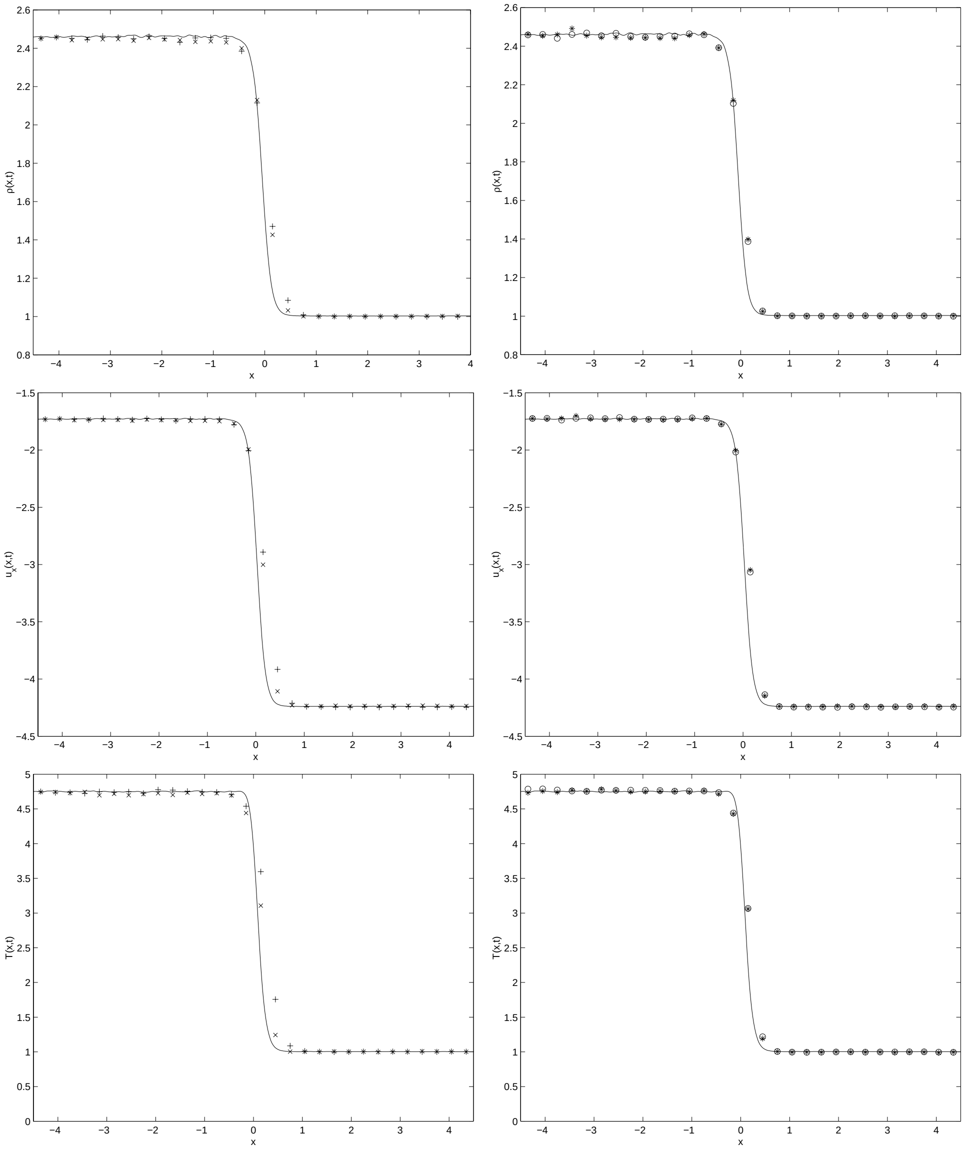


FIG. 5. *Shock wave profiles (intermediate regime): DSMC(+) and first order TRMC (×) (left column), second order (∗) and third order (○) TRMC (right column) for $\epsilon = 1.0$ and $\Delta t = 0.0025$ for DSMC, $\Delta t = 0.025$ for TRMC. From top to bottom: ρ, u, T. The line is the reference solution.*

given by

$$u_x = -M\sqrt{(\gamma T)}, \quad u_y = 0,$$

with $\gamma = 2$ since we have considered a two-dimensional monatomic gas in velocity space.

The infinite physical space is truncated to the finite region $[-7.5, 7.5]$. A suitable stabilizing technique [3] has been used to keep the number of particles constant during the time evolution. We report the result obtained with the different schemes using

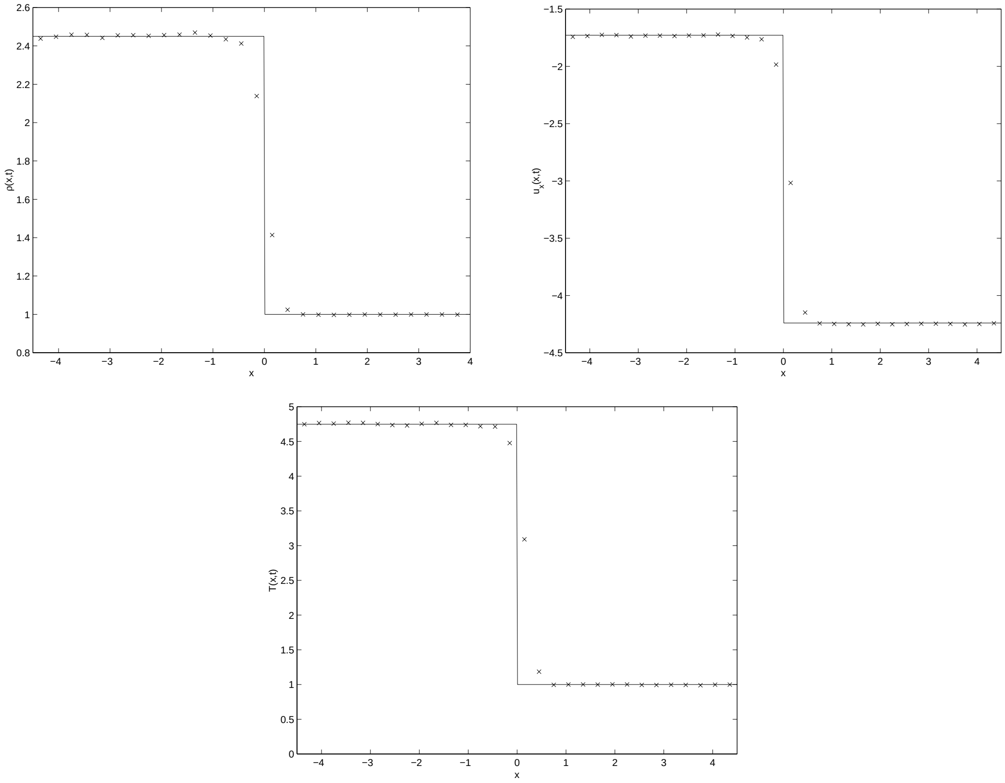


FIG. 6. *Shock wave profiles (fluid regime): First order TRMC* ($\times$) *for* $\epsilon = 10^{-6}$ *and* $\Delta t = 0.025$. *From top to bottom:* ρ, u, T.

50 space cells and 500 particles in each downstream cell. Since we are computing a stationary solution after a fixed initial time, we can strongly improve the accuracy of the Monte Carlo solution by averaging in time the solution itself. To this aim, we started accumulating statistics at time $t = 5$, and we average up to $t = 20$, which corresponds approximatively to 1500 time steps. The reference solution in the rarefied and intermediate regime is obtained using the DSMC method with 200 space cells and 500 particles in each downstream cell and averaging over approximatively 8000 time steps. All the results are reported on the space interval $[-4.5, 4.5]$.

In Figure 4 we plot the result obtained in the rarefied regime ($\epsilon = 1$) using the different Monte Carlo methods. The computational cost of the methods is comparable, since here we are far from the fluid limit. As expected, the results show that, essentially, the TRMC methods and the DSMC method are equivalent and provide a good description of the rarefied shock.

Next we consider the intermediate regime ($\epsilon = 0.1$). Two different time steps have been used for DSMC, a convection time step Δt and a collision time step $\Delta t_{\text{coll}} = \Delta t/10$. The collision time step for TRMC is equal to the convection time step. The results show that, despite the larger time step, the accuracy of the solution obtained with TRMC schemes is essentially the same of DSMC (see Figure 5).

Finally, we give the result of the computations close to the Euler limit ($\epsilon = 10^{-6}$). The profiles obtained with the TRMC methods are reported in Figure 6. Due to the small Knudsen number, the DSMC method is unusable in practice in this test. Note that the TRMC methods are all equivalent to Pullin's method. Thus we have plotted only one approximate solution.

Conclusion and future works. In this paper, a new family of schemes is proposed for the numerical solution of the Boltzmann equation.

The schemes have been tested on several problems for the Boltzmann equation, and they provide good results compared with standard Monte Carlo methods.

There is still a lot to do to improve the accuracy and robustness of the schemes. In particular, the optimal choice of the coefficients A_k, $k = 1, \ldots, m$, is still an open problem. Several possibilities are presently under consideration. The first one corresponds to choosing the coefficients in such a way that some moments of the distribution function for Maxwellian molecules are computed exactly. It is well known, in fact, that exact closure for the moment equations is possible in the case of Maxwellian molecules [23]. The same coefficients could then be used for VHS molecules.

A second alternative is to compute several terms of the Wild sum expansion. If the weights of the expansion decay fast enough, the computation can be done very efficiently by a recursive algorithm, even for large values of k. The recursive tree can eventually be truncated and the corresponding value of f_k substituted by sampling from a Maxwellian. Such an approach is very promising and is considered in [17].

REFERENCES

[1] H. Babovsky, *On a simulation scheme for the Boltzmann equation*, Math. Methods Appl. Sci., 8 (1986), pp. 223–233.

[2] H. Babovsky and R. Illner, *A convergence proof for Nanbu's simulation method for the full Boltzmann equation*, SIAM J. Numer. Anal., 26 (1989), pp. 45–65.

[3] G. A. Bird, *Molecular Gas Dynamics*, Oxford University Press, London, 1976.

[4] A. V. Bobylev, *Exact solutions of the Boltzmann equation*, Dokl. Akad. Nauk SSSR, 225 (1975), pp. 1296–1299 (in Russian).

[5] J. F. Bourgat, P. LeTallec, B. Perthame, and Y. Qiu, *Coupling Boltzmann and Euler equations without overlapping*, in Domain Decomposition Methods in Science and Engineering, Contemp. Math. 157, AMS, Providence, RI, 1994, pp. 377–398.

[6] E. A. Carlen, M. C. Carvalho, and E. Gabetta, *Central limit theorem for Maxwellian molecules and truncation of the Wild expansion*, Comm. Pure Appl. Math., 53 (2000), pp. 370–397.

[7] C. Cercignani, *The Boltzmann Equation and Its Applications*, Springer-Verlag, New York, 1988.

[8] L. Desvillettes and R. E. Peralta Herrera, *A vectorizable simulation method for the Boltzmann equation*, RAIRO Modél. Math. Anal. Numér., 28 (1994), pp. 745–760.

[9] E. Gabetta, L. Pareschi, and G. Toscani, *Relaxation schemes for nonlinear kinetic equations*, SIAM J. Numer. Anal., 34 (1997), pp. 2168–2194.

[10] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations* II: *Stiff and Differential-Algebraic Problems*, Springer-Verlag, New York, 1987.

[11] R. W. Hockney and J. W. Eastwood, *Computer Simulation Using Particles*, McGraw-Hill, New York, 1981.

[12] M. Kac, *Probability and Related Topics in Physical Sciences*, Lectures in Appl. Math., Interscience Publishers, London, New York, 1959.

[13] M. Krook and T. T. Wu, *Formation of Maxwellian tails*, Phys. Rev. Lett., 36 (1976), pp. 1107–1109.

[14] K. Nanbu, *Direct simulation scheme derived from the Boltzmann equation*, J. Phys. Soc. Japan, 49 (1980), pp. 2042–2049.

[15] L. Pareschi and R. E. Caflisch, *Implicit Monte Carlo methods for rarefied gas dynamics* I: *The space homogeneous case*, J. Comput. Phys., 154 (1999), pp. 90–116.

[16] L. Pareschi and G. Russo, *Asymptotic preserving Monte Carlo methods for the Boltzmann equation*, Transport Theory Statist. Phys., 29 (2000), pp. 415–430.

[17] L. Pareschi and B. Wennberg, *A recursive Monte Carlo method for the Boltzmann equation in the Maxwellian case*, Monte Carlo Methods Appl., 7 (2001), pp. 349–358.

[18] D. I. Pullin, *Direct simulation methods for compressible inviscid ideal gas flow*, J. Comput.

Phys., 34 (1980), pp. 231–244.

[19] D. I. Pullin, *Generation of normal variates with given sample*, J. Statist. Comput. Simulation, 9 (1979), pp. 303–309.

[20] G. Russo and R. E. Caflisch, *Implicit methods for kinetic equations*, in Rarefied Gas Dynamics: Theory and Simulations, Progress in Aeronautics and Astronautics 159, AIAA, New York, pp. 344–352.

[21] G. B. Whitham, *Linear and Nonlinear Waves*, Wiley-Interscience, New York, 1974.

[22] E. Wild, *On Boltzmann's equation in the kinetic theory of gases*, Proc. Cambridge Philos. Soc., 47 (1951), pp. 602–609.

[23] C. Truesdell and R. G. Muncaster, *Fundamentals of Maxwell Kinetic Theory of a Simple Monatomic Gas*, Academic Press, New York, 1980.

SIAM J. SCI. COMPUT.
Vol. 23, No. 4, pp. 1274–1290

CONDITIONS OF NONDEGENERACY OF THREE-DIMENSIONAL CELLS. A FORMULA OF A VOLUME OF CELLS[*]

OLGA V. USHAKOVA[†]

Abstract. Numerical solutions of partial differential equations in three dimensions often use hexahedral cells composing a computational grid. After the grid is constructed, it is first of all verified whether the grid is unfolded or not. For such a test in this paper, conditions of nondegeneracy are found for hexahedral cells which are given by eight corner points and generated by the trilinear map from a unit cube to a region defined by these points. The conditions include necessary conditions and two sets of sufficient conditions. They are found as conditions of positivity of the Jacobian of the trilinear map. Thus, the conditions which guarantee the invertibility of the trilinear map from a unit cube to a hexahedron are given. How general the nondegeneracy conditions are is shown by a numerical experiment. Formulas of the Jacobian of the trilinear map are obtained, and following from them, as a separate result, a formula of a volume of a cell is obtained.

Key words. grid generation, hexahedral cells, positivity of the Jacobian of the trilinear map, conditions of nondegeneracy of cells, formula of a volume of a cell

AMS subject classifications. 65M50, 65M06, 65M60

PII. S1064827500380702

Introduction. For grid generation methods, good grid quality tests are needed. In the two-dimensional case [1, 2], the tests are suggested in [3]. Among them, in the first place, there is the criterion of nondegeneracy of cells since this criterion is a common requirement for a computational grid and a major objective of grid generation algorithms. Therefore, both a mathematician developing a grid generation method and a practicing engineer must care about this. Such tests are very important, especially, in three dimensions when a visualization of a grid is complicated. The purpose of the present research was to find this criterion for a grid consisting of hexahedrons generated by a trilinear map of a unit cube. Hexahedral cells are often used [4] by finite difference and finite element methods. As a result, the conditions of nondegeneracy of cells were found. The obtained conditions were briefly presented in [5]. In this paper, conditions of nondegeneracy and their deduction are given in detail.

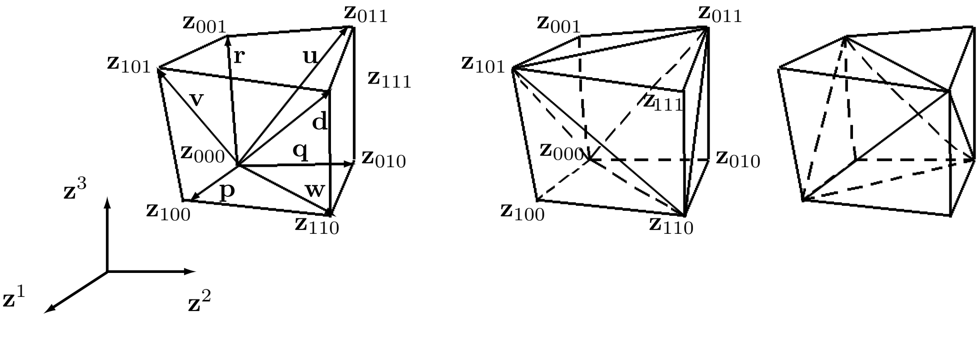
Earlier, the problem of nondegeneracy of hexahedral cells (the problem of invertibility of the trilinear map) was investigated in a number of works; see, for example, [6, 7, 8].

In section 1, the criterion of nondegeneracy of cells is formulated in terms of positivity of the Jacobian of the transformation used for generation of cells. Formulas of the Jacobian are obtained in section 2. In section 3, both necessary and sufficient conditions of positivity of the Jacobian are found. The conditions are formulated as restrictions on the volumes of tetrahedrons with the vertices located at the corners

[*] Received by the editors November 9, 2000; accepted for publication (in revised form) February 21, 2001; published electronically December 4, 2001. A preliminary version of this paper appeared as *Numerical grid generation in computational field simulations*, in Proceedings of the 7th International Conference, B. K. Soni, J. Haeuser, J. F. Thompson, and P. Eiseman, eds., 2000, International Society of Grid Generation (ISGG), Mississippi State, MS, pp. 659–668. This work was supported by the Russian Foundation for Basic Research projects 99-01-00326 and 00-15-96042.

http://www.siam.org/journals/sisc/23-4/38070.html

[†] Institute of Mathematics and Mechanics, Ural Branch of the Russian Academy of Sciences, S. Kovalevskaya Street 16, Ekatherinburg, 620219 Russia (uov@imm.uran.ru).



(a) (b)

FIG. 1.1. *Three-dimensional cells:* (a) *hexahedron;* (b) *dodecahedrons.*

of hexahedral cells. (The faces of tetrahedrons are planar.) The formulas of the Jacobian are written in the form of polynomials with the coefficients proportional to the volumes of tetrahedrons. The results of computations for hexahedrons randomly selected by the computer are presented. The results give the success rate of the obtained conditions as well as examples of hexahedral cells, the nondegeneracy of which is verified by means of these conditions.

In section 4, a new formula of a volume of a cell is derived. Formulas of a volume of a cell published in [4, 9] are rather complex and demand a large amount of computations. Efficient volume computation and another formula of a cell volume were suggested in [10]. The formula obtained in this work is similar to [10] but requires computation of volumes of ten tetrahedrons.

1. Generation of ruled hexahedral cells. Let eight points $\mathbf{z}_{i_1i_2i_3} = (z^1_{i_1i_2i_3}, z^2_{i_1\text{“}i_2i_3}, z^3_{i_1i_2i_3})$, $i_1, i_2, i_3 = 0, 1$ be given. The trilinear map

$$\begin{aligned} \mathbf{z}(\mathbf{y}) = {} & \mathbf{a}_{000} + \mathbf{a}_{100}y^1 + \mathbf{a}_{010}y^2 + \mathbf{a}_{001}y^3 \\ & + \mathbf{a}_{110}y^1y^2 + \mathbf{a}_{101}y^1y^3 + \mathbf{a}_{011}y^2y^3 + \mathbf{a}_{111}y^1y^2y^3 \end{aligned} \tag{1.1}$$

of the unit cube $P = \{\mathbf{y} = (y^1, y^2, y^3) : 0 \le y^l \le 1,\ l = 1, 2, 3\}$ defines the ruled cell with the corners $\mathbf{z}_{i_1i_2i_3} = \mathbf{z}(i_1, i_2, i_3)$, $i_1, i_2, i_3 = 0, 1$. The vectors $\mathbf{a}_{i_1i_2i_3}$ are found from the following relations:

$$\begin{aligned} &\mathbf{a}_{000} = \mathbf{z}_{000}, && \mathbf{a}_{111} = \mathbf{z}_{111} - \mathbf{z}_{110} - \mathbf{z}_{101} - \mathbf{z}_{011} + \mathbf{z}_{100} + \mathbf{z}_{010} + \mathbf{z}_{001} - \mathbf{z}_{000}, \\ &\mathbf{a}_{001} = \mathbf{z}_{001} - \mathbf{z}_{000}, && \mathbf{a}_{011} = \mathbf{z}_{011} - \mathbf{z}_{010} - \mathbf{z}_{001} + \mathbf{z}_{000}, \\ &\mathbf{a}_{010} = \mathbf{z}_{010} - \mathbf{z}_{000}, && \mathbf{a}_{101} = \mathbf{z}_{101} - \mathbf{z}_{100} - \mathbf{z}_{001} + \mathbf{z}_{000}, \\ &\mathbf{a}_{100} = \mathbf{z}_{100} - \mathbf{z}_{000}, && \mathbf{a}_{110} = \mathbf{z}_{110} - \mathbf{z}_{100} - \mathbf{z}_{010} + \mathbf{z}_{000}. \end{aligned} \tag{1.2}$$

The concept of the ruled cell and techniques of generation of grids by such cells can be found, for example, in [4, 9]. In two dimensions, the ruled cell is a quadrilateral. If all eight corners of the cell are different, the edges of the cube are transformed by the trilinear map to straight line edges of the hexahedron, and the faces of the cube are transformed to ruled surfaces of the second order or planes (Figures 1.1, 3.1, and 3.2).

Many of the grid properties can be controlled by the Jacobian matrix of the map used for generating a grid or a cell [11]. The criterion of nondegeneracy of a cell is usually formulated in terms of positivity of the Jacobian (the determinant of the Jacobian matrix) of such a map [7, 8, 12, 13]. A grid element or a cell is said to be nondegenerate (noninverted, valid, unfolded, nonsingular) if the Jacobian is positive. In the general case, the nonzero Jacobian does not globally guarantee a one-to-one correspondence of the map ([13, Example 1.3.4, p. 9]). It guarantees only locally. In the case of a trilinear map (smooth map) from a cube (a domain), the nonzero Jacobian in the whole cube (including boundary) globally guarantees a one-to-one correspondence of the map ([8, Theorem 1, p. 8-4]). Positivity of the Jacobian provides the same orientation of edges of a cell that a unit cube has. Nondegeneracy of the joint grid is attained by the nondegeneracy of all its cells [8].

The Jacobian of map (1.1),

$$J(y^1, y^2, y^3) = \frac{\partial(z^1, z^2, z^3)}{\partial(y^1, y^2, y^3)} = \det\left(\frac{\partial z^i}{\partial y^j}\right)_{i=1,2,3,\, j=1,2,3},$$

is equal to the triple scalar product

$$J = \left[\frac{\partial \mathbf{z}}{\partial y^1}, \frac{\partial \mathbf{z}}{\partial y^2}, \frac{\partial \mathbf{z}}{\partial y^3}\right], \tag{1.3}$$

where

$$\begin{aligned}
\frac{\partial \mathbf{z}}{\partial y^1} &= \mathbf{a}_{100} + \mathbf{a}_{110} y^2 + \mathbf{a}_{101} y^3 + \mathbf{a}_{111} y^2 y^3,\\
\frac{\partial \mathbf{z}}{\partial y^2} &= \mathbf{a}_{010} + \mathbf{a}_{110} y^1 + \mathbf{a}_{011} y^3 + \mathbf{a}_{111} y^1 y^3,\\
\frac{\partial \mathbf{z}}{\partial y^3} &= \mathbf{a}_{001} + \mathbf{a}_{101} y^1 + \mathbf{a}_{011} y^2 + \mathbf{a}_{111} y^1 y^2.
\end{aligned} \tag{1.4}$$

The properties of the Jacobian are studied in [6, 7, 8, 12]. In two dimensions, a bilinear map of a unit square is considered. Its Jacobian is a linear function. If the Jacobian is positive at the corners of the square, then due to linearity the Jacobian will be positive everywhere in the square. The converse is also true. In two dimensions, if the Jacobian is positive, then the cell is convex, and the condition of nondegeneracy of cells is equivalent to the condition of convexity of cells [8, 12, 13].

A three-dimensional case is much more complicated. Since the faces of the cell can be nonplanar, the cell can be nonconvex. In [6] it is implied that the Jacobian in three dimensions is positive everywhere if and only if the Jacobian is positive at the corners of a cube. In [7] this statement is shown to be false. It is also demonstrated that for the positivity of the Jacobian in the interior of the cube the positivity of the Jacobian on the edges of the cube is not sufficient. The Jacobian in [7] is written in terms of polynomials, the coefficients of which are the values of the Jacobian on the edges.

2. The Jacobian of the trilinear map.

For each point $\mathbf{z}_{i_1 i_2 i_3}$, $i_1, i_2, i_3 = 0, 1$, consider the vectors (Figure 1.1(a))

$$\begin{aligned}
\mathbf{p}_{i_1 i_2 i_3} &= \mathbf{z}_{\bar{i}_1 i_2 i_3} - \mathbf{z}_{i_1 i_2 i_3}, & \mathbf{q}_{i_1 i_2 i_3} &= \mathbf{z}_{i_1 \bar{i}_2 i_3} - \mathbf{z}_{i_1 i_2 i_3},\\
\mathbf{r}_{i_1 i_2 i_3} &= \mathbf{z}_{i_1 i_2 \bar{i}_3} - \mathbf{z}_{i_1 i_2 i_3}, & \mathbf{u}_{i_1 i_2 i_3} &= \mathbf{z}_{i_1 \bar{i}_2 \bar{i}_3} - \mathbf{z}_{i_1 i_2 i_3},\\
\mathbf{v}_{i_1 i_2 i_3} &= \mathbf{z}_{\bar{i}_1 i_2 \bar{i}_3} - \mathbf{z}_{i_1 i_2 i_3}, & \mathbf{w}_{i_1 i_2 i_3} &= \mathbf{z}_{\bar{i}_1 \bar{i}_2 i_3} - \mathbf{z}_{i_1 i_2 i_3},\\
\mathbf{d}_{i_1 i_2 i_3} &= \mathbf{z}_{\bar{i}_1 \bar{i}_2 \bar{i}_3} - \mathbf{z}_{i_1 i_2 i_3}.
\end{aligned} \tag{2.1}$$

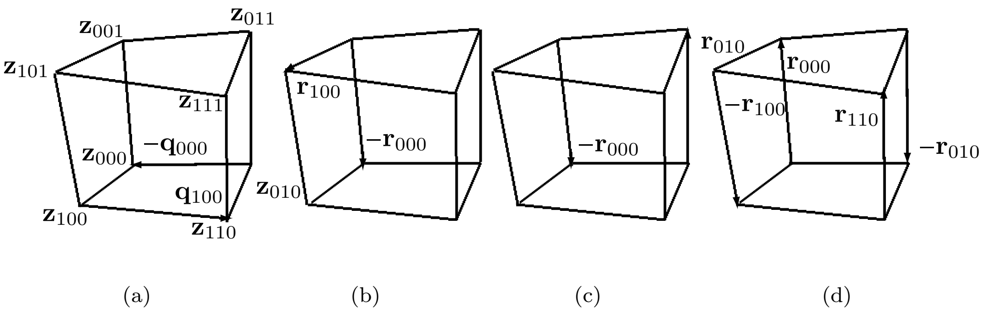


FIG. 2.1. *Vectors* $\mathbf{a}_{i_1 i_2 i_3}$: (a) $\mathbf{a}_{110}$; (b) $\mathbf{a}_{101}$; (c) $\mathbf{a}_{011}$; (d) $\mathbf{a}_{111}$.

Here and hereafter, $\bar{0} = 1$, $\bar{1} = 0$. The vectors $\mathbf{p}$, $\mathbf{q}$, $\mathbf{r}$ are directed along the edges, the vectors $\mathbf{u}$, $\mathbf{v}$, $\mathbf{w}$ play the role of "diagonals" of the faces, and vectors $\mathbf{d}$ are "inner diagonals" of the cell.

The vectors (2.1) have the properties

$$\begin{aligned} \mathbf{p}_{0 i_2 i_3} &= (-1)^{i_1} \mathbf{p}_{i_1 i_2 i_3}, \\ \mathbf{q}_{i_1 0 i_3} &= (-1)^{i_2} \mathbf{q}_{i_1 i_2 i_3}, \\ \mathbf{r}_{i_1 i_2 0} &= (-1)^{i_3} \mathbf{r}_{i_1 i_2 i_3}, \quad i_1, i_2, i_3 = 0, 1, \end{aligned} \tag{2.2}$$

and they can also be represented by vector sums; for example,

$$\begin{aligned} \mathbf{r}_{\bar{i}_1 i_2 0} &= (-1)^{i_3} (\mathbf{v} - \mathbf{p})_{i_1 i_2 i_3}, \\ \mathbf{r}_{i_1 \bar{i}_2 0} &= (-1)^{i_3} (\mathbf{u} - \mathbf{q})_{i_1 i_2 i_3}, \\ \mathbf{q}_{i_1 0 \bar{i}_3} &= (-1)^{i_2} (\mathbf{u} - \mathbf{r})_{i_1 i_2 i_3}. \end{aligned} \tag{2.3}$$

Consider the decompositions of the coefficients (1.2) with respect to vectors (2.1) (see Figure 2.1):

$$\begin{aligned} &\mathbf{a}_{100} = \mathbf{p}_{000}, \quad \mathbf{a}_{010} = \mathbf{q}_{000}, \quad \mathbf{a}_{001} = \mathbf{r}_{000}, \\ &\mathbf{a}_{110} = -\mathbf{q}_{000} + \mathbf{q}_{100} = \mathbf{w}_{010} - \mathbf{p}_{000} - \mathbf{w}_{010} + \mathbf{p}_{010} = -\mathbf{p}_{000} + \mathbf{p}_{010}, \\ &\mathbf{a}_{101} = -\mathbf{r}_{000} + \mathbf{r}_{100} = \mathbf{v}_{000} - \mathbf{p}_{000} - \mathbf{v}_{000} + \mathbf{p}_{001} = -\mathbf{p}_{000} + \mathbf{p}_{001}, \\ &\mathbf{a}_{011} = -\mathbf{r}_{000} + \mathbf{r}_{010} = \mathbf{u}_{000} - \mathbf{q}_{000} - \mathbf{u}_{000} + \mathbf{q}_{001} = -\mathbf{q}_{000} + \mathbf{q}_{001}, \\ &\mathbf{a}_{111} = -\mathbf{a}_{101} + \mathbf{r}_{110} - \mathbf{r}_{010} = -\mathbf{a}_{101} - \mathbf{p}_{010} + \mathbf{v}_{010} - \mathbf{v}_{010} + \mathbf{p}_{011} \\ &\qquad = \mathbf{p}_{000} - \mathbf{p}_{001} - \mathbf{p}_{010} + \mathbf{p}_{011} = -\mathbf{a}_{011} - \mathbf{r}_{100} + \mathbf{r}_{110} \\ &\qquad = \mathbf{r}_{000} - \mathbf{r}_{010} - \mathbf{r}_{100} + \mathbf{r}_{110} = -\mathbf{a}_{011} - \mathbf{q}_{100} + \mathbf{u}_{100} - \mathbf{u}_{100} + \mathbf{q}_{101} \\ &\qquad = \mathbf{q}_{000} - \mathbf{q}_{001} - \mathbf{q}_{100} + \mathbf{q}_{101}. \end{aligned} \tag{2.4}$$

By (2.4), vectors (1.4) can be obtained in the form

$$\begin{aligned} \frac{\partial \mathbf{z}}{\partial y^1} &= \mathbf{p}_{000}(1 - y^2)(1 - y^3) + \mathbf{p}_{010} y^2 (1 - y^3) + \mathbf{p}_{001} y^3 (1 - y^2) + \mathbf{p}_{011} y^2 y^3, \\ \frac{\partial \mathbf{z}}{\partial y^2} &= \mathbf{q}_{000}(1 - y^1)(1 - y^3) + \mathbf{q}_{100} y^1 (1 - y^3) + \mathbf{q}_{001} y^3 (1 - y^1) + \mathbf{q}_{101} y^1 y^3, \\ \frac{\partial \mathbf{z}}{\partial y^3} &= \mathbf{r}_{000}(1 - y^1)(1 - y^2) + \mathbf{r}_{100} y^1 (1 - y^2) + \mathbf{r}_{010} y^2 (1 - y^1) + \mathbf{r}_{110} y^1 y^2. \end{aligned} \tag{2.5}$$

Using properties (2.2) and formulas (2.5), we compute the Jacobians at the corners

$$J_{i_1 i_2 i_3} = J(i_1, i_2, i_3) = \delta_{i_1 i_2 i_3} [\mathbf{p}, \mathbf{q}, \mathbf{r}]_{i_1 i_2 i_3} = 6V^{pqr}_{i_1 i_2 i_3}, \quad i_1, i_2, i_3 = 0, 1.$$

Here and hereafter, $\delta_{i_1 i_2 i_3} = (-1)^{i_1+i_2+i_3}$, and lower indices related to brackets refer to each element inside brackets. The value $J_{i_1 i_2 i_3}$ is six times the volume of a tetrahedron with the corner $\mathbf{z}_{i_1 i_2 i_3}$ and edges $\mathbf{p}_{i_1 i_2 i_3}$, $\mathbf{q}_{i_1 i_2 i_3}$, $\mathbf{r}_{i_1 i_2 i_3}$ (the notation $V^{pqr}_{i_1 i_2 i_3}$). The notations of such type will be used to designate the volumes of tetrahedrons with the corner $\mathbf{z}_{i_1 i_2 i_3}$ and the edges corresponding to superscripts.

Lemma 2.1. *The Jacobian of trilinear map* (1.1) *can be written in the form*

$$\begin{aligned}
J = &\sum_{i_1, i_2, i_3 = 0}^{1} \alpha_{i_1 i_2 i_3} {Y_{i_1}}^2 {Y_{i_2}}^2 {Y_{i_3}}^2 \\
&+ \sum_{k=1}^{3} \sum_{\substack{i_l, i_m = 0, \\ (klm)=(123)}}^{1} \left(\sum_{i_k=0}^{1} \beta^{k i_k}_{i_l i_m} \right) y^k (1 - y^k) {Y_{i_l}}^2 {Y_{i_m}}^2 \\
&+ \sum_{k=1}^{3} \sum_{i_k=0}^{1} \left(\sum_{\substack{i_l, i_m = 0, \\ (klm)=(123)}}^{1} \gamma^{k i_k}_{i_l i_m} \right) {Y_{i_k}}^2 y^l (1 - y^l) y^m (1 - y^m) \\
&+ \sum_{i_1, i_2, i_3 = 0}^{1} \kappa_{i_1 i_2 i_3} y^1 (1 - y^1) y^2 (1 - y^2) y^3 (1 - y^3),
\end{aligned} \tag{2.6}$$

where $Y_{i_l} = i_l + (-1)^{i_l}(1 - y^l)$, $l = 1, 2, 3$, $i_l = 0, 1$ ($Y_{i_l} = 1 - y^l$ *if* $i_l = 0$ *and* $Y_{i_l} = y^l$ *if* $i_l = 1$),

$$\begin{aligned}
&\alpha_{i_1 i_2 i_3} = [\mathbf{p}_{0 i_2 i_3}, \mathbf{q}_{i_1 0 i_3}, \mathbf{r}_{i_1 i_2 0}], \quad &&\kappa_{i_1 i_2 i_3} = [\mathbf{p}_{0 i_2 i_3}, \mathbf{q}_{i_1 0 \bar{i}_3}, \mathbf{r}_{\bar{i}_1 \bar{i}_2 0}], \\
&\beta^{1 i_1}_{i_2 i_3} = \beta^1_{i_1 i_2 i_3} = [\mathbf{p}_{0 i_2 i_3}, \mathbf{q}_{i_1 0 i_3}, \mathbf{r}_{\bar{i}_1 i_2 0}], &&\gamma^{1 i_1}_{i_2 i_3} = \gamma^1_{i_1 i_2 i_3} = [\mathbf{p}_{0 i_2 i_3}, \mathbf{q}_{i_1 0 \bar{i}_3}, \mathbf{r}_{i_1 \bar{i}_2 0}], \\
&\beta^{2 i_2}_{i_3 i_1} = \beta^2_{i_1 i_2 i_3} = [\mathbf{p}_{0 i_2 i_3}, \mathbf{q}_{i_1 0 i_3}, \mathbf{r}_{i_1 \bar{i}_2 0}], &&\gamma^{2 i_2}_{i_3 i_1} = \gamma^2_{i_1 i_2 i_3} = [\mathbf{p}_{0 i_2 i_3}, \mathbf{q}_{i_1 0 \bar{i}_3}, \mathbf{r}_{\bar{i}_1 i_2 0}], \\
&\beta^{3 i_3}_{i_1 i_2} = \beta^3_{i_1 i_2 i_3} = [\mathbf{p}_{0 i_2 i_3}, \mathbf{q}_{i_1 0 \bar{i}_3}, \mathbf{r}_{i_1 i_2 0}], &&\gamma^{3 i_3}_{i_1 i_2} = \gamma^3_{i_1 i_2 i_3} = [\mathbf{p}_{0 i_2 i_3}, \mathbf{q}_{i_1 0 i_3}, \mathbf{r}_{\bar{i}_1 \bar{i}_2 0}],
\end{aligned} \tag{2.7}$$

and indices k, l, m *form the permutation of the cycle* (123)*; i.e.,* k, l, m *are equal to the values* $1, 2, 3,\ 2, 3, 1,\ 3, 1, 2$, *respectively.* (*The last one is denoted by* $(klm) = (123)$.)

Proof. Substituting (2.5) into (1.3) gives

$$J = \left[\sum_{i_2, i_3 = 0}^{1} \mathbf{p}_{0 i_2 i_3} Y_{i_2} Y_{i_3}, \sum_{i_1, i_3 = 0}^{1} \mathbf{q}_{i_1 0 i_3} Y_{i_1} Y_{i_3}, \sum_{i_1, i_2 = 0}^{1} \mathbf{r}_{i_1 i_2 0} Y_{i_1} Y_{i_2} \right].$$

We shall decompose the triple scalar product of sums of vectors into a sum of triple scalar products. First we obtain

$$J = \sum_{i_2, i_3 = 0}^{1} \left[\mathbf{p}_{0 i_2 i_3} Y_{i_2} Y_{i_3}, \sum_{i_1=0}^{1} (\mathbf{q}_{i_1 0 i_3} Y_{i_1} Y_{i_3} + \mathbf{q}_{i_1 0 \bar{i}_3} Y_{i_1} Y_{\bar{i}_3}), \mathbf{R}_{i_2} \right],$$

where

$$\mathbf{R}_{i_2} = \sum_{j_1=0}^{1} \left(\mathbf{r}_{j_1 i_2 0} Y_{j_1} Y_{i_2} + \mathbf{r}_{j_1 \bar{i}_2 0} Y_{j_1} Y_{\bar{i}_2} \right) = \sum_{j_1, j_2 = 0}^{1} \mathbf{r}_{j_1 j_2 0} Y_{j_1} Y_{j_2}, \quad i_2 = 0, 1.$$

Now we decompose with respect to the sum containing vectors $\mathbf{q}_{i_1 0 i_3}$ and take common factors of vectors out of the sign of the triple scalar product. We get

$$\begin{aligned} J = &\sum_{i_1,i_2,i_3=0}^{1} [\mathbf{p}_{0 i_2 i_3}, \mathbf{q}_{i_1 0 i_3}, \mathbf{R}_{i_1 i_2}] Y_{i_1} Y_{i_2} Y_{i_3}^2 \\ &+ \sum_{i_1,i_2,i_3=0}^{1} [\mathbf{p}_{0 i_2 i_3}, \mathbf{q}_{i_1 0 \bar{i}_3}, \mathbf{R}_{i_1 i_2}] Y_{i_1} Y_{i_2} Y_{i_3} Y_{\bar{i}_3}, \end{aligned}$$

where

$$\mathbf{R}_{i_1 i_2} = \mathbf{r}_{i_1 i_2 0} Y_{i_1} Y_{i_2} + \mathbf{r}_{\bar{i}_1 i_2 0} Y_{\bar{i}_1} Y_{i_2} + \mathbf{r}_{i_1 \bar{i}_2 0} Y_{i_1} Y_{\bar{i}_2} + \mathbf{r}_{\bar{i}_1 \bar{i}_2 0} Y_{\bar{i}_1} Y_{\bar{i}_2} = \mathbf{R}_{i_2}, \quad i_1, i_2 = 0, 1.$$

Now decompose the rest of the triple scalar products (with respect to the sum $\mathbf{R}_{i_1 i_2}$). Using the relations $Y_{i_l} Y_{\bar{i}_l} = y^l(1 - y^l)$, $i_l = 0, 1$, $l = 1, 2, 3$, we obtain (2.6) and (2.7). □

Let us introduce the notations $\alpha^{1 i_1}_{i_2 i_3} = \alpha^{2 i_2}_{i_3 i_1} = \alpha^{3 i_3}_{i_1 i_2} = \alpha_{i_1 i_2 i_3}$.

LEMMA 2.2. *The following relations are valid for coefficients* (2.7):

$$\alpha_{i_1 i_2 i_3} = [\mathbf{p}_{0 i_2 i_3}, \mathbf{q}_{i_1 0 i_3}, \mathbf{r}_{i_1 i_2 0}] = \delta_{i_1 i_2 i_3} [\mathbf{p}, \mathbf{q}, \mathbf{r}]_{i_1 i_2 i_3} = J_{i_1 i_2 i_3} = 6 V^{pqr}_{i_1 i_2 i_3} \tag{2.8}$$

$$\begin{aligned} \beta^1_{i_1 i_2 i_3} &= [\mathbf{p}_{0 i_2 i_3}, \mathbf{q}_{i_1 0 i_3}, \mathbf{r}_{\bar{i}_1 i_2 1}] = \delta_{i_1 i_2 i_3} [\mathbf{p}, \mathbf{q}, \mathbf{v}]_{i_1 i_2 i_3} = 6 V^{pqv}_{i_1 i_2 i_3}, \\ \beta^2_{i_1 i_2 i_3} &= [\mathbf{p}_{0 i_2 i_3}, \mathbf{q}_{i_1 0 i_3}, \mathbf{r}_{i_1 \bar{i}_2 0}] = \delta_{i_1 i_2 i_3} [\mathbf{p}, \mathbf{q}, \mathbf{u}]_{i_1 i_2 i_3} = 6 V^{pqu}_{i_1 i_2 i_3}, \\ \beta^3_{i_1 i_2 i_3} &= [\mathbf{p}_{0 i_2 i_3}, \mathbf{q}_{i_1 0 \bar{i}_3}, \mathbf{r}_{i_1 i_2 0}] = \delta_{i_1 i_2 i_3} [\mathbf{p}, \mathbf{u}, \mathbf{r}]_{i_1 i_2 i_3} = 6 V^{pur}_{i_1 i_2 i_3}, \end{aligned} \tag{2.9}$$

$$\begin{aligned} \sum_{\substack{i_l, i_m = 0 \\ (klm)=(123)}}^{1} \gamma^{k i_k}_{i_l i_m} &= \sum_{\substack{i_l, i_m = 0 \\ (klm)=(123)}}^{1} \left(\bar{\gamma}^{k i_k}_{i_l i_m} + \mu^{k i_k}_{i_l i_m}\right) = \sum_{\substack{i_l, i_m = 0 \\ (klm)=(123)}}^{1} \bar{\gamma}^{k i_k}_{i_l i_m} \\ &= \sum_{\substack{i_l, i_m = 0 \\ (klm)=(123)}}^{1} \left(-\alpha^{k i_k}_{i_l i_m} + \beta^{l i_l}_{i_m i_k} + \beta^{m i_m}_{i_k i_l}\right), \end{aligned} \tag{2.10}$$

$$\begin{aligned} \mu^{1 i_1}_{i_2 i_3} &= \mu^1_{i_1 i_2 i_3} = \delta_{i_1 i_2 i_3} [\mathbf{q}, \mathbf{u}, \mathbf{r}]_{i_1 i_2 i_3}, \\ \bar{\gamma}^{1 i_1}_{i_2 i_3} &= \bar{\gamma}^1_{i_1 i_2 i_3} = \delta_{i_1 i_2 i_3} [\mathbf{d}, \mathbf{q}, \mathbf{r}]_{i_1 i_2 i_3} = 6 V^{dqr}_{i_1 i_2 i_3}, \\ \mu^{2 i_2}_{i_3 i_1} &= \mu^2_{i_1 i_2 i_3} = \delta_{i_1 i_2 i_3} [\mathbf{v}, \mathbf{p}, \mathbf{r}]_{i_1 i_2 i_3}, \\ \bar{\gamma}^{2 i_2}_{i_3 i_1} &= \bar{\gamma}^2_{i_1 i_2 i_3} = \delta_{i_1 i_2 i_3} [\mathbf{p}, \mathbf{d}, \mathbf{r}]_{i_1 i_2 i_3} = 6 V^{pdr}_{i_1 i_2 i_3}, \\ \mu^{3 i_3}_{i_1 i_2} &= \mu^3_{i_1 i_2 i_3} = \delta_{i_1 i_2 i_3} [\mathbf{q}, \mathbf{p}, \mathbf{w}]_{i_1 i_2 i_3}, \\ \bar{\gamma}^{3 i_3}_{i_1 i_2} &= \bar{\gamma}^3_{i_1 i_2 i_3} = \delta_{i_1 i_2 i_3} [\mathbf{p}, \mathbf{q}, \mathbf{d}]_{i_1 i_2 i_3} = 6 V^{pqd}_{i_1 i_2 i_3}, \end{aligned} \tag{2.11}$$

$$\sum_{i_1,i_2,i_3=0}^{1} \kappa_{i_1 i_2 i_3} = \sum_{i_1,i_2,i_3=0}^{1} \sum_{k=1}^{3} \beta^k_{i_1 i_2 i_3} - 2 \sum_{i_1,i_2,i_3=0}^{1} \alpha_{i_1 i_2 i_3} = 2\bar{\kappa}_{000} + 2\bar{\kappa}_{111}, \tag{2.12}$$

$$\bar{\kappa}_{lll} = \delta_{lll} [\mathbf{u}, \mathbf{v}, \mathbf{w}]_{lll} = 6 V^{uvw}_{lll}, \quad l = 0, 1, \tag{2.13}$$

$$\sum_{i_1,i_2,i_3=0}^{1} \sum_{k=1}^{3} \gamma^k_{i_1 i_2 i_3} = \sum_{i_1,i_2,i_3=0}^{1} \alpha_{i_1 i_2 i_3} + 4(\bar{\kappa}_{000} + \bar{\kappa}_{111}), \tag{2.14}$$

$$\sum_{i_1,i_2,i_3=0}^{1} \sum_{k=1}^{3} \beta^k_{i_1 i_2 i_3} = 2 \sum_{i_1,i_2,i_3=0}^{1} \alpha_{i_1 i_2 i_3} + 2(\bar{\kappa}_{000} + \bar{\kappa}_{111}). \tag{2.15}$$

Proof. Obviously, (2.8) and (2.9) are valid by (2.2) and (2.3).

Now we prove (2.10) and (2.11). It is seen that

$$\begin{aligned}\gamma^1_{i_1i_2i_3} &= [\mathbf{p}_{0i_2i_3}, \mathbf{q}_{i_10\bar{i}_3}, \mathbf{r}_{i_1\bar{i}_20}] = [(-1)^{i_1}\mathbf{p}, (-1)^{i_2}(\mathbf{u}-\mathbf{r}), (-1)^{i_3}(\mathbf{u}-\mathbf{q})]_{i_1i_2i_3}\\ &= -\delta_{i_1i_2i_3}[\mathbf{p},\mathbf{q},\mathbf{r}]_{i_1i_2i_3} + \delta_{i_1i_2i_3}[\mathbf{p},\mathbf{q},\mathbf{u}]_{i_1i_2i_3} + \delta_{i_1i_2i_3}[\mathbf{p},\mathbf{u},\mathbf{r}]_{i_1i_2i_3}\\ &= -\alpha_{i_1i_2i_3} + \beta^2_{i_1i_2i_3} + \beta^3_{i_1i_2i_3}\end{aligned}$$

and

$$\begin{aligned}\gamma^1_{i_1i_2i_3} &= [\mathbf{p}_{0i_2i_3}, \mathbf{q}_{i_10\bar{i}_3}, \mathbf{r}_{i_1\bar{i}_20}] = \left[(-1)^{i_1}(\mathbf{d}-\mathbf{u}), (-1)^{\bar{i}_2}\mathbf{q}, (-1)^{\bar{i}_3}\mathbf{r}\right]_{i_1\bar{i}_2\bar{i}_3}\\ &= \delta_{i_1\bar{i}_2\bar{i}_3}[\mathbf{d},\mathbf{q},\mathbf{r}]_{i_1\bar{i}_2\bar{i}_3} + \delta_{i_1\bar{i}_2\bar{i}_3}[\mathbf{q},\mathbf{u},\mathbf{r}]_{i_1\bar{i}_2\bar{i}_3} = \bar{\gamma}^1_{i_1\bar{i}_2\bar{i}_3} + \mu^1_{i_1\bar{i}_2\bar{i}_3}.\end{aligned}$$

Because of $\sum_{i_2,i_3=0}^{1}\mu^1_{i_1i_2i_3} = 0$ ($[\mathbf{q},\mathbf{u},\mathbf{r}]_{i_100} = [\mathbf{u}-\mathbf{r},\mathbf{q}-\mathbf{r},-\mathbf{r}]_{i_101} = [\mathbf{q},\mathbf{u},\mathbf{r}]_{i_101} = [\mathbf{q},\mathbf{u},\mathbf{r}]_{i_111} = [\mathbf{q},\mathbf{u},\mathbf{r}]_{i_110}$, $i_1 = 0,1$), the first formula from (2.10) is valid. Similarly, we can prove other relations from (2.10) and (2.11). Also note that if all $\mu^k_{i_1i_2i_3} = 0$, $i_1, i_2, i_3 = 0,1$, $k = 1,2,3$, the faces of the cell are planar.

Let us prove further (2.14). We have

$$\begin{aligned}&\bar{\gamma}^1_{i_1i_2i_3} + \bar{\gamma}^2_{i_1i_2i_3} + \bar{\gamma}^3_{i_1i_2i_3}\\ &\quad= \delta_{i_1i_2i_3}\left([\mathbf{d},\mathbf{q},\mathbf{r}] + [\mathbf{p},\mathbf{d},\mathbf{r}] + [\mathbf{p},\mathbf{q},\mathbf{d}]\right)_{i_1i_2i_3} = \delta_{i_1i_2i_3}\left([\mathbf{p}-\mathbf{r},\mathbf{q},\mathbf{d}] + [\mathbf{p},\mathbf{d},\mathbf{r}]\right)_{i_1i_2i_3}\\ &\quad= \delta_{i_1i_2i_3}\left([\mathbf{v},\mathbf{u}-\mathbf{r},\mathbf{w}-\mathbf{r}]_{i_1i_2\bar{i}_3} + \left[\mathbf{p}_{i_1i_2i_3}, \mathbf{q}_{i_1i_2i_3} + \mathbf{v}_{i_1\bar{i}_2i_3}, \mathbf{r}_{i_1i_2i_3}\right]\right)\\ &\quad= \delta_{i_1i_2i_3}\Big(\left([\mathbf{v},\mathbf{u},\mathbf{w}] - [\mathbf{v},\mathbf{r},\mathbf{w}] - [\mathbf{v},\mathbf{u},\mathbf{r}]\right)_{i_1i_2\bar{i}_3} + [\mathbf{p},\mathbf{q},\mathbf{r}]_{i_1i_2i_3}\\ &\qquad+ \left[\mathbf{p}_{i_1i_2i_3}, \mathbf{v}_{i_1\bar{i}_2i_3}, \mathbf{r}_{i_1i_2i_3}\right]\Big)\\ &\quad= \delta_{i_1i_2i_3}\left([\mathbf{p},\mathbf{q},\mathbf{r}]_{i_1i_2i_3} + \left([\mathbf{v},\mathbf{u},\mathbf{w}] - [\mathbf{v},\mathbf{r},\mathbf{w}] - [\mathbf{v},\mathbf{u},\mathbf{r}] + [\mathbf{v}-\mathbf{r},\mathbf{w}-\mathbf{u},-\mathbf{r}]\right)_{i_1i_2\bar{i}_3}\right)\\ &\quad= \delta_{i_1i_2i_3}\left([\mathbf{p},\mathbf{q},\mathbf{r}]_{i_1i_2i_3} + \left([\mathbf{w},\mathbf{v},\mathbf{u}] - [\mathbf{v},\mathbf{r},\mathbf{w}] - [\mathbf{v},\mathbf{u},\mathbf{r}] + [\mathbf{v},\mathbf{r},\mathbf{w}] + [\mathbf{v},\mathbf{u},\mathbf{r}]\right)_{i_1i_2\bar{i}_3}\right)\\ &\quad= \delta_{i_1i_2i_3}\left[\mathbf{p},\mathbf{q},\mathbf{r}\right]_{i_1i_2i_3} - \delta_{i_1i_2i_3}\left[\mathbf{u},\mathbf{v},\mathbf{w}\right]_{i_1i_2\bar{i}_3}.\end{aligned}$$

Relations (2.13) (since $[\mathbf{u},\mathbf{v},\mathbf{w}]_{lll} = [-\mathbf{u},\mathbf{w}-\mathbf{u},\mathbf{v}-\mathbf{u}]_{l\bar{l}\bar{l}} = [\mathbf{u},\mathbf{v},\mathbf{w}]_{l\bar{l}\bar{l}} = [\mathbf{u},\mathbf{v},\mathbf{w}]_{\bar{l}\bar{l}l} = [\mathbf{u},\mathbf{v},\mathbf{w}]_{\bar{l}l\bar{l}}$, $l = 0,1$) finally leads to

$$\bar{\gamma}^1_{i_1i_2i_3} + \bar{\gamma}^2_{i_1i_2i_3} + \bar{\gamma}^3_{i_1i_2i_3} = \alpha_{i_1i_2i_3} + \bar{\kappa}_{lll},$$

where $l = 0$ if $\delta_{i_1i_2i_3} = -1$, and $l = 1$ if $\delta_{i_1i_2i_3} = 1$. Then (2.10) implies (2.14) and (2.15).

Next we prove (2.12). For each $\kappa_{i_1i_2i_3}$, consider the representation

$$\begin{aligned}\kappa_{i_1i_2i_3} &= \left[\mathbf{p}_{0i_2i_3}, \mathbf{q}_{i_10\bar{i}_3}, \mathbf{r}_{\bar{i}_1\bar{i}_20}\right] = \left[(-1)^{i_1}\mathbf{p}, (-1)^{i_2}(\mathbf{u}-\mathbf{r}), (-1)^{i_3}(\mathbf{d}-\mathbf{w})\right]_{i_1i_2i_3}\\ &= \delta_{i_1i_2i_3}\left([\mathbf{p},\mathbf{u},\mathbf{d}] + [\mathbf{p},\mathbf{d},\mathbf{r}] + [\mathbf{p},\mathbf{w},\mathbf{u}] - [\mathbf{p},\mathbf{w},\mathbf{r}]\right)_{i_1i_2i_3}.\end{aligned}$$

Since

$$\begin{aligned}\delta_{i_1i_2i_3}\left[\mathbf{p},\mathbf{u},\mathbf{d}\right]_{i_1i_2i_3} &= \delta_{i_1i_2i_3}\left[\mathbf{u}-\mathbf{d},\mathbf{p}-\mathbf{d},-\mathbf{d}\right]_{\bar{i}_1\bar{i}_2\bar{i}_3} = -\delta_{\bar{i}_1\bar{i}_2\bar{i}_3}\left[\mathbf{p},\mathbf{u},\mathbf{d}\right]_{\bar{i}_1\bar{i}_2\bar{i}_3}\\ \delta_{i_1i_2i_3}\left[\mathbf{p},\mathbf{w},\mathbf{u}\right]_{i_1i_2i_3} &= -\delta_{i_1i_2i_3}\left[-\mathbf{p},\mathbf{d}-\mathbf{p},\mathbf{q}-\mathbf{p}\right]_{\bar{i}_1i_2i_3} = -\delta_{i_1i_2i_3}\left[\mathbf{p},\mathbf{q},\mathbf{d}\right]_{\bar{i}_1i_2i_3}\\ &= \bar{\gamma}^3_{\bar{i}_1i_2i_3},\\ -\delta_{i_1i_2i_3}\left[\mathbf{p},\mathbf{w},\mathbf{r}\right]_{i_1i_2i_3} &= -\delta_{i_1i_2i_3}\left[-\mathbf{p},\mathbf{q}-\mathbf{p},\mathbf{v}-\mathbf{p}\right]_{\bar{i}_1i_2i_3} = -\beta^1_{i_1i_2i_3},\\ \delta_{i_1i_2i_3}\left[\mathbf{p},\mathbf{d},\mathbf{r}\right]_{i_1i_2i_3} &= \bar{\gamma}^2_{i_1i_2i_3},\end{aligned}$$

we have $\sum_{i_1,i_2,i_3=0}^{1} \delta_{i_1 i_2 i_3}[\mathbf{p}, \mathbf{u}, \mathbf{d}]_{i_1 i_2 i_3} = 0$. Hence, the relation (2.12) is proved by (2.10) and (2.15). □

Therefore, the Jacobian of the map can be represented by the sum of polynomials of the sixth degree (second degree in each of variables). The coefficients of the polynomial are expressed in terms of the volumes of tetrahedrons of four types (2.8), (2.9), (2.11) ($\bar{\gamma}^k_{i_1 i_2 i_3}$), (2.13). Tetrahedrons (2.8) are formed by three edges (with the common corner), tetrahedrons (2.9) by two edges and "diagonal" of one of adjacent faces, tetrahedrons (2.11) ($\bar{\gamma}^k_{i_1 i_2 i_3}$) by two edges and "inner diagonal" of the cell, tetrahedrons (2.13) by "diagonals" of faces. The total number of tetrahedrons is $8 + 24 + 24 + 2 = 58$.

Lemmas 2.1 and 2.2 allow us to prove the theorem.

THEOREM 2.3. *The Jacobian of trilinear map* (1.1) *is a polynomial of the degree not higher than fourth* (*second in each of variables*). *It can be written in the form*

$$\begin{aligned} J = &\sum_{i_1,i_2,i_3=0}^{1} \alpha_{i_1 i_2 i_3} Y_{i_1} Y_{i_2} Y_{i_3} (Y_{i_1} + Y_{i_2} + Y_{i_3} - 2) \\ &+ \sum_{k=1}^{3} \sum_{\substack{i_l, i_m = 0 \\ (klm)=(123)}}^{1} \left(\sum_{i_k=0}^{1} \beta^{k i_k}_{i_l i_m} \right) y^k (1 - y^k) Y_{i_l} Y_{i_m}. \end{aligned} \tag{2.16}$$

Proof. Substitute (2.10) and (2.12) in (2.6). Because of $Y_{i_l} + Y_{\bar{i}_l} = 1$, $i_l = 0, 1$, $l = 1, 2, 3$, the factors of coefficients $\alpha_{i_1 i_2 i_3}$, $\beta^k_{i_1 i_2 i_3}$ are equal to

$$Y_{i_1} Y_{i_2} Y_{i_3} - Y_{i_1} Y_{\bar{i}_2} Y_{\bar{i}_3} - Y_{\bar{i}_1} Y_{i_2} Y_{\bar{i}_3} - Y_{\bar{i}_1} Y_{\bar{i}_2} Y_{i_3} - 2 Y_{\bar{i}_1} Y_{\bar{i}_2} Y_{\bar{i}_3} = Y_{i_1} + Y_{i_2} + Y_{i_3} - 2,$$

$$Y_{i_n} Y_{i_m} + Y_{i_n} Y_{\bar{i}_m} + Y_{\bar{i}_n} Y_{i_m} + Y_{\bar{i}_n} Y_{\bar{i}_m} = 1, \qquad n \neq m \neq k, \quad n, m = 1, 2, 3,$$

respectively. This proves (2.16). □

Since $Y_{i_1} + Y_{i_2} + Y_{i_3} - 2 = 1 - Y_{\bar{i}_1} - Y_{\bar{i}_2} - Y_{\bar{i}_3}$, the formula (2.16) can be reduced to the following one:

$$J = \sum_{i_1,i_2,i_3=0}^{1} \alpha_{i_1 i_2 i_3} Y_{i_1} Y_{i_2} Y_{i_3} + \sum_{k=1}^{3} \sum_{\substack{i_l, i_m = 0 \\ (klm)=(123)}}^{1} \left(\sum_{i_k=0}^{1} \left(\beta^{k i_k}_{i_l i_m} - \alpha^{k i_k}_{i_l i_m} \right) \right) y^k (1 - y^k) Y_{i_l} Y_{i_m}. \tag{2.17}$$

Formula (2.17) requires computations of eight volumes of tetrahedrons (2.8) and 12 triple scalar products

$$\begin{aligned} \delta_{i_1 i_2 i_3} \left[\mathbf{p}_{i_1 i_2 i_3}, \mathbf{p}_{\bar{i}_1 \bar{i}_2 i_3}, \mathbf{p}_{i_1 i_2 \bar{i}_3} \right] &= \sum_{i_1=0}^{1} \left(\beta^1_{i_1 i_2 i_3} - \alpha_{i_1 i_2 i_3} \right), \\ \delta_{i_1 i_2 i_3} \left[\mathbf{q}_{\bar{i}_1 \bar{i}_2 i_3}, \mathbf{q}_{i_1 i_2 i_3}, \mathbf{q}_{i_1 i_2 \bar{i}_3} \right] &= \sum_{i_2=0}^{1} \left(\beta^2_{i_1 i_2 i_3} - \alpha_{i_1 i_2 i_3} \right), \\ \delta_{i_1 i_2 i_3} \left[\mathbf{r}_{\bar{i}_1 i_2 \bar{i}_3}, \mathbf{r}_{i_1 \bar{i}_2 i_3}, \mathbf{r}_{i_1 i_2 i_3} \right] &= \sum_{i_3=0}^{1} \left(\beta^3_{i_1 i_2 i_3} - \alpha_{i_1 i_2 i_3} \right). \end{aligned} \tag{2.18}$$

Expressions for J on the edges have the form

$$J\left(y^1, y^2, y^3\right) \Big|_{\substack{y^l = i_l \\ y^m = i_m}} = \alpha^{k0}_{i_l i_m} \left(1 - y^k\right)^2 + \alpha^{k1}_{i_l i_m} {y^k}^2 + \left(\beta^{k0}_{i_l i_m} + \beta^{k1}_{i_l i_m}\right) y^k \left(1 - y^k\right),$$
$$(klm) = (123), \quad i_l, i_m = 0, 1. \tag{2.19}$$

Using (2.19), the formula (2.17) can be rewritten in the following terms of values of J on the edges as in [7]:

$$\begin{aligned}(2.20)\qquad J = &\sum_{i_1,i_2=0}^{1} J\left(i_1,i_2,y^3\right) Y_{i_1} Y_{i_2} + \sum_{i_1,i_3=0}^{1} J\left(i_1,y^2,i_3\right) Y_{i_1} Y_{i_3} \\ &+ \sum_{i_2,i_3=0}^{1} J\left(y^1,i_2,i_3\right) Y_{i_2} Y_{i_3} - 2 \sum_{i_1,i_2,i_3=0}^{1} J\left(i_1,i_2,i_3\right) Y_{i_1} Y_{i_2} Y_{i_3}.\end{aligned}$$

Apparently, if $\sum_{i_k=0}^{1} \beta^k_{i_1 i_2 i_3}$ can be expressed in terms of the values $J_{i_1 i_2 i_3}$, $J_{\frac{1}{2} i_2 i_3}$, $J_{i_1 \frac{1}{2} i_3}$, $J_{i_1 i_2 \frac{1}{2}}$ (see (3.1)), then J can be represented as a polynomial with coefficients that are the values of J at the corners and midpoints of edges. Such representation is also given in [7].

If a hexahedral cell is a parallelepiped, then

$$(2.21)\qquad \alpha_{i_1 i_2 i_3} = \alpha_{000} = \beta^k_{i_1 i_2 i_3}, \quad k = 1,2,3, \quad i_1,i_2,i_3 = 0,1,$$

and the formula for the Jacobian (2.17) becomes

$$(2.22)\qquad J = \alpha_{000}(1 - y^1 - y^2 - y^3) + \alpha_{100} y^1 + \alpha_{010} y^2 + \alpha_{001} y^3.$$

(In [7] the coefficients α_{000}, α_{100}, α_{010}, α_{001} from (2.22) are the values J_{000}, J_{100}, J_{010}, J_{001}, respectively.) Using (2.21) and (2.22), we have

$$J = \alpha_{000} = \text{const}.$$

3. Positivity of the Jacobian of a trilinear map.

3.1. Necessary conditions. The condition $J > 0$ implies the inequalities

$$\begin{aligned}(3.1)\qquad & J\left(i_1,i_2,i_3\right) = \alpha_{i_1 i_2 i_3} > 0, \quad i_1,i_2,i_3 = 0,1; \\ & J\left(y^1,y^2,y^3\right)\Bigg|_{\substack{y^k=\frac{1}{2} \\ y^l,y^m=i_l,i_m}} = \frac{1}{4}\sum_{i_k=0}^{1}\left(\alpha^{k i_k}_{i_l i_m} + \beta^{k i_k}_{i_l i_m}\right) > 0, \quad (klm) = (123), \quad i_l,i_m = 0,1; \\ & J\left(y^1,y^2,y^3\right)\Bigg|_{\substack{y^k=i_k \\ y^l,y^m=\frac{1}{2}}} = \frac{1}{8}\sum_{i_l,i_m=0}^{1}\left(\beta^{l i_l}_{i_m i_k} + \beta^{m i_m}_{i_k i_l}\right) > 0, \quad (klm) = (123), \quad i_k = 0,1; \\ & J\left(\frac{1}{2},\frac{1}{2},\frac{1}{2}\right) = \frac{1}{16}\left(\sum_{i_1,i_2,i_3=0}^{1}\sum_{k=1}^{3}\beta^k_{i_1 i_2 i_3} - \sum_{i_1,i_2,i_3=0}^{1}\alpha_{i_1 i_2 i_3}\right) > 0,\end{aligned}$$

which compose the necessary conditions of nondegeneracy of a cell.

3.2. Sufficient conditions 1. The polynomials corresponding to the coefficients from (2.6) in the interior of the cube P are positive. Hence, if coefficients $\alpha_{i_1 i_2 i_3}$ are positive, and the rest of the coefficients are greater than or equal to zero, then the Jacobian is positive in the interior. It is easy to see that J is positive on the boundary.

Since the coefficients $\gamma^k_{i_1 i_2 i_3}$, $\sum_{i_1,i_2,i_3=0}^{1} \kappa_{i_1 i_2 i_3}$ can be expressed in terms of $\alpha_{i_1 i_2 i_3}$, $\beta^k_{i_1 i_2 i_3}$, the conditions of the positivity of the Jacobian (sufficient conditions 1) have the form

$$
\begin{aligned}
&(3.2)\qquad \alpha_{i_1 i_2 i_3} > 0, \quad i_1, i_2, i_3 = 0, 1;\\
&\sum_{i_k=0}^{1} \beta^{k i_k}_{i_l i_m} \geq \mathrm{B}^k_{i_l i_m}, \quad (klm) = (123), \quad i_l, i_m = 0, 1;\\
&\sum_{i_l, i_m=0}^{1} \gamma^{k i_k}_{i_l i_m} = \sum_{i_l, i_m=0}^{1} \left(\beta^{l i_l}_{i_m i_k} + \beta^{m i_m}_{i_k i_l} - \alpha^{k i_k}_{i_l i_m}\right) \geq \Gamma^k_{i_k}, \quad (klm) = (123), \quad i_k = 0, 1;\\
&2\bar{\kappa}_{000} + 2\bar{\kappa}_{111} = \sum_{i_1, i_2, i_3=0}^{1} \sum_{k=1}^{3} \beta^k_{i_1 i_2 i_3} - 2 \sum_{i_1, i_2, i_3=0}^{1} \alpha_{i_1 i_2 i_3} \geq \mathrm{K},
\end{aligned}
$$

where $\alpha_{i_1 i_2 i_3}$, $\beta^k_{i_1 i_2 i_3}$ are calculated according to formulas (2.8), (2.9), and

$$\mathrm{B}^k_{i_l i_m} = \Gamma^k_{i_k} = \mathrm{K} = 0. \tag{3.3}$$

The conditions (3.2) and (3.3) are satisfied, in particular, for hexahedral cells with the same orientation of vectors of 58 tetrahedrons (2.8), (2.9), (2.11), (2.13), corresponding to coefficients $\alpha_{i_1 i_2 i_3}$, $\beta^k_{i_1 i_2 i_3}$, $\bar{\gamma}^k_{i_1 i_2 i_3}$, $\bar{\kappa}_{000}$, $\bar{\kappa}_{111}$, that a cube has (right-handed orientation). Also note that necessary conditions coincide with (3.2), where equality signs are excluded (strict inequality form of (3.2)) and the following expressions are used:

$$\mathrm{B}^k_{i_l i_m} = -\sum_{i_k=0}^{1} \alpha^{k i_k}_{i_l i_m}, \quad \Gamma^k_{i_k} = -\sum_{i_l, i_m=0}^{1} \alpha^{k i_k}_{i_l i_m}, \quad \mathrm{K} = -\sum_{i_1, i_2, i_3=0}^{1} \alpha_{i_1 i_2 i_3}.$$

It is clear that necessary conditions assume a wider range of values for $\alpha_{i_1 i_2 i_3}$, $\beta^k_{i_1 i_2 i_3}$ than sufficient conditions 1. Both conditions include 27 inequalities (for 8 corners, 12 edges, 6 faces, and the interior part of a cell).

The expression for J is also positive if (2.18) and (2.8) are positive. However, these conditions restrict (in comparison with (3.2)) the set of values of $\alpha_{i_1 i_2 i_3}$, $\beta^k_{i_1 i_2 i_3}$ for which J is positive. Therefore, sufficient conditions 1 are more general conditions.

Thus, the result of sections 3.1 and 3.2 is the following theorem.

THEOREM 3.1. *In order that the Jacobian of trilinear map* (1.1) *be positive in the whole cube P including boundary,*

1. *it is necessary that conditions* (3.1) *be satisfied;*
2. *it is sufficient that conditions* (3.2) *with parameters* (3.3) *be satisfied.*

3.3. Necessary and sufficient conditions of positivity of the Jacobian on the edges. Consider the quadratic trinomial

$$f(y) = -ay^2 + (a + \alpha_1 - \alpha_0)y + \alpha_0, \tag{3.4}$$

where a, α_0, α_1 are parameters, and $a \neq 0$.

LEMMA 3.2. *Function $f(y)$* (3.4) *is positive on the segment* $[0, 1]$ *if and only if the parameters a, α_0, α_1 satisfy the inequalities*

$$\alpha_0 > 0, \quad \alpha_1 > 0, \tag{3.5}$$

$$a + \alpha_0 + \alpha_1 > -2\sqrt{\alpha_0 \alpha_1}. \tag{3.6}$$

Proof. Since $f'(y) = -2ay + a + \alpha_1 - \alpha_0$, the extremum of function (3.4) $y_* = \frac{1}{2} + \frac{\alpha_1 - \alpha_0}{2a}$ belongs to the segment $[0, 1]$ if

$$\left|\frac{\alpha_1 - \alpha_0}{2a}\right| \leq \frac{1}{2}.$$

Let $f_M = \max_{[0,1]} f(y)$, $f_m = \min_{[0,1]} f(y)$, $\alpha_M = \max(\alpha_0, \alpha_1)$, $\alpha_m = \min(\alpha_0, \alpha_1)$.

The discriminant of quadratic trinomial (3.4) has the form

$$D = (a + \alpha_1 - \alpha_0)^2 + 4a\alpha_0 = (a + \alpha_1 + \alpha_0)^2 - 4\alpha_0\alpha_1. \tag{3.7}$$

Inequalities (3.5) are obvious. Assume that they are satisfied.

Consider the following five cases.

1. If $a > \max[(\alpha_1 - \alpha_0), (\alpha_0 - \alpha_1)]$, then the function $f(y)$ has a maximum at the point $y = y_*$, $y_* \in [0, 1]$, $f_M = f(y_*) > 0$, $f_m = \alpha_m$.
2. In the case $0 \leq a \leq \max[(\alpha_1 - \alpha_0), (\alpha_0 - \alpha_1)]$ the function $f(y)$ has a maximum at the point $y = y_*$, $y_* \notin [0, 1]$, $f_M = \alpha_M$, $f_m = \alpha_m$.
3. If $\min[(\alpha_1 - \alpha_0), (\alpha_0 - \alpha_1)] \leq a < 0$, the function $f(y)$ has a minimum at the point $y = y_*$, $y_* \notin [0, 1]$, $f_M = \alpha_M$, $f_m = \alpha_m$.
4. If $-\alpha_1 - \alpha_0 - 2\sqrt{\alpha_0\alpha_1} < a < \min[(\alpha_1 - \alpha_0), (\alpha_0 - \alpha_1)]$, the function $f(y)$ has a minimum at the point $y = y_*$, $y_* \in [0, 1]$, $f_M = \alpha_M$, $f_m = f(y_*) > 0$.
5. If $a \leq -\alpha_1 - \alpha_0 - 2\sqrt{\alpha_0\alpha_1}$, then the function $f(y)$ has a minimum at the point $y = y_*$, $y_* \in [0, 1]$, $f_M = \alpha_M$, but $f_m = f(y_*) \leq 0$.

Cases 1–5 imply (3.6). □

Note. If $a = 0$, then necessary and sufficient conditions of positivity for f coincide with (3.5).

For quadratic trinomial (3.4), also consider the sufficient conditions of its positivity involving inequalities (3.5) and the conditions

$$a + \alpha_1 + \alpha_0 > -2\alpha_j \quad (a + 3\alpha_{\bar{j}} + \alpha_j > 0), \quad j = 0, 1, \tag{3.8}$$

or their equivalent form with max/min operation

$$a > \max[-3\alpha_0 - \alpha_1, -\alpha_0 - 3\alpha_1] = -\min[3\alpha_0 + \alpha_1, \alpha_0 + 3\alpha_1],$$

arising from cases 1–5 above and the condition $D < 0$ for (3.7). If (3.5) holds, then the value $a = 0$ satisfies inequalities (3.8). Therefore, for any values of parameter a, for the positivity of function f (3.4), sufficient conditions (3.5) and (3.8) will be considered.

Polynomials (2.19) can be reduced to the form (3.4), where

$$\begin{aligned}
a = \phi^3_{i_1 i_2} &= \beta^3_{i_1 i_2 1} + \beta^3_{i_1 i_2 0} - \alpha_{i_1 i_2 1} - \alpha_{i_1 i_2 0}, \quad \alpha_1 = \alpha_{i_1 i_2 1}, \quad \alpha_0 = \alpha_{i_1 i_2 0}, \quad y = y^3;\\
a = \phi^2_{i_1 i_3} &= \beta^2_{i_1 1 i_3} + \beta^2_{i_1 0 i_3} - \alpha_{i_1 1 i_3} - \alpha_{i_1 0 i_3}, \quad \alpha_1 = \alpha_{i_1 1 i_3}, \quad \alpha_0 = \alpha_{i_1 0 i_3}, \quad y = y^2;\\
a = \phi^1_{i_2 i_3} &= \beta^1_{1 i_2 i_3} + \beta^1_{0 i_2 i_3} - \alpha_{1 i_2 i_3} - \alpha_{0 i_2 i_3}, \quad \alpha_1 = \alpha_{1 i_2 i_3}, \quad \alpha_0 = \alpha_{0 i_2 i_3}, \quad y = y^1
\end{aligned} \tag{3.9}$$

for $J(i_1, i_2, y^3)$, $J(i_1, y^2, i_3)$, $J(y^1, i_2, i_3)$, respectively. The following theorem follows from Lemma 3.2 and (3.4) and (3.9).

Theorem 3.3. *Necessary and sufficient conditions of positivity of J on the edges of the cube have the form*

$$\begin{aligned}
&\alpha_{i_1 i_2 i_3} > 0, \quad i_1, i_2, i_3 = 0, 1;\\
&\sum_{i_k=0}^{1} \beta^{k i_k}_{i_l i_m} > -2\sqrt{\alpha^{k0}_{i_l i_m}\alpha^{k1}_{i_l i_m}}, \quad (klm) = (123), \quad i_l, i_m = 0, 1.
\end{aligned} \tag{3.10}$$

Note. If for some edges the value a turned out to be equal to zero, then corresponding restrictions on coefficients $\beta^k_{i_1 i_2 i_3}$ can be excluded from conditions (3.10). (In this case, such restrictions are satisfied.)

Now we reduce the expression for the Jacobian J on the faces $y^l = 0, 1$, $l = 1, 2, 3$, and the general formula (2.17) to the form (3.4).

Using (2.20), the Jacobian for the face $y^3 = 0$ can be written as

$$\begin{aligned} J(y^1, y^2, 0) &= g(y^1, y^2) \\ &= J(0, y^2, 0)(1 - y^1) + J(1, y^2, 0)y^1 + J(y^1, 0, 0)(1 - y^2) \\ &\quad + J(y^1, 1, 0)y^2 - J(0, 0, 0)(1 - y^1)(1 - y^2) - J(0, 1, 0)(1 - y^1)y^2 \\ &\quad - J(1, 0, 0)y^1(1 - y^2) - J(1, 1, 0)y^1y^2 \\ &= -(\phi^2_{00}(1 - y^1) + \phi^2_{10}y^1){y^2}^2 \\ &\quad + (\phi^2_{00}(1 - y^1) + \phi^2_{10}y^1 + J(y^1, 1, 0) - J(y^1, 0, 0))y^2 + J(y^1, 0, 0). \end{aligned} \tag{3.11}$$

Here, $a = a^2_{00}(y_1) = \phi^2_{00}(1-y^1)+\phi^2_{10}y^1$, $\alpha_1 = J(y^1, 1, 0)$, $\alpha_0 = J(y^1, 0, 0)$, $y = y^2$, and $\phi^2_{i_1 0}$, $i_1 = 0, 1$, are defined in (3.9). A similar representation in the form of quadratic trinomial is valid with respect to the variable y^1.

The representations (3.4), where

$$\begin{aligned} &a = a^3_{i_1 0}(y^2) = \phi^3_{i_1 0}(1 - y^2) + \phi^3_{i_1 1}y^2, \quad \alpha_1 = J(i_1, y^2, 1), \quad \alpha_0 = J(i_1, y^2, 0), \quad y = y^3; \\ &a = a^3_{0 i_2}(y^1) = \phi^3_{0 i_2}(1 - y^1) + \phi^3_{1 i_2}y^1, \quad \alpha_1 = J(y^1, i_2, 1), \quad \alpha_0 = J(y^1, i_2, 0), \quad y = y^3; \\ &a = a^2_{0 i_3}(y^1) = \phi^2_{0 i_3}(1 - y^1) + \phi^2_{1 i_3}y^1, \quad \alpha_1 = J(y^1, 1, i_3), \quad \alpha_0 = J(y^1, 0, i_3), \quad y = y^2, \end{aligned} \tag{3.12}$$

will be used for the faces $y^1 = i_1$, $i_1 = 0, 1$ $y^2 = i_2$, $i_2 = 0, 1$, $y^3 = i_3$, $i_3 = 0, 1$, respectively.

The Jacobian in the general case can be represented in the form of a quadratic trinomial in one variable with fixed two other variables. For example,

$$J(y^1, y^2, y^3) = -A^3_{00}{y^3}^2 + (A^3_{00} + J(y^1, y^2, 1) - J(y^1, y^2, 0))y^3 + J(y^1, y^2, 0), \tag{3.13}$$

where $A^3_{00} = A^3_{00}(y^1, y^2) = a^3_{00}(y^1) + (a^3_{01}(y^1) - a^3_{00}(y^1))y^2$, and $a^3_{01}(y^1), a^3_{00}(y^1)$ are defined in (3.12).

An attempt to find necessary and sufficient conditions of positivity of the Jacobian using its representation in the form of a quadratic trinomial fails since even in the case of faces (3.12) the discriminant (3.7) of quadratic trinomial (3.4) is a fourth-degree polynomial in one variable; in the case (3.13), the discriminant will be the polynomial in two variables. Because of the above reasons an analysis of the discriminant on the property of having fixed sign fails. However, using conditions (3.5) and (3.8), it is possible to find sufficient conditions more general than (3.2).

3.4. Sufficient conditions 2. To obtain sufficient conditions 2, we shall write down (3.5) and (3.8) for different representations (3.4).

Sufficient conditions for the edges include relations (3.5), (3.8), and (3.9).

Sufficient conditions of positivity of the Jacobian on the faces of a cell are conditions (3.5), (3.8), and (3.12). Substituting (3.12) into (3.5) and (3.8) gives the conditions of positivity of polynomials (3.4) with the following parameters for the planes:

$$\begin{array}{l}
y^1 = i_1, i_1 = 0,1 : \\
\quad a = 3\phi^2_{i_1 1} + \phi^2_{i_1 0}; \quad \alpha_0 = \phi^3_{i_1 0} + \alpha_{i_1 00} + 3\alpha_{i_1 01}, \quad \alpha_1 = \phi^3_{i_1 1} + \alpha_{i_1 10} + 3\alpha_{i_1 11}, \quad y = y^2; \\
\quad a = \phi^2_{i_1 1} + 3\phi^2_{i_1 0}; \quad \alpha_0 = \phi^3_{i_1 0} + 3\alpha_{i_1 00} + \alpha_{i_1 01}, \quad \alpha_1 = \phi^3_{i_1 1} + 3\alpha_{i_1 10} + \alpha_{i_1 11}, \quad y = y^2; \\
y^2 = i_2, i_2 = 0,1 : \\
\quad a = 3\phi^1_{i_2 1} + \phi^1_{i_2 0}; \quad \alpha_0 = \phi^3_{0 i_2} + \alpha_{0 i_2 0} + 3\alpha_{0 i_2 1}, \quad \alpha_1 = \phi^3_{1 i_2} + \alpha_{1 i_2 0} + 3\alpha_{1 i_2 1}, \quad y = y^1; \\
\quad a = \phi^1_{i_2 1} + 3\phi^1_{i_2 0}; \quad \alpha_0 = \phi^3_{0 i_2} + 3\alpha_{0 i_2 0} + \alpha_{0 i_2 1}, \quad \alpha_1 = \phi^3_{1 i_2} + 3\alpha_{1 i_2 0} + \alpha_{1 i_2 1}, \quad y = y^1; \\
y^3 = i_3, i_3 = 0,1 : \\
\quad a = 3\phi^1_{1 i_3} + \phi^1_{0 i_3}; \quad \alpha_0 = \phi^2_{0 i_3} + \alpha_{00 i_3} + 3\alpha_{01 i_3}, \quad \alpha_1 = \phi^2_{1 i_3} + \alpha_{10 i_3} + 3\alpha_{11 i_3}, \quad y = y^1; \\
\quad a = \phi^1_{1 i_3} + 3\phi^1_{0 i_3}; \quad \alpha_0 = \phi^2_{0 i_3} + 3\alpha_{00 i_3} + \alpha_{01 i_3}, \quad \alpha_1 = \phi^2_{1 i_3} + 3\alpha_{10 i_3} + \alpha_{11 i_3}, \quad y = y^1.
\end{array}$$
(3.14)

In turn, considering conditions (3.5) and (3.8) for polynomials (3.4) and (3.14), together with conditions (3.5) and (3.12), we obtain sufficient conditions of positivity of J on the faces of a cell.

Write down conditions (3.5) and (3.8) for the plane $y^3 = 0$. They have the form

$$\begin{array}{l}
\Phi^2_{i_1 0} + \mathrm{A}_{i_1 00} + 3\mathrm{A}_{i_1 10} > 0, \\
\Phi^2_{i_1 0} + 3\mathrm{A}_{i_1 00} + \mathrm{A}_{i_1 10} > 0, \quad i_1 = 0,1,
\end{array} \tag{3.15}$$

$$\begin{array}{l}
3\Phi^1_{10} + \Phi^1_{00} + 3\Phi^2_{10} + \Phi^2_{00} + \mathrm{A}_{000} + 3\mathrm{A}_{010} + 3\mathrm{A}_{100} + 9\mathrm{A}_{110} > 0, \\
3\Phi^1_{10} + \Phi^1_{00} + \Phi^2_{10} + 3\Phi^2_{00} + \mathrm{A}_{100} + 3\mathrm{A}_{110} + 3\mathrm{A}_{000} + 9\mathrm{A}_{010} > 0, \\
\Phi^1_{10} + 3\Phi^1_{00} + \Phi^2_{10} + 3\Phi^2_{00} + \mathrm{A}_{110} + 3\mathrm{A}_{010} + 3\mathrm{A}_{100} + 9\mathrm{A}_{000} > 0, \\
\Phi^1_{10} + 3\Phi^1_{00} + 3\Phi^2_{10} + \Phi^2_{00} + \mathrm{A}_{010} + 3\mathrm{A}_{000} + 3\mathrm{A}_{110} + 9\mathrm{A}_{100} > 0,
\end{array} \tag{3.16}$$

where

$$\Phi^k_{ij} = \phi^k_{ij}, \quad \mathrm{A}_{ij0} = \alpha_{ij0}, \quad k = 1,2, \quad i,j = 0,1. \tag{3.17}$$

Conditions (3.15) and (3.17) are equivalent to conditions (3.5), and (3.16) and (3.17) are equivalent to conditions (3.8) for the face $y^3 = 0$. Apparently, conditions (3.5) for the face $y^3 = 0$ coincide with conditions (3.8) for the edges $y^1 = 0, 1$, $y^3 = 0$.

Using (3.5) in (3.11),

$$g(y^1, i_2) = \alpha_{i_2} = J(y^1, i_2, 0) > 0, \quad i_2 = 0,1, \tag{3.18}$$

conditions (3.15)–(3.17), and the conditions for $\mathrm{A}_{i_1 i_2 0} = \alpha_{i_1 i_2 0}$,

$$\mathrm{A}_{i_1 i_2 0} > 0, \tag{3.19}$$

we get the conditions of positivity for function g (3.11) in the square $0 \leq y^1, y^2 \leq 1$.

Now consider the Jacobian in the general case. By representation (3.13), $J > 0$ in the cube P, if conditions (3.5) for (3.13),

$$\alpha_{i_3} = J(y^1, y^2, i_3) > 0, \quad 0 \leq y^1, y^2 \leq 1, \quad i_3 = 0,1, \tag{3.20}$$

are satisfied, and if conditions (3.8), also for (3.13),

$$g_0(y^1, y^2) = A^3_{00} + 3J(y^1, y^2, 1) + J(y^1, y^2, 0) > 0, \tag{3.21}$$

$$g_1(y^1, y^2) = A^3_{00} + J(y^1, y^2, 1) + 3J(y^1, y^2, 0) > 0 \tag{3.22}$$

are satisfied. Conditions (3.21) and (3.22) hold if inequalities (3.15)–(3.16) and (3.18)–(3.19) for functions $g = g_l(y^1, y^2)$ from (3.21) and (3.22) are satisfied for the following parameters:

$$\begin{aligned} &(3.23) && \Phi^k_{00} = 3\phi^k_{01} + \phi^k_{00}, \quad \Phi^k_{10} = 3\phi^k_{11} + \phi^k_{10}, \quad k = 1, 2,\\ &&& \mathrm{A}_{i_1 i_2 0} = 3\alpha_{i_1 i_2 1} + \alpha_{i_1 i_2 0} + \phi^3_{i_1 i_2}, \quad i_1, i_2 = 0, 1, \quad l = 0;\\ &(3.24) && \Phi^k_{00} = \phi^k_{01} + 3\phi^k_{00}; \quad \Phi^k_{10} = \phi^k_{11} + 3\phi^k_{10}, \quad k = 1, 2,\\ &&& \mathrm{A}_{i_1 i_2 0} = \alpha_{i_1 i_2 1} + 3\alpha_{i_1 i_2 0} + \phi^3_{i_1 i_2}, \quad i_1, i_2 = 0, 1, \quad l = 1. \end{aligned}$$

The conditions (3.16), (3.23), and (3.24) (8 inequalities) are equivalent to conditions (3.8) for the general form of J. The conditions (3.20), $g_l(y^1, 0) > 0$, and (3.15) and (3.23)–(3.24) give conditions (3.8) for planes $y^3 = 0, 1$, $y^2 = 0, 1$, $y^1 = 0, 1$, respectively ($6 \times 4 = 24$ inequalities); conditions (3.19), (3.23), and (3.24) give conditions (3.8) for the edges $y^1 = 0, 1$, $y^2 = 0, 1$; inequalities (3.20) also give conditions (3.8) for the edges $y^1 = 0, 1$, $y^3 = 0, 1$. Adding to the above conditions the conditions (3.8) for the rest of edges (the conditions of the form (3.18) must be satisfied for (3.20), for the edges there will be $12 \times 2 = 24$ inequalities) and 8 conditions $\alpha_{i_1 i_2 i_3} > 0$ yield sufficient conditions 2 of positivity of J in the cube. There will be 64 inequalities. We shall formulate them in the following theorem using the equivalent form of inequalities (as for (3.8)) with min operation and notations $\alpha_{i_1 i_2 i_3}$, $\beta^k_{i_1 i_2 i_3}$.

Theorem 3.4 (sufficient conditions 2). *In order that the Jacobian of trilinear map* (1.1) *be positive in the whole cube P including boundary, it is sufficient that conditions* (3.2) (*in strict inequality form*), *where*

$$\mathrm{B}^k_{i_l i_m} = -2 \min(\alpha^{k0}_{i_l i_m}, \alpha^{k1}_{i_l i_m}), \quad (klm) = (123), \quad i_l, i_m = 0, 1;$$

$$\Gamma^k_{i_k} = -2 \min_{j_l, j_m = 0,1} \left(2\alpha^{k i_k}_{j_l j_m} + \sum_{i_l=0}^{1} \beta^{l i_l}_{j_m i_k} + \sum_{i_m=0}^{1} \beta^{m i_m}_{i_k j_l} \right), \quad (klm) = (123), \quad i_k = 0, 1;$$

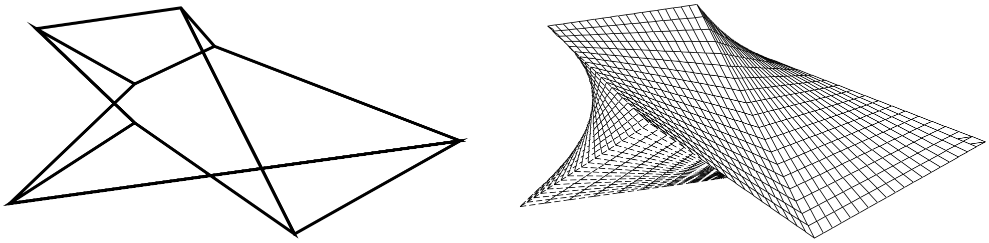
$$\begin{aligned} \mathrm{K} = -2 \min_{j_1, j_2, j_3 = 0,1} & \left(3\alpha_{j_1 j_2 j_3} + 2 \left(\sum_{i_2, i_3 = 0}^{1} \gamma^1_{j_1 i_2 i_3} + \sum_{i_1, i_3 = 0}^{1} \gamma^2_{i_1 j_2 i_3} + \sum_{i_1, i_2 = 0}^{1} \gamma^3_{i_1 i_2 j_3} \right) \right.\\ & \left. + \sum_{i_1=0}^{1} \beta^1_{i_1 j_2 j_3} + \sum_{i_2=0}^{1} \beta^2_{j_1 i_2 j_3} + \sum_{i_3=0}^{1} \beta^3_{j_1 j_2 i_3} \right), \end{aligned}$$

be satisfied.

Note. In sufficient conditions 2 the strict inequality form of (3.2) is considered: from inequalities (3.2) equality signs are excluded.

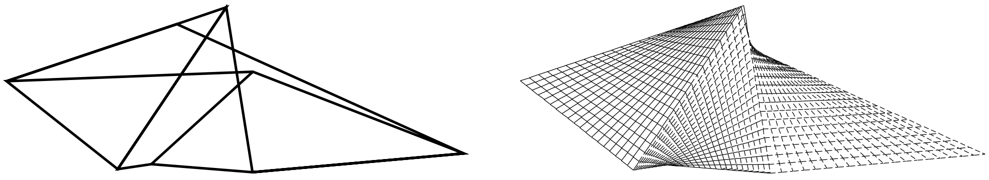
It is obvious that cells satisfying sufficient conditions 1 satisfy sufficient condition 2. Sufficient conditions 2 are more general than sufficient conditions 1 but demand more computations. It is possible to show that when sufficient conditions 1 are satisfied, necessary conditions (3.1) also hold (but not vice versa). Really, considering the form of sufficient conditions 2 without min operation and summing inequalities corresponding to each min operation, we have system of inequalities (3.1).

3.5. Numerical results. To see how general the obtained conditions are, a numerical experiment was carried out. The corners of a hexahedron were selected randomly. Of 10^7 hexahedrons randomly generated by the computer only 36251 were found to have positive Jacobians at the corners of a cell. From them only 14622 cases satisfied necessary conditions for edges, 14010 cases satisfied necessary conditions for



(a) (b)

FIG. 3.1. *Hexahedral cell satisfying sufficient conditions* 1: (a) *edges;* (b) *faces.*



(a) (b)

FIG. 3.2. *Hexahedral cell satisfying sufficient conditions* 2: (a) *edges;* (b) *faces.*

TABLE 3.1
Corner points of cells.

Figure	3.1			3.2		
Values i_1, i_2, i_3	$z^1_{i_1 i_2 i_3}$	$z^2_{i_1 i_2 i_3}$	$z^3_{i_1 i_2 i_3}$	$z^1_{i_1 i_2 i_3}$	$z^2_{i_1 i_2 i_3}$	$z^3_{i_1 i_2 i_3}$
0,0,0	0.94058	0.95360	0.43885	0.70048	0.70080	0.19745
0,0,1	0.88002	0.45455	0.35625	0.92840	0.03615	0.11635
0,1,0	0.70707	0.93755	0.41895	0.26458	0.42831	0.00110
0,1,1	0.16146	0.78644	0.30840	0.35206	0.06901	0.44930
1,0,0	0.54506	0.73818	0.88086	0.11448	0.89730	0.36412
1,0,1	0.22399	0.30102	0.40037	0.56794	0.84961	0.67088
1,1,0	0.45779	0.94522	0.74387	0.10908	0.31775	0.30655
1,1,1	0.24743	0.61450	0.65427	0.64597	0.73848	0.99582

faces, and 14004 cases satisfied necessary conditions for the whole cell. The Jacobian was positive in 11582 cases. Sufficient conditions 1 were satisfied in 33.63% of the cases (from the number 11582), while sufficient conditions 2 were satisfied in 63.91% of the cases. In 11660 cases the Jacobian was positive on the edges (conditions (3.10)); 99.33% of them had positive Jacobian everywhere in the whole cell.

In Figures 3.1 and 3.2 there are hexahedrons nondegeneracy of which was checked by means of sufficient conditions 1 and 2, respectively. Figures 3.1(a) and 3.2(a) show the edges of cells, while Figures 3.1(b) and 3.2(b) show the faces of cells (hidden lines are removed). The corners of cells for Figures 3.1 and 3.2 are given in Table 3.1.

All computations were performed on a personal computer. A computer code first generated a hexahedron for each case and then checked different types of conditions. For all cases, necessary conditions were checked. For some cases, necessary and sufficient conditions on the edges, sufficient conditions 1, and sufficient conditions 2 were checked. Besides checking all these conditions, in some cases a special algorithm of a search for a minimum of the Jacobian in the whole cube was applied. On the personal computer Pentium 3 (800 MHz), such a test of 10^7 cases demanded about 1 minute of computation (58 seconds). The test of sufficient conditions 2 (as the most expensive) for 10^7 nondegenerate cells required about one and half minute of computation (95 seconds).

Numerical result showed the following.

1. Necessary conditions allowed the exclusion of a great number of cells which were degenerate.
2. In less than one third of cases when the Jacobian was positive at the corners of a cell the Jacobian was positive everywhere in the cell. Therefore, it would be unreliable to draw a conclusion about the invertibility of the Jacobian on the basis of positive Jacobians at the corners of a cell.
3. Necessary and sufficient conditions of positivity of the Jacobian on the edges (provided that necessary conditions (3.1) were satisfied) in a large percentage of cases gave positive Jacobians everywhere in the cell. However, these conditions did not also guarantee the invertibility of the trilinear map.
4. Sufficient conditions 2 permitted recognition of the nondegeneracy of cells in most of the cases.

4. A formula of a volume of a cell.

THEOREM 4.1. *The volume of a ruled hexahedral cell is*

$$V = \frac{1}{12}\left(\sum_{i_1,i_2,i_3=0}^{1} \alpha_{i_1i_2i_3} + \bar{\kappa}_{000} + \bar{\kappa}_{111}\right) = \frac{1}{2}\left(\sum_{i_1,i_2,i_3=0}^{1} V^{pqr}_{i_1i_2i_3} + V^{uvw}_{000} + V^{uvw}_{111}\right).$$

Proof. Integrate the Jacobian J. Using (2.17) gives

$$\begin{aligned} V &= \iiint_{0\le y^l\le 1} J dy^1 dy^2 dy^3 = \frac{1}{24}\sum_{i_1,i_2,i_3=0}^{1}\sum_{k=1}^{3}\beta^k_{i_1i_2i_3} \\ &= \frac{1}{4}\sum_{i_1,i_2,i_3=0}^{1}\left(V^{pqv}_{i_1i_2i_3} + V^{pqu}_{i_1i_2i_3} + V^{pur}_{i_1i_2i_3}\right), \end{aligned} \tag{4.1}$$

and then (2.15) implies the above formula. □

This formula requires computation of only 10 volumes of tetrahedrons.

It is possible to see that (if volumes $V^{pqr}_{i_1i_2i_3}$, V^{uvw}_{000}, and V^{uvw}_{111} are positive) the volume of the ruled cell is equal to one-half of a sum of volumes of two dodecahedrons with planar faces, with the same corners $\mathbf{z}_{i_1i_2i_3}$, edges $\mathbf{p}_{i_1i_2i_3}$, $\mathbf{q}_{i_1i_2i_3}$, $\mathbf{r}_{i_1i_2i_3}$, and with different $\mathbf{u}$, $\mathbf{v}$, $\mathbf{w}$ (or faces) (see Figure 1.1(b)).

5. Conclusions. The conditions obtained allow the establishment of nondegeneracy of a general class of cells. If necessary conditions of positivity of the Jacobian in the whole cell (or necessary and sufficient conditions of positivity of the Jacobian on the edges of the cell) are not satisfied, the cell is degenerate. If sufficient conditions 1 or 2 hold, the cell is nondegenerate. Sufficient conditions 1 demand fewer computations than sufficient conditions 2.

In the case of cells satisfying necessary conditions but not satisfying sufficient conditions 2 there is nothing for it but to find numerically the minimum of the Jacobian in the cube using, for example, one of the obtained representations. If the minimum is positive, the cell is nondegenerate. Otherwise, it is degenerate.

The formula of a volume of a cell is simple and does not demand verification of numerous conditions and computations as in [4, 9].

Acknowledgment. I devote this work to the memory of my teacher, academician of the Russian Academy of Sciences, Anatolii F. Sidorov. I am thankful to him for the discussion of part of the results.

REFERENCES

[1] O. B. Khairullina, A. F. Sidorov, and O. V. Ushakova, *Variational methods of construction of optimal grids*, in Handbook of Grid Generation, J. F. Thompson, B. K. Soni, and N. P. Weatherill, eds., CRC Press, Boca Raton, FL, 1999, pp. 36-1–36-25.

[2] O. V. Ushakova, *Algorithm of two-dimensional optimal grid generation*, in Numerical Grid Generation in Computational Field Simulation, B. K. Soni and J. F. Thompson, eds., Mississippi State University, Mississippi State, MS, 1996, pp. 37–46.

[3] O. B. Khairullina, A. F. Sidorov, and O. V. Ushakova, *Tests for two-dimensional grid generation*, in Handbook of Grid Generation, Appendix B, J. F. Thompson, B. K. Soni, and N. P. Weatherill, eds., CRC Press, Boca Raton, FL, 1999, pp. B-22–B-26.

[4] J. Killeen, *Controlled Fusion*, Academic Press, New York, San Francisco, London, 1976.

[5] O. V. Ushakova, *Conditions of nondegeneracy of three-dimensional cells: A formula of a volume of cells*, in Numerical Grid Generation in Computational Field Simulations, B. K. Soni, J. Haeuser, J. F. Thompson, and P. Eiseman, eds., International Society of Grid Generation (ISGG), Mississippi State, MS, 2000, pp. 659–668.

[6] G. Strang and G. Fix, *An Analysis of the Finite Element Method*, Prentice–Hall, New York, 1973.

[7] P. M. Knupp, *On the invertibility of isoparametric map*, Comput. Methods Appl. Mech. Engrg., 78 (1990), pp. 313–329.

[8] S. A. Ivanenko, *Harmonic mappings*, in Handbook of Grid Generation, J. F. Thompson, B. K. Soni, and N. P. Weatherill, eds., CRC Press, Boca Raton, FL, 1999, pp. 8-1–8-43.

[9] A. S. Shvedov, *Formulas of a volume of cells*, Mat. Zametki, 39 (1986), pp. 597–605.

[10] J. K. Dukowicz, *Efficient volume computation for three-dimensional hexahedral cells*, J. Comput. Phys., 74 (1988), pp. 493–496.

[11] P. M. Knupp and N. Robidoux, *A framework for variational grid generation: Conditioning the Jacobian matrix with matrix norms*, SIAM J. Sci. Comput., 21 (2000), pp. 2029–2047.

[12] S. A. Ivanenko and A. A Charakhch'yan, *Curvilinear grids of convex quadrilaterals*, U.S.S.R. Comput. Math. and Math. Phys, 28 (1988), pp. 126–133.

[13] P. M. Knupp and S. Steinberg, *Fundamentals of Grid Generation*, CRC Press, Boca Raton, FL, 1994.

SIAM J. SCI. COMPUT.
Vol. 23, No. 4, pp. 1291–1315

A MULTIGRID METHOD ENHANCED BY KRYLOV SUBSPACE ITERATION FOR DISCRETE HELMHOLTZ EQUATIONS[*]

HOWARD C. ELMAN[†], OLIVER G. ERNST[‡], AND DIANNE P. O'LEARY[§]

Abstract. Standard multigrid algorithms have proven ineffective for the solution of discretizations of Helmholtz equations. In this work we modify the standard algorithm by adding GMRES iterations at coarse levels and as an outer iteration. We demonstrate the algorithm's effectiveness through theoretical analysis of a model problem and experimental results. In particular, we show that the combined use of GMRES as a smoother and outer iteration produces an algorithm whose performance depends relatively mildly on wave number and is robust for normalized wave numbers as large as 200. For fixed wave numbers, it displays grid-independent convergence rates and has costs proportional to the number of unknowns.

Key words. Helmholtz equation, multigrid, Krylov subspace methods

AMS subject classifications. Primary, 65N55, 65F10, 65N22; Secondary, 78A45, 76Q05

PII. S1064827501357190

1. Introduction.

Multigrid algorithms are effective for the numerical solution of many partial differential equations, providing a solution in time proportional to the number of unknowns. For some important classes of problems, however, standard multigrid algorithms have not been useful, and in this paper we focus on developing effective multigrid algorithms for one such class, the discrete Helmholtz equation.

Our main interest lies in solving exterior boundary value problems of the form

$$-\Delta u - k^2 u = f \qquad \text{on } \Omega \subset \mathbb{R}^d, \tag{1.1}$$

$$Bu = g \qquad \text{on } \Gamma \subset \partial\Omega, \tag{1.2}$$

$$\frac{\partial u}{\partial n} = Mu \qquad \text{on } \Gamma_\infty \subset \partial\Omega, \tag{1.3}$$

that arise in the modeling of time-harmonic acoustic or plane-polarized electromagnetic scattering by an obstacle. The boundary Γ represents the scattering obstacle, and the boundary operator B can be chosen so that a Dirichlet, Neumann, or Robin boundary condition is imposed. The original unbounded domain is truncated to the finite domain Ω by introducing the artificial boundary Γ_∞ on which the radiation boundary condition (1.3) approximates the outgoing Sommerfeld radiation condition. Depending on what type of radiation condition is chosen, M can be either a (local) differential operator or a global integral operator coupling all points on Γ_∞ (see [16]). The data for the problem are given by the right-hand side f and the boundary data

[*] Received by the editors June 16, 1999; accepted for publication (in revised form) June 17, 2001; published electronically December 4, 2001.

http://www.siam.org/journals/sisc/23-4/35719.html

[†] Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742 (elman@cs.umd.edu). This author's work was supported in part by the National Science Foundation under grant DMS9423133.

[‡] Department of Mathematics and Computer Science, TU Bergakademie Freiberg, 09596 Freiberg, Germany (ernst@math.tu-freiberg.de). This author's work was supported in part by the National Science Foundation under grant ASC9704683 and by the University of Maryland Institute for Advanced Computer Studies.

[§] Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742 (oleary@cs.umd.edu). This author's work was supported in part by National Science Foundation grants NSF CCR-95-03126 and NSF CCR-97-32022.

g. In the most common case, $f \equiv 0$ and $-g$ is the boundary data of an incident plane wave. The critical parameter is the wave number k, which is positive in the case of unattenuated wave propagation. Due to the radiation boundary condition, the solution of (1.1)–(1.3) is a complex-valued function $u : \Omega \to \mathbb{C}$.

Discretization of (1.1)–(1.3) by finite differences or finite elements leads to a linear system of equations

$$\boldsymbol{A}\boldsymbol{u} = \boldsymbol{f} \tag{1.4}$$

in which the coefficient matrix $\boldsymbol{A}$ is complex-symmetric, i.e., not Hermitian. Moreover, for large values of the wave number k, it becomes highly indefinite.

This indefiniteness has prevented multigrid methods from being applied to the discrete Helmholtz equation with the same success as these methods have enjoyed for symmetric positive-definite problems. Some proposed multilevel strategies for the Helmholtz equation impose restrictions on the coarse grid, requiring that these grids be sufficiently fine for the algorithm to be convergent [2, 6, 10, 31, 35]; such restrictions limit the utility of these techniques. It is also possible to precondition the indefinite Helmholtz problem with preconditioners for the leading (second order) term [4, 5, 17, 36], although the effectiveness of this approach is limited for problems with large wave numbers. Our aim in this work is to identify the difficulties arising in a standard multigrid iteration for the Helmholtz equation and to analyze and test techniques designed to address these difficulties. In particular, in section 2, we show there are difficulties with both of the main multigrid components, smoothing and coarse grid correction. Standard smoothers such as Jacobi or Gauß–Seidel relaxation become unstable for indefinite problems since there are always error components—usually the smooth ones—that are amplified by these smoothers. The difficulties with the coarse grid correction are usually attributed to the poor approximation of the Helmholtz operator on very coarse meshes, since such meshes cannot adequately resolve waves with wavelength $\lambda = 2\pi/k$ of which the solution primarily consists. We show, however, that although the coarse grid correction is inaccurate when coarse grid eigenvalues do not agree well with their fine-grid counterparts, coarse meshes can still yield useful information in a multigrid cycle.

Our approach for smoothing is to use standard damped Jacobi relaxation when it works reasonably well (on fine enough grids) and then to replace this with a Krylov subspace iteration when it fails as a smoother. Earlier works by Bank [2] and Brandt and Ta'asan [12] have employed relaxation on the normal equations in this context. Krylov subspace smoothing, principally using the conjugate gradient method, has been considered by a variety of authors [3, 8, 9, 30, 32].

For coarse grid correction, we identify the type and number of eigenvalues that are handled poorly during the correction, and we remedy the difficulty by introducing an acceleration for multigrid; that is, we use multigrid as a preconditioner for an outer Krylov subspace iteration. This approach has been used by many authors; see, e.g., [7, 22, 25, 26, 31, 35]. In some settings, where it is used for positive-definite systems [7, 22], the Krylov subspace method accelerates the multigrid computations, but multigrid alone is also rapidly convergent and improvements in performance are not dramatic. A significant point in the present study is that multigrid does a poor job of eliminating some modes from the error. As a result, it converges slowly and even diverges in some cases, and the outer Krylov subspace iteration is needed for the method to be robust. (This phenomenon is also observed for the convection-diffusion equation in [25, 26].) Any Krylov subspace method is an option for both the smoother and the outer iteration; we use GMRES [28].

A different approach for adapting multigrid to the Helmholtz equation, based on representing oscillatory error components on coarse grids as the product of an oscillatory Fourier mode and a smooth amplitude—or ray—function, has been developed by Brandt and Livshits [11]. The standard V-cycle is augmented by so-called ray cycles, in which the oscillatory error components are eliminated by approximating the associated ray functions in a multigrid fashion. This wave-ray methodology has also been combined by Lee et al. [23] with a first-order system least-squares formulation for the Helmholtz equation. Results in [11, 23] suggest that these techniques may be somewhat more efficient than the methods of this paper, but they are considerably more difficult to implement, and it is unclear whether they can be generalized to handle variable coefficient problems.

An outline of the paper is as follows. In section 2, we perform a model problem analysis, using a one-dimensional problem to identify the difficulties encountered by both smoothers and coarse grid correction, and supplementing these observations with an analysis of how dimensionality of the problem affects the computations. In section 3, we present the refined multigrid algorithms and test their performance on a set of two-dimensional benchmark problems on a square domain. In particular, we demonstrate the effectiveness of an automated stopping criterion for use with GMRES smoothing, and we show that the combined use of GMRES as a smoother and outer iteration produces an algorithm whose performance depends relatively mildly on wave number and is robust for wave numbers as large as 200. In section 4, we show the performance of the multigrid solver on an exterior scattering problem. Finally, in section 5, we draw some conclusions.

2. Model problem analysis.

Most of the deficiencies of standard multigrid methods for solving Helmholtz problems can be seen from a one-dimensional model problem. Therefore, we consider the Helmholtz equation on the unit interval $(0, 1)$ with homogeneous Dirichlet boundary conditions

$$-u'' - k^2 u = f, \quad u(0) = u(1) = 0. \tag{2.1}$$

This problem is guaranteed to be nonsingular only if k^2 is not an eigenvalue of the negative Laplacian, and we will assume here that this requirement holds. The problem is indefinite for $k^2 > \pi^2$, which is the smallest eigenvalue of the negative Laplacian.

Finite difference discretization of (2.1) on a uniform grid containing N interior points leads to a linear system of equations (1.4) with the $N \times N$ coefficient matrix $\boldsymbol{A} = \boldsymbol{A}^h = (1/h^2)\,\mathrm{tridiag}(-1, 2, -1) - k^2\boldsymbol{I}$, where $h = 1/(N + 1)$ denotes the mesh width and $\boldsymbol{I}$ denotes the identity matrix. Under the assumptions on k above, it is well known (see [29]) that for sufficiently fine discretizations, the discrete problems are also nonsingular. We also assume that all coarse grid problems are nonsingular.

The eigenvalues of $\boldsymbol{A}$ are

$$\lambda_j = \frac{2(1 - \cos j\pi h)}{h^2} - k^2 = \frac{4}{h^2}\sin^2\frac{j\pi h}{2} - k^2, \qquad j = 1, \dots, N, \tag{2.2}$$

and the eigenvectors are

$$\boldsymbol{v}_j = \sqrt{2h}\,[\sin ij\pi h]_{i=1}^{N}, \quad j = 1, \dots, N. \tag{2.3}$$

The choice of Dirichlet boundary conditions in (2.1) allows us to perform Fourier analysis using these analytic expressions for eigenvalues and eigenvectors. In experiments described in section 3, we will examine how our observations coincide with performance on problems with radiation conditions, which are nonsingular for all k [20]. Aspects of the algorithm that depend on the dimensionality of the problem will be considered at the end of this section.

2.1. Smoothing. For the smoothing operator, we consider damped Jacobi relaxation, defined by the stationary iteration

$$\boldsymbol{u}_{m+1} = \boldsymbol{u}_m + \omega \boldsymbol{D}^{-1}\boldsymbol{r}_m = \boldsymbol{u}_m + \omega \boldsymbol{D}^{-1}\boldsymbol{A}\boldsymbol{e}_m,$$

where $\boldsymbol{r}_m = \boldsymbol{f} - \boldsymbol{A}\boldsymbol{u}_m$ and $\boldsymbol{e}_m = \boldsymbol{A}^{-1}\boldsymbol{f} - \boldsymbol{u}_m$ denote the residual and error vectors at step m, respectively. $\boldsymbol{D} = (2/h^2 - k^2)\boldsymbol{I}$ denotes the matrix consisting of the diagonal of $\boldsymbol{A}$, and ω is the damping parameter. The associated error propagation matrix is $\boldsymbol{S}_\omega = \boldsymbol{I} - \omega\boldsymbol{D}^{-1}\boldsymbol{A}$, and the eigenstructure of this matrix governs the behavior of the error $\boldsymbol{e}_{m+1} = \boldsymbol{S}_\omega \boldsymbol{e}_m$. Since $\boldsymbol{D}$ is a multiple of the identity matrix, $\boldsymbol{S}_\omega$ is a polynomial in $\boldsymbol{A}$ and hence shares the same system of orthonormal eigenvectors (2.3). The eigenvalues of $\boldsymbol{S}_\omega$ are

$$\mu_j = 1 - \omega\left(1 - \frac{\cos j\pi h}{1 - \frac{1}{2}k^2h^2}\right), \quad j = 1, \dots, N. \tag{2.4}$$

Thus, the eigenvalue μ_j of $\boldsymbol{S}_\omega$ is the damping factor for the error component corresponding to the eigenvalue λ_j of $\boldsymbol{A}$.

We now consider the effects of damped Jacobi smoothing on three levels of grids: fine, coarse, and intermediate.

2.1.1. Fine grids. The fine-grid mesh size is determined by accuracy requirements on the discretization, and this allows us to make certain assumptions on the size of h versus k on the fine grid. Recall that the wavelength λ associated with a time-harmonic wave with wave number $k > 0$ is given by $\lambda = 2\pi/k$. The quantity

$$\frac{\lambda}{h} = \frac{2\pi}{kh} = \frac{2\pi}{k}(N+1)$$

is the number of mesh points per wavelength. We will enforce the condition

$$\lambda/h \geq 10 \quad \text{or, equivalently,} \quad kh \leq \pi/5, \tag{2.5}$$

which is a rule of thumb for approximability of second order discretizations commonly used in engineering computations [19]. We also note that, for reasons of stability, a bound on the quantity h^2k^3 is also required [20]; for high wave numbers this bound is more restrictive than the bound on kh.

As a consequence of (2.5), the quantity multiplying the smoothing parameter ω in (2.4) will vary between about $-1/4$ and $9/4$ for $j = 1, \dots, N$, and plain Jacobi smoothing ($\omega = 1$) results in a slight amplification of the most oscillatory modes as well as of the smoothest modes. One can adjust ω so that the most oscillatory mode is damped, and this is the case as long as $\omega < \omega_1 := (4 - 2k^2h^2)/(4 - k^2h^2)$. For $\boldsymbol{S}_\omega$ to be an effective smoother, ω is usually chosen to maximize damping for the oscillatory half of the spectrum. This leads to the choice

$$\omega_0 = \frac{2 - k^2h^2}{3 - k^2h^2}, \tag{2.6}$$

which is equal to the familiar optimal value of $2/3$ for the Laplacian [24, p. 11] when $k = 0$ and equals 0.61 when $\lambda/h = 10$. However, the smoothest mode is amplified for

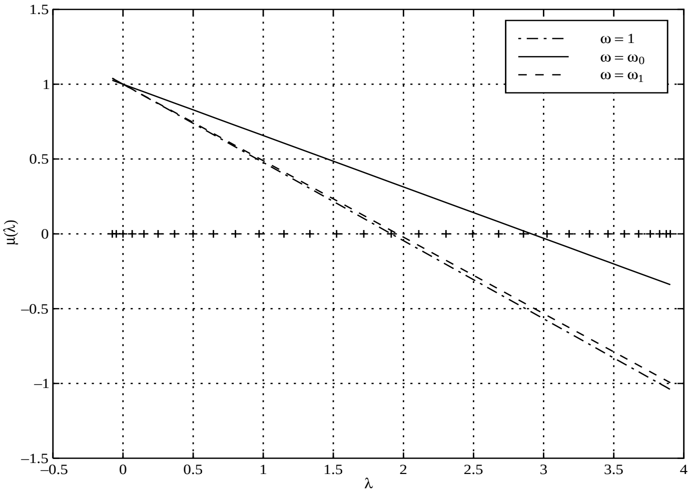


FIG. 2.1. *The damping factors for the damped Jacobi relaxation plotted against the eigenvalues of $\boldsymbol{A}$ (+) for $\omega = 1$, $\omega = \omega_0$, and $\omega = \omega_1$ ($N = 31$, $k = 3\pi$).*

any positive choice of ω when the discrete problem is indefinite, and this is the case for the discrete Helmholtz operator (1.4) when $k^2 > \pi^2$. As can be seen from (2.4), more smooth-mode eigenvalues of $\boldsymbol{S}_\omega$ become larger than one in magnitude as h is increased, thus making damped Jacobi—as well as other standard smoothers—increasingly more unstable as the mesh is coarsened.

Figure 2.1 shows the damping factors μ_j for each of the eigenvalues λ_j of $\boldsymbol{A}$ for wave number $k = 3\pi$ on a grid with $N = 31$. The maximal amplification occurs for the smoothest mode, corresponding to the leftmost eigenvalue of $\boldsymbol{A}$. When $\omega = \omega_0$ this amplification factor is approximately equal to

$$\rho = \rho(kh) = \frac{1}{1 - \frac{1}{3}k^2h^2}. \tag{2.7}$$

Figure 2.2 shows how ρ varies with kh. Limiting this largest amplification factor, say, to $\rho \le 1.1$, would lead to the mesh size restriction $kh \le 0.52$, somewhat stronger than (2.5). One also observes that, for $kh > \sqrt{6}$, this mode is once again damped.

In summary, the situation on the finest grids is similar to the positive-definite case, except for the small number of amplified smooth modes whose number and amplification factors increase as the mesh is coarsened.

2.1.2. Very coarse grids.

As the mesh is coarsened, the eigenvalues of $\boldsymbol{A}$ that correspond to the larger eigenvalues of the underlying differential operator disappear from the discrete problem, while the small ones—those with smooth eigenfunctions—remain. This means that, for a fixed k large enough for the differential problem to be indefinite, there is a coarsening level below which all eigenvalues are negative. For the model problem (2.1), this occurs for $kh > 2\cos(\pi h/2)$ for any fixed $k > \pi$. In this (negative-definite) case, the damped Jacobi iteration is convergent for $\omega \in (0, \omega_2)$, with

$$\omega_2 = \frac{2 - k^2h^2}{2\sin^2(\pi h/2) - \frac{1}{2}k^2h^2},$$

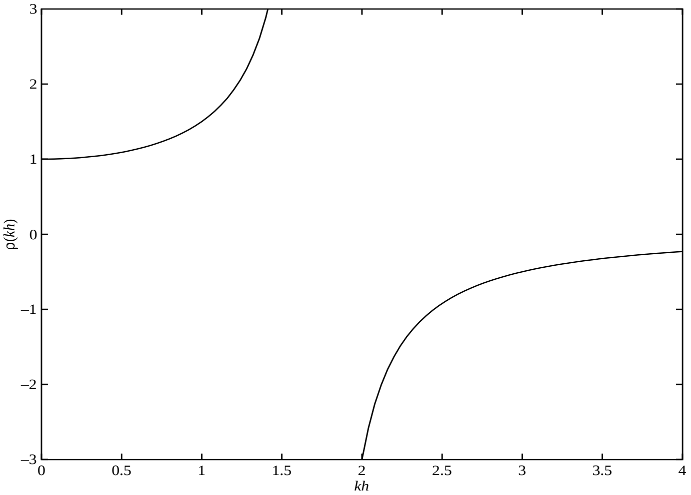


FIG. 2.2. *The variation of the damping/amplification factor of the smoothest mode as a function of kh for $\omega = \omega_0$.*

and the spectral radius of $\boldsymbol{S}_\omega$ is minimized for $\omega = 1$. This would permit the use of (undamped) Jacobi as a smoother on very coarse grids, but we shall not make use of this.

2.1.3. Intermediate grids. What remains is the difficult case: values of kh for which the problem is not yet negative definite but for which a large number of smooth modes are amplified by damped Jacobi relaxation. Jacobi smoothing and other standard smoothers are therefore no longer suitable, and it becomes necessary to use a different smoothing procedure. In [12] and [18] it was proposed to replace classical smoothers with the Kaczmarz iteration, which is Gauß–Seidel relaxation applied to the symmetric positive-definite system $\boldsymbol{A}\boldsymbol{A}^*\boldsymbol{v} = \boldsymbol{f}$ for the auxiliary variable $\boldsymbol{v}$ defined by $\boldsymbol{A}^*\boldsymbol{v} = \boldsymbol{u}$. This method has the advantage of not amplifying any modes, but it suffers from the drawback that the damping of the oscillatory modes is very weak. In the following section we propose using Krylov subspace methods such as GMRES for smoothing. These methods possess the advantage of reducing error components on both sides of the imaginary axis without resorting to the normal equations.

2.2. Coarse grid correction. The rationale behind coarse grid correction is that smooth error components can be well represented on coarser grids, and hence a sufficiently good approximation of the error can be obtained by approximating the fine-grid residual equation using the analogous system on a coarser mesh. This assumes both that the error consists mainly of smooth modes and that the solution of the coarse grid residual equation is close to its counterpart on the fine grid. In this section, we present an analysis of what goes wrong for the Helmholtz problem.

2.2.1. Amplification of certain modes. Assume the number of interior grid points on the fine grid is odd, and consider the next coarser mesh, with $n = (N-1)/2$ interior points. We identify $\mathbb{R}^N$ and $\mathbb{R}^n$, respectively, with the spaces of grid functions on these two meshes that vanish at the endpoints, and we indicate the mesh such vectors are associated with using the superscripts h and H. Let $\boldsymbol{e}^h = \boldsymbol{u} - \boldsymbol{u}^h$ denote the fine-grid error, let $\boldsymbol{r}^h = \boldsymbol{f} - \boldsymbol{A}^h\boldsymbol{u}^h$ denote the residual, and let $H = 2h$ denote the coarse mesh size. Let the coarse-to-fine transformation be given by the interpolation operator $\boldsymbol{I}_H^h : \mathbb{R}^n \to \mathbb{R}^N$,

$$\left[\boldsymbol{I}_H^h \boldsymbol{w}^H\right]_i := \begin{cases} \boldsymbol{w}_{i/2}^H, & i \text{ even}, \\ \frac{1}{2}[\boldsymbol{w}_{(i-1)/2}^H + \boldsymbol{w}_{(i+1)/2}^H], & i \text{ odd}, \end{cases} \qquad i = 1, \dots, N.$$

The following indication of what can go wrong with the (exact) coarse grid correction was given in [12]: consider a fine-grid error $\boldsymbol{e}^h = \boldsymbol{v}^h$ consisting of only the smoothest eigenvector $\boldsymbol{v}^h$ of $\boldsymbol{A}^h$ with associated eigenvalue λ^h. The fine-grid residual is thus given by $\boldsymbol{r}^h = \boldsymbol{A}^h \boldsymbol{e}^h = \lambda^h \boldsymbol{v}^h$, and, since we are assuming that $\boldsymbol{v}^h$ is smooth, its restriction $\hat{\boldsymbol{r}}^H := \boldsymbol{I}_h^H \boldsymbol{r}^h = \lambda^h \boldsymbol{I}_h^H \boldsymbol{v}^h$ to the coarse grid will again be close to an eigenvector of the coarse grid operator $\boldsymbol{A}^H$ but with respect to a slightly different eigenvalue λ^H. The coarse grid version of the correction is

$$\boldsymbol{e}^H = (\boldsymbol{A}^H)^{-1} \hat{\boldsymbol{r}}^H = \lambda^h (\boldsymbol{A}^H)^{-1} \boldsymbol{I}_h^H \boldsymbol{v}^h \approx \frac{\lambda^h}{\lambda^H} \boldsymbol{I}_h^H \boldsymbol{v}^h.$$

Hence the error on the fine grid after the correction is

$$\boldsymbol{e}^h - \boldsymbol{I}_H^h \boldsymbol{e}^H \approx \boldsymbol{v}^h - \frac{\lambda^h}{\lambda^H} \boldsymbol{I}_H^h \boldsymbol{I}_h^H \boldsymbol{v}^h = \left(1 - \frac{\lambda^h}{\lambda^H}\right) \boldsymbol{v}^h, \tag{2.8}$$

where we have assumed that the smooth mode $\boldsymbol{v}^h$ is invariant under restriction followed by interpolation. This tells us that, under the assumption that the restrictions of smooth eigenvectors are again eigenvectors of $\boldsymbol{A}^H$, the quality of the correction depends on the ratio λ^h/λ^H. If the two are equal, then the correction is perfect, but if the relative error is large, the correction can be arbitrarily bad. This occurs whenever one of λ^h, λ^H is close to the origin and the other is not. Moreover, if λ^h and λ^H have opposite signs, then the correction is in the wrong direction.

We now go beyond existing analysis and examine which eigenvalues are problematic in this sense for finite differences; a similar analysis can also be performed for linear finite elements. Consider the coarse grid eigenfunctions $\boldsymbol{v}_j^H = [\sin ij\pi H]_{j=1}^n$. To understand the effects of interpolation of these grid functions to the fine grid, we must examine both the first n fine-grid eigenfunctions $\{\boldsymbol{v}_j^h\}_{j=1}^n$ and their *complementary modes* $\{\boldsymbol{v}_{N+1-j}^h\}_{j=1}^n$; these are related by $\left[\boldsymbol{v}_{N+1-j}^h\right]_i = (-1)^{i+1} \left[\boldsymbol{v}_j^h\right]_i$. As is easily verified, there holds [13]

$$\boldsymbol{I}_H^h \boldsymbol{v}_j^H = c_j^2 \boldsymbol{v}_j^h - s_j^2 \boldsymbol{v}_{N+1-j}^h, \qquad j = 1, \dots, n, \tag{2.9}$$

with $c_j := \cos j\pi h/2$ and $s_j := \sin j\pi h/2$, $j = 1, \dots, N$.

If full weighting is used for the restriction operator $\boldsymbol{I}_h^H : \mathbb{R}^N \to \mathbb{R}^n$, we have componentwise

$$\left[\boldsymbol{I}_h^H \boldsymbol{u}^h\right]_i := \frac{1}{4} \left(\left[\boldsymbol{u}^h\right]_{2i-1} + 2 \left[\boldsymbol{u}^h\right]_{2i} + \left[\boldsymbol{u}^h\right]_{2i+1} \right), \qquad i = 1, \dots, n,$$

and the relation $\boldsymbol{I}_H^h = 2 \left(\boldsymbol{I}_h^H\right)^\top$. The following mapping properties are easily established:

$$\boldsymbol{I}_h^H \boldsymbol{v}_j^h = \begin{cases} c_j^2 \boldsymbol{v}_j^H, & j = 1, \dots, n, \\ 0, & j = n+1, \\ -c_j^2 \boldsymbol{v}_{N+1-j}^H, & j = n+2, \dots, N, \end{cases} \tag{2.10}$$

with c_j and s_j as defined above.

If $\boldsymbol{A}^H$ denotes the coarse grid discretization matrix, then the corrected iterate $\tilde{\boldsymbol{u}}^h := \boldsymbol{u}^h + \boldsymbol{I}_H^h(\boldsymbol{A}^H)^{-1}\boldsymbol{r}^H$ possesses the error propagation operator $\boldsymbol{C} := \boldsymbol{I} - \boldsymbol{I}_H^h(\boldsymbol{A}^H)^{-1}\boldsymbol{I}_h^H\boldsymbol{A}^h$. Denoting the eigenvalues of $\boldsymbol{A}^h$ and $\boldsymbol{A}^H$ by $\{\lambda_j^h\}_{j=1}^N$ and $\{\lambda_j^H\}_{j=1}^n$, respectively, we may summarize the action of $\boldsymbol{C}$ on the eigenvectors using (2.9) and (2.10) as follows.

THEOREM 2.1. *The image of the fine-grid eigenfunctions* $\{\boldsymbol{v}_h^h\}_{j=1}^N$ *under the error propagation operator* $\boldsymbol{C}$ *of the exact coarse grid correction is given by*

$$\boldsymbol{C}\boldsymbol{v}_j^h = \begin{cases} \left(1 - c_j^4\frac{\lambda_j^h}{\lambda_j^H}\right)\boldsymbol{v}_j^h + s_j^2c_j^2\frac{\lambda_j^h}{\lambda_j^H}\boldsymbol{v}_{N+1-j}^h, & j = 1,\dots,n, \\ \boldsymbol{v}_{n+1}^h, & j = n+1, \\ \left(1 - c_j^4\frac{\lambda_j^h}{\lambda_{N+1-j}^H}\right)\boldsymbol{v}_j^h + s_j^2c_j^2\frac{\lambda_j^h}{\lambda_{N+1-j}^H}\boldsymbol{v}_{N+1-j}^h, & j = n+2,\dots,N. \end{cases} \tag{2.11}$$

As a consequence, the two-dimensional spaces spanned by a smooth mode and its complementary mode are invariant under $\boldsymbol{C}$: $\boldsymbol{C}[\boldsymbol{v}_j^h, \boldsymbol{v}_{N+1-j}^h] = [\boldsymbol{v}_j^h, \boldsymbol{v}_{N+1-j}^h]\boldsymbol{C}_j$ with

$$\boldsymbol{C}_j := \begin{bmatrix} 1 - c_j^4\frac{\lambda_j^h}{\lambda_j^H} & c_j^2s_j^2\frac{\lambda_{N+1-j}^h}{\lambda_j^H} \\ s_j^2c_j^2\frac{\lambda_j^h}{\lambda_j^H} & 1 - s_j^4\frac{\lambda_{N+1-j}^h}{\lambda_j^H} \end{bmatrix}, \quad j = 1,\dots,n. \tag{2.12}$$

The following result shows the dependence of the matrices $\boldsymbol{C}_j$ on kh.

THEOREM 2.2. *Using the notation defined above, there holds*

$$\boldsymbol{C}_j = \begin{bmatrix} s_j^2\left(1 - \frac{k^2c_j^2}{\lambda_j^H}\right) & c_j^2\left(1 + \frac{k^2c_j^2}{\lambda_j^H}\right) \\ s_j^2\left(1 + \frac{k^2s_j^2}{\lambda_j^H}\right) & c_j^2\left(1 - \frac{k^2s_j^2}{\lambda_j^H}\right) \end{bmatrix}, \qquad j = 1,\dots,n. \tag{2.13}$$

Moreover,

$$\lim_{kh\to 0}\boldsymbol{C}_j = \begin{bmatrix} s_j^2 & c_j^2 \\ s_j^2 & c_j^2 \end{bmatrix}, \quad \lim_{kh\to\infty}\boldsymbol{C}_j = \begin{bmatrix} s_j^2(1+c_j^2) & s_j^2c_j^2 \\ s_j^2c_j^2 & c_j^2(1+s_j^2) \end{bmatrix}, \qquad j = 1,\dots,n. \tag{2.14}$$

Proof. Both (2.13) and (2.14) are simple consequences of (2.12) and the representation (2.2) of the eigenvalues λ_j^h. □

Application of the error propagation operator to a smooth mode $\boldsymbol{v}_j^h$ gives

$$\boldsymbol{C}\boldsymbol{v}_j^h = \boldsymbol{C}[\boldsymbol{v}_j^h, \boldsymbol{v}_{N+1-j}^h]\begin{pmatrix} 1 \\ 0 \end{pmatrix} = [\boldsymbol{v}_j^h, \boldsymbol{v}_{N+1-j}^h]\boldsymbol{C}_j\begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

If the entries of the first column of $\boldsymbol{C}_j$ are small, then this mode is damped by the coarse grid correction. However, if the $(1,1)$-entry is large, then this mode is amplified; and if the $(2,1)$-entry is large (somewhat less likely), then the smooth mode is corrupted by its complementary mode. As seen from (2.13), these difficulties occur whenever λ_j^H is small in magnitude. From the limits (2.14), it is evident that no such problems arise in the symmetric positive-definite case (a fact that is well known), but they also do not occur when kh is very large, i.e., when the coarse grid Helmholtz operator is negative definite. These observations can be extended by returning to (2.8) and using (2.2), wherein it holds that

$$\frac{\lambda_j^h}{\lambda_j^H} = \frac{4s_j^2/h^2 - k^2}{4s_j^2c_j^2/h^2 - k^2} = 1 + \frac{s_j^4}{s_j^2c_j^2 - (kh/2)^2}. \tag{2.15}$$

That is, the coarse grid correction strongly damps smooth error modes for either very small or very large values of kh, but it may fail to do so in the intermediate range where $s_j^2c_j^2 \approx (kh/2)^2$ for some index j associated with a smooth mode.

We also note that in the limit $k = 0$ the eigenvalues of $\boldsymbol{C}_j$ are 0 and 1, so that $\boldsymbol{C}_j$ is a projection, and in this case the projection is orthogonal with respect to the inner product induced by the symmetric and positive-definite operator $\boldsymbol{A}^h$. The projection property is lost for $k > 0$, since the coarse grid operator as we have defined it fails to satisfy the Galerkin condition $\boldsymbol{A}^H = \boldsymbol{I}_h^H\boldsymbol{A}^h\boldsymbol{I}_H^h$. (The Galerkin condition is, however, satisfied, e.g., for finite element discretizations with interpolation by inclusion.) Moreover, regardless of the type of discretization, the term $\boldsymbol{A}^h$-orthogonality ceases to makes sense once k is sufficiently large that $\boldsymbol{A}^h$ is indefinite.

2.2.2. Number of sign changes. In this section, we determine the number of eigenvalues that undergo a sign change during the coarsening process. In light of the discussion above, this number gives an indication of the number of smooth modes that are not eliminated from the error by coarse grid correction. Since these modes are not properly handled by the multigrid algorithm, another construction is needed to reduce the error associated with them. In the next section, we will introduce an *outer* Krylov subspace iteration designed for this task.

This is the only aspect of the algorithm that significantly depends on the dimensionality of the problem. Thus, here we are considering the Helmholtz equation (1.1) on the d-dimensional unit cube $(0,1)^d$, $d = 1$, 2 or 3, with homogeneous Dirichlet boundary conditions. We consider standard finite differences (second order three-point, five-point, or seven-point discretizations of the Laplacian in one, two, or three dimensions, respectively), as well as the class of low order finite elements consisting of linear, bilinear, or trilinear elements.

We first state the issue more precisely using finite differences. In d dimensions, the eigenvalues of the discrete operator on a grid with mesh size h and N grid points in each direction are

$$\lambda_{\mathcal{I}}^h = \sum_{i=1}^{d} \frac{4}{h^2}\left(\sin^2\frac{j_i\pi h}{2}\right) - k^2, \quad \mathcal{I} = \{j_1, \dots, j_d\}, \quad j_i = 1, \dots N. \tag{2.16}$$

For any fixed multi-index $\mathcal{I}$, this eigenvalue is a well-defined function of h that converges to the corresponding eigenvalue of the differential operator as $h \to 0$. Our concern is the indices for which this function changes sign, for these are the troublesome eigenvalues that are not treated correctly by some coarse grid correction. As the mesh is coarsened, the oscillatory modes ($j_i > N/2$ for some i) are not represented on the next coarser mesh, but the smooth-mode eigenvalues $\{\lambda_{\mathcal{I}}^H\}$ are slightly shifted to the left with respect to their fine-grid counterparts $\{\lambda_{\mathcal{I}}^h\}$, and some of these eigenvalues change sign at some point during the coarsening process.

The following theorem gives a bound, as a function of k, on the maximal number eigenvalue sign changes occurring on all grids.

Theorem 2.3. *For finite difference discretization of the Helmholtz equation with Dirichlet boundary conditions on the unit cube in d dimensions ($d = 1, 2, 3$), the* *number of eigenvalues that undergo a change in sign during the multigrid coarsening process is bounded above by*

$$\begin{cases} k\left(\frac{1}{2}-\frac{1}{\pi}\right) \approx 0.18\,k, & d=1, \\ k^2\left(\frac{1}{8}-\frac{1}{4\pi}\right) \approx 0.045\,k^2, & d=2, \\ k^3\left(\frac{1}{24\sqrt{3}}-\frac{1}{6\pi^2}\right) \approx 0.0072\,k^3, & d=3. \end{cases} \tag{2.17}$$

For the finite element discretizations, the number of sign changes is bounded above by

$$\begin{cases} k\left(\frac{1}{\pi}-\frac{1}{\sqrt{12}}\right) \approx .030\,k, & d=1, \\ k^2\left(\frac{1}{4\pi}-\frac{1}{24}\right) \approx .038\,k^2, & d=2, \\ k^3\left(\frac{1}{6\pi^2}-\frac{1}{216}\right) \approx .012\,k^3, & d=3. \end{cases} \tag{2.18}$$

Proof. For finite differences, let η^-_{fine} denote the number of negative eigenvalues on some given fine grid, and let η^-_{lim} denote the number of negative eigenvalues of the continuous Helmholtz operator. Because eigenvalues (2.16) with the same index $\mathcal{I}$ shift from right to left with grid coarsening, it follows that

$$\eta^-_{\text{lim}} \le \eta^-_{\text{fine}}\,; \tag{2.19}$$

this is an equality for all fine enough grids, as the discrete eigenvalues tend to the continuous ones. To identify η^-_{lim}, consider the continuous eigenvalues

$$\begin{aligned} \lambda_\ell &= \pi^2\ell^2 - k^2, & \ell &\in \mathbb{N} & (d&=1), \\ \lambda_{\ell,m} &= \pi^2(\ell^2+m^2) - k^2, & \ell,m &\in \mathbb{N} & (d&=2), \\ \lambda_{\ell,m,n} &= \pi^2(\ell^2+m^2+n^2) - k^2, & \ell,m,n &\in \mathbb{N} & (d&=3). \end{aligned}$$

It is convenient to view the indices of these eigenvalues as lying in the positive orthant of a d-dimensional coordinate system. The negative eigenvalues are contained in the intersection of this orthant with a d-dimensional sphere of radius k/π centered at the origin. Let $\mathcal{N}$ denote this intersection, and let $\hat{\mathcal{N}}$ denote the d-dimensional cube enclosing $\mathcal{N}$. The number of indices in $\hat{\mathcal{N}}$ is $\lfloor k/\pi \rfloor^d$, and the number in $\mathcal{N}$ is $\rho\lfloor k/\pi \rfloor^d$, where

$$\rho = \begin{cases} \left(\frac{k}{\pi}\right) / \left(\frac{k}{\pi}\right) = 1, & d=1, \\ \frac{1}{4}\pi\left(\frac{k}{\pi}\right)^2 / \left(\frac{k}{\pi}\right)^2 = \frac{\pi}{4}, & d=2, \\ \frac{1}{8}\cdot\frac{4}{3}\pi\left(\frac{k}{\pi}\right)^3 / \left(\frac{k}{\pi}\right)^3 = \frac{\pi}{6}, & d=3 \end{cases}$$

is the ratio of the volume of $\mathcal{N}$ to that of $\hat{\mathcal{N}}$. It follows that

$$\eta^-_{\text{lim}} = \begin{cases} k\cdot\frac{1}{\pi}, & d=1, \\ k^2\cdot\frac{1}{4\pi}, & d=2, \\ k^3\cdot\frac{1}{6\pi^2}, & d=3. \end{cases} \tag{2.20}$$

Now consider the eigenvalues of discrete problems. Again, since sign changes occur from right to left with coarsening, the mesh size that yields the maximum number of negative eigenvalues is the smallest value h for which the discrete operator is negative semidefinite. With N mesh points in each coordinate direction, this is equivalent to

$$d \sin^2 \frac{N\pi h}{2} = \left(\frac{kh}{2}\right)^2, \qquad d = 1, 2, 3.$$

Thus, $h = 2\sqrt{d}/k$, and

$$\eta^-_{\max} = \left(\frac{k}{2\sqrt{d}}\right)^d = \begin{cases} k \cdot \frac{1}{2}, & d = 1, \\ k^2 \cdot \frac{1}{8}, & d = 2, \\ k^3 \cdot \frac{1}{24\sqrt{3}}, & d = 3. \end{cases}$$

Combining (2.19) with the fact that $\eta^-_{\text{fine}} \le \eta^-_{\max}$, it follows that

$$\eta^-_{\max} - \eta^-_{\text{fine}} \le \eta^-_{\max} - \eta^-_{\lim}.$$

The latter difference, shown in (2.17), is then a bound on number of sign changes.

For finite elements, we are concerned with the eigenvalues of the coefficient matrix $\boldsymbol{A}^h$, but it is also convenient to consider the associated operator $\mathcal{A}^h$ defined on the finite element space V^h. The eigenvalues of $\mathcal{A}^h$ are those of the generalized matrix eigenvalue problem

$$\boldsymbol{A}^h \boldsymbol{u}^h = \sigma^h \boldsymbol{M}^h \boldsymbol{u}^h, \tag{2.21}$$

where $\boldsymbol{M}^h$ is the mass matrix. These eigenvalues tend to those of the continuous operator. Moreover, since V^H is a subspace of V^h, the Courant–Fischer min-max theorem implies that eigenvalues σ^h and σ^H with a common index shift to the right with coarsening (or to the left with refinement). In addition, since $\boldsymbol{M}^h$ is symmetric positive definite, Sylvester's inertia theorem implies that the number of negative eigenvalues of $\boldsymbol{A}^h$ is the same as that of (2.21). It follows from these observations that the maximal number of negative eigenvalues of $\boldsymbol{A}^h$ is bounded above by the fine-grid limit $\eta^-_{\lim}$.

This is also a bound on the number of sign changes. It can be improved by examining the eigenvalues of $\boldsymbol{A}^h$ more closely. Using the tensor product form of the operators, we can express these eigenvalues as

$$\lambda^h_{j_1,j_2,j_3} = \kappa_{j_1}\mu_{j_2}\mu_{j_3} + \mu_{j_1}\kappa_{j_2}\mu_{j_3} + \mu_{j_1}\mu_{j_2}\kappa_{j_3} - k^2\mu_{j_1}\mu_{j_2}\mu_{j_3}, \tag{2.22}$$

where $h = 1/(N+1)$, the indices j_1, j_2, j_3 run from 1 to N, and

$$\kappa_j = \frac{1}{h}(2 - 2\cos j\pi h), \qquad \mu_j = \frac{h}{6}(4 + 2\cos j\pi h). \tag{2.23}$$

Consider the requirement $\lambda^h_{j_1,j_2,j_3} \le 0$ for all indices j_1, j_2, j_3, so that $\boldsymbol{A}^h$ is negative semidefinite. This is equivalent to

$$\frac{\kappa_{j_1}}{\mu_{j_1}} + \frac{\kappa_{j_2}}{\mu_{j_2}} + \frac{\kappa_{j_3}}{\mu_{j_3}} \le k^2.$$

Since the expression κ_j/μ_j is monotonically increasing with j, the largest eigenvalue in d dimensions equals zero if

$$k^2 = d \cdot \frac{\kappa_N}{\mu_N} = d \cdot \frac{6}{h^2}\frac{1 - \cos N\pi h}{2 + \cos N\pi h} \approx d \cdot \frac{12}{h^2},$$

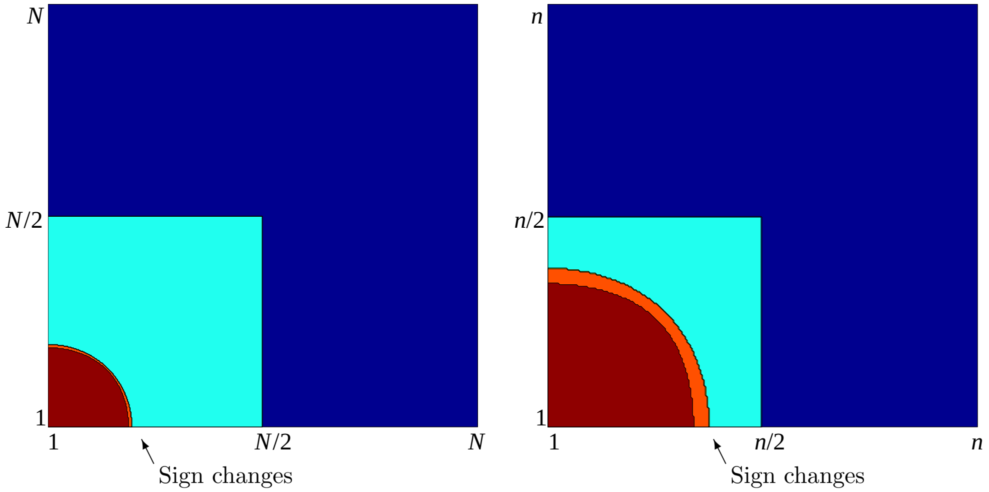


Fig. 2.3. *Indices of eigenvalues undergoing a sign change during coarsening of an $N \times N$ finite element grid with $kh = \pi/5$, $kH = 2\pi/5$ (left) and during further coarsening of the next coarser ($n \times n$) grid with $kh = 2\pi/5$, $kH = 4\pi/5$ (right).*

i.e., $h = \sqrt{12d}/k$. For this value of h, there are $\eta^-_{\max} = (k/\sqrt{12d})^d$ negative eigenvalues, and, on coarser meshes, the problem remains negative definite. Consequently, none of these $\eta^-_{\max}$ quantities undergo a sign change, giving the bound $\eta^-_{\lim} - \eta^-_{\max}$ of (2.18). □

Figure 2.3 gives an idea of how sign changes are distributed for bilinear elements in two dimensions. At levels where the changes take place, the indices of the eigenvalues lie in a curved strip in the two-dimensional plane of indices. Typically, there is one level where the majority of sign changes occur. As k is increased and h decreased correspondingly via (2.5), the shape of these strips remains fixed, but the number of indices contained in them grows like $O(h^{-d}) = O(k^d)$. (Note, however, that (2.5) is not needed for the analysis.) The behavior for finite differences is similar.

The remedy suggested in [12] for these difficulties consists of maintaining an approximation of the eigenspace V^H of the troublesome eigenvalues. A projection scheme is then used to orthogonalize the coarse grid correction against V^H, and the coefficients of the solution for this problematic space are obtained separately. Since it involves an explicit separate treatment of the problematic modes, this approach is restricted to cases where there are only very a small number of these.

Finally, we note that although the closed forms for the eigenvalues studied here are restricted to rectangular domains, we expect the trends displayed to be general. For example, the direction of shifts of eigenvalues, derived from the form of the matrices for finite differences and from the Courant–Fischer theorem for bilinear elements, will be the same for general domains. This assertion is also largely borne out by experiments on a nonsquare domain shown in section 4.

3. Incorporation of Krylov subspace methods. In view of the observations about smoothing in section 2.1 and coarse grid correction in section 2.2, we modify the standard multigrid method in the following way to treat Helmholtz problems:

- To obtain smoothers that are stable and still provide a strong reduction of oscillatory components, we use Krylov subspace iteration such as GMRES as smoothers on intermediate grids.

- To handle modes with eigenvalues that are either close to the origin on all grids—and hence belong to modes not sufficiently damped on any grid—or that cross the imaginary axis and are thus treated incorrectly by some coarse grid corrections, we add an outer iteration; that is, we use multigrid as a preconditioner for a GMRES iteration for (1.4).

We will demonstrate the effectiveness of this approach with a series of numerical experiments. In all tests the outer iteration is run until the stopping criterion

$$\|\boldsymbol{r}_m\|/\|\boldsymbol{r}_0\| < 10^{-6}$$

is satisfied, where $\boldsymbol{r}_m = \boldsymbol{f} - \boldsymbol{A}\boldsymbol{u}_m$ is the residual of the mth GMRES iterate and the norm is the vector Euclidean norm. The multigrid algorithm is a V-cycle in all cases; the smoothing schedules are specified below.

3.1. GMRES accelerated multigrid.

We begin with an experiment for the one-dimensional Helmholtz equation on the unit interval with forcing term $f = 0$ and inhomogeneous Dirichlet boundary condition $u(0) = 1$ on the left and Sommerfeld condition on the right. We discretize using linear finite elements on a uniform grid, where the discrete right-hand side $\boldsymbol{f}$ is determined by the boundary conditions. We apply both a V-cycle multigrid algorithm and a GMRES iteration preconditioned by the same V-cycle multigrid method. The smoother in these tests is one step of damped Jacobi iteration for both presmoothing and postsmoothing, using $\omega = (12 - 4k^2h^2)/(18 - 3k^2h^2)$, the analogue of (2.6) for finite element discretization that provides maximal damping of oscillatory modes. The initial guess was a vector with normally distributed entries of mean zero and variance one, generated by the Matlab function `randn`.

Table 3.1 shows the iteration counts for increasing numbers of levels beginning with fine grids containing $N = 256$, 512, and 1024 elements, for wave numbers $k = 4\pi$ and $k = 8\pi$, which correspond to two and four wavelengths in the unit interval, respectively.

Table 3.1

Iteration counts for multigrid V-cycle as an iteration and as a preconditioner for GMRES applied to the one-dimensional model Helmholtz problem with damped Jacobi smoothing. A dash denotes divergence of the iteration.

	256 elements				512 elements				1024 elements			
	$k = 4\pi$		$k = 8\pi$		$k = 4\pi$		$k = 8\pi$		$k = 4\pi$		$k = 8\pi$	
# levels	MG	GMRS	MG	GMRS	MG	GMRS	MG	GMRS	MG	GMRS	MG	GMRS
2	7	3	7	4	7	3	7	3	7	3	7	3
3	7	5	7	6	7	5	7	5	7	5	7	5
4	7	6	9	7	7	5	7	6	7	5	7	5
5	7	6	76	10	7	6	8	7	7	5	7	6
6	16	8	–	13	7	6	69	10	7	6	7	7
7	–	9	–	16	13	8	–	13	7	6	50	10
8	–	11	–	16	–	9	–	16	9	8	–	12
9	–	11	–	17	–	11	–	16	–	9	–	16
10					–	11	–	17	–	11	–	16

We observe first that both methods display typical h-independent multigrid behavior until the mesh size on the coarsest grid reaches $kh \approx \pi/2$. (With 256 elements, this occurs for $k = 4\pi$ with 6 levels, coarsest mesh $h = 1/8$, and for $k = 8\pi$ with 5 levels, coarsest $h = 1/16$.) At this point both methods require noticeably more iterations, the increase being much more pronounced in the stand-alone multigrid case.

Table 3.2

Iteration counts for the two-dimensional problem for fine grids with $k = 4\pi$ and $k = 8\pi$ on 128×128 and 256×256 meshes. A dash denotes divergence of the iteration.

	128 × 128 elements				256 × 256 elements				512 × 512 elements			
	$k = 4\pi$		$k = 8\pi$		$k = 4\pi$		$k = 8\pi$		$k = 4\pi$		$k = 8\pi$	
# levels	MG	GMRS	MG	GMRS	MG	GMRS	MG	GMRS	MG	GMRS	MG	GMRS
2	12	7	12	7	12	7	12	7	12	6	12	6
3	12	7	12	7	12	7	12	7	12	6	12	6
4	12	7	22	11	12	7	12	8	12	6	12	7
5	13	8	–	33	12	7	21	11	12	7	21	7
6	–	15	–	64	12	8	–	34	12	7	–	11
7	–	19	–	64	–	15	–	64	12	8	–	33
8					–	19	–	63	–	15	–	63

When yet coarser levels are added, multigrid diverges, whereas the multigrid preconditioned GMRES method again settles down to an h-independent iteration count, which does, however, increase with k.

Table 3.2 shows the same iteration counts for the two-dimensional Helmholtz problem on the unit square with a second order absorbing boundary condition (see [1, 14]) imposed on all four sides and discretized using bilinear quadrilateral finite elements on a uniform mesh. Since the problem cannot be forced with a radiation condition on the entire boundary, in this and the remaining examples of section 3, an inhomogeneity was imposed by choosing a discrete right-hand side consisting of a random vector with mean zero and variance one, generated by `randn`. The initial guess was identically zero. (Trends for problems with smooth right-hand sides were the same.) In addition, for all two-dimensional problems, we use two Jacobi pre- and postsmoothing steps whenever Jacobi smoothing is used. The damping parameter ω is chosen to maximize damping of the oscillatory modes. For the grids on which we use damped Jacobi smoothing this optimum value was determined to be $\omega = 8/9$. The results show the same qualitative behavior as for the one-dimensional problem, in that multigrid begins to diverge as coarse levels are added while the GMRES-accelerated iteration converges in an h-independent number of iterations growing with k, although with a larger number of iterations than in the one-dimensional case.

A natural question is whether corrections computed on the very coarse grids, in particular those with meshes containing fewer than ten points per wavelength ($2\pi/k < 10$), make any contribution at all towards reducing the error. We investigate this in a sequence of experiments with GMRES accelerated multigrid with $k = 8\pi$, where we omit all calculations—be they smoothing or direct solves—on an increasing number of coarse levels.

The results, for a one-dimensional example with 512 elements and a two-dimensional example with 256^2 elements are in Table 3.3. The leftmost entries of the table show the iteration counts when no coarse grid information is used, i.e., for GMRES with preconditioning by two steps of Jacobi iteration. Reading from left to right, subsequent entries show the counts when smoothings on a succession of coarser grids are included, but no computations are done at grid levels below that of the coarsest grid. For the rightmost entry, a direct solve was done on the coarsest mesh; this is a full V-cycle computation. The results in two dimensions indicate improved performance for coarse grids with mesh width less than 1/16, which corresponds to four points per wave; performance degrades, but not dramatically so, for coarser meshes. For the one-dimensional test, the coarse grid at level two, which has only two points per wavelength, still accelerates the outer iteration. These results also show that multigrid

Table 3.3

Iteration counts when varying amount of course grid information is used. Table entries further to the right result from using more multigrid levels than those to the left. Test problems are $k = 8\pi$ with 512 elements in the one-dimensional example and 256^2 elements in the two-dimensional example.

		No V-Cycle ↓							Full V-cycle ↓
1D	h^{-1} for coarsest grid	512	256	128	64	32	16	8	4
	GMRES iterations	152	78	42	25	18	17	16	16
2D	h^{-1} for coarsest grid	256	128	64	32	16	8	4	2
	GMRES iterations	387	173	89	57	52	64	63	63

is dramatically superior to simple Jacobi-preconditioned GMRES.

These results show that, although multigrid by itself may diverge, it is nevertheless a powerful enough preconditioner for GMRES to converge in an h-independent number of steps. Two additional questions are whether replacing the unstable Jacobi smoother with a Krylov subspace iteration leads to a convergent multigrid method and how sensitive convergence behavior is as a function of the wave number k. We address the former in the following section.

3.2. GMRES as a smoother.

In this section we replace the unstable Jacobi smoother with GMRES smoothing. We use GMRES on all levels j where $kh_j \geq 1/2$ and continue using damped Jacobi relaxation when $kh_j < 1/2$. This choice is motivated by the discussion at the end of section 2.1.1, and it ensures that the largest amplification factor for the Jacobi smoother does not become too large. The results of section 2.1.2 show that we could switch back to Jacobi smoothing for very coarse meshes, but we have not explored this option.

3.2.1. Nonconstant preconditioners.

This introduces a slight complication with regard to the *outer* GMRES iteration when multigrid is used as a preconditioner. The inner GMRES smoothing steps are not linear iterations, and therefore a different preconditioner is being applied at every step of the outer iteration. A variant of GMRES able to accommodate a changing preconditioner (known as flexible GMRES (FGMRES)) is due to Saad [27]. It requires the following minor modification of the standard (right preconditioned) GMRES algorithm: if the orthonormal basis of the $(m+1)$st Krylov space $\mathcal{K}_{m+1}(\boldsymbol{A}\boldsymbol{M}^{-1}, \boldsymbol{r}_0)$ in the case of a constant preconditioner $\boldsymbol{M}$ is denoted by $\boldsymbol{V}_{m+1} = [\boldsymbol{v}_1, \dots, \boldsymbol{v}_{m+1}]$, then the Arnoldi relation $\boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{V}_m = \boldsymbol{V}_{m+1}\tilde{\boldsymbol{H}}_m$ holds with an $(m+1) \times m$ upper Hessenberg matrix $\tilde{\boldsymbol{H}}_m$. If the preconditioning and matrix multiplication step

$$\boldsymbol{z}_m := \boldsymbol{M}^{-1}\boldsymbol{v}_m, \qquad \boldsymbol{w} := \boldsymbol{A}\boldsymbol{z}_m,$$

is performed with a changing preconditioner $\boldsymbol{M} = \boldsymbol{M}_m$, this results in the modified Arnoldi relation

$$\boldsymbol{A}\boldsymbol{Z}_m = \boldsymbol{V}_{m+1}\tilde{\boldsymbol{H}}_m,$$

where $\boldsymbol{Z}_m = [\boldsymbol{z}_1, \dots, \boldsymbol{z}_m]$. The residual vector is now minimized over the space $\operatorname{span}\{\boldsymbol{z}_1, \dots, \boldsymbol{z}_m\}$, which need no longer be a Krylov space. This requires storing the vectors $\{\boldsymbol{z}_j\}$ in addition to the orthonormal vectors $\{\boldsymbol{v}_j\}$, which form a basis of $\operatorname{span}\{\boldsymbol{A}\boldsymbol{z}_j : j = 1, \dots m\}$.

Table 3.4

Manually optimized GMRES smoothing schedule for the two-dimensional model Helmholtz problem: "J" denotes Jacobi smoothing and "D" denotes a direct solve. The FGMRES algorithm uses the multigrid V-cycle as a preconditioner.

256×256, $k = 4\pi$, $o_{\max} = 6$

# levels	Smoothing schedule	MG	FGMRES
6	J J J J 13 D	8	6
7	J J J J 13 16 D	8	6
8	J J J J 13 16 1 D	8	6
9	J J J J 13 16 1 0 D	8	6

128×128, $k = 8\pi$, $o_{\max} = 7$

# levels	Smoothing schedule	MG	FGMRES
4	J J 25 D	9	7
5	J J 25 39 D	9	7
6	J J 25 39 24 D	9	7
7	J J 25 39 16 2 D	9	7
8	J J 25 39 16 2 0 D	9	7

256×256, $k = 8\pi$, $o_{\max} = 7$

# levels	Smoothing schedule	MG	FGMRES
5	J J J 23 D	9	7
6	J J J 23 42 D	9	7
7	J J J 23 39 0 D	9	7
8	J J J 23 38 0 0 D	9	7
9	J J J 23 38 0 0 0 D	9	7

256×256, $k = 16\pi$, $o_{\max} = 10$

# levels	Smoothing schedule	MG	FGMRES
4	J J 34 D	12	10
5	J J 38 29 D	12	10
6	J J 35 20 6 D	12	10
7	J J 35 20 6 3 D	12	10
8	J J 35 20 6 3 0 D	12	10
9	J J 35 20 6 3 0 0 D	12	10

256×256, $k = 32\pi$, $o_{\max} = 18$

# levels	Smoothing schedule	MG	FGMRES
3	J 34 D	21	18
4	J 39 39 D	22	18
5	J 35 33 5 D	23	18
6	J 35 33 5 3 D	23	18
7	J 35 33 5 3 2 D	23	18
8	J 35 33 5 3 2 0 D	23	18
9	J 35 32 6 5 3 0 0 D	23	18

3.2.2. Hand-tuned smoothing schedules. Numerical experiments with a fixed number of GMRES smoothing steps at every level did not result in good performance. To get an idea of an appropriate smoothing schedule, we proceed as follows. For given k, we calculate the number $o_{\max}$ of FGMRES iterations needed with j-level multigrid preconditioning, where we use Jacobi smoothing on all grids for which $kh_i < 1/2$ and do a direct solve at the next coarser grid, making j grids in all. We then replace the direct solve on the coarsest grid of the j-level scheme with GMRES smoothing on this grid, coupled with a direct solve on the next coarser grid, and determine the smallest number m_j of GMRES smoothing steps required for the outer iteration to converge in $o_{\max}$ steps. For example, for the first line of Table 3.4, six outer FGMRES steps were needed for a 5-level scheme, and then $m_5 = 13$ GMRES smoothing steps were needed for the outer iteration of the new 6-level preconditioner to converge in six steps. When the number m_j has been determined, we could fix the number of

GMRES smoothing steps to m_j on this grid, add one coarser level, determine the optimal number of GMRES smoothing steps on the coarser grid, and continue in this fashion until the maximal number of levels is reached. This approach is modified slightly by, whenever possible, trying to reduce the number of smoothings on finer levels once coarser levels have been added. This is often possible, since replacing the exact solve on the coarsest grid with several GMRES smoothing steps often has a regularizing effect, avoiding some damage possibly done by an exact coarse grid correction in modes whose eigenvalues are not well represented on the coarse grid. This hand-tuning procedure gives insight into the best possible behavior of this algorithm.

In contrast to classical linear smoothers, whose damping properties for different modes is fixed, the damping properties of GMRES depend on the initial residual. In particular, since GMRES is constructed to minimize the residual, it will most damp those modes that lead to the largest residual norm reduction. For this reason, we will favor postsmoothing over presmoothing to prevent the unnecessary damping of smoother modes that should be handled by the coarse grid correction. We do include two GMRES presmoothing steps to avoid overly large oscillatory components in the residual prior to restricting it to the next lower level, which could otherwise lead to spurious oscillatory error components being introduced by the coarse grid correction.

The results are shown in Table 3.4. The entry "D" denotes a direct solve on the corresponding level, and "J" indicates that damped Jacobi smoothing was used on this level. Looking at the smoothing schedules, we observe a "hump" in the number of GMRES smoothing steps on the first two levels on which GMRES smoothing is used. Below this, the number decreases and is often zero for the coarsest levels. However, GMRES smoothing still helps on levels which are extremely coarse with regard to resolution of the waves: in the case $k = 32\pi$, for instance, performing three GMRES smoothing steps on level 4 (which corresponds to 1/2 point per wavelength) still improves convergence.

We remark that the number of outer iterations in all these tests, for both preconditioned FGMRES and stand-alone MG, is the same as for the corresponding two-grid versions of these methods, so we cannot expect faster convergence with respect to the wave number k. We also note that the number of iterations for multigrid is very close to that for FGMRES with multigrid preconditioning. We believe this is because the relatively large number of GMRES smoothing steps on intermediate levels eliminates lower frequency errors, and this mitigates the effects of axis crossings. We will return to this point in section 3.4.

3.3. A stopping criterion based on L^2-sections.

Hand tuning as in the previous section is clearly not suitable for a practical algorithm. In this section, we develop a heuristic for finite element discretizations that automatically determines a stopping criterion for the GMRES smoother. This technique is based on an idea introduced in [32].

We briefly introduce some standard terminology for multilevel methods applied to second order elliptic boundary value problems on a bounded domain $\Omega \subset \mathbb{R}^2$ (see [34]). We assume a nested hierarchy of finite element spaces

$$V_1 \subset V_2 \subset \cdots \subset V_J \subset V = H^1(\Omega)$$

in which the largest space V_J corresponds to the grid on which the solution is sought. We require the L^2-orthogonal projections $Q_\ell : V \to V_\ell$, defined by

$$(Q_\ell u, v) = (u, v) \qquad \forall v \in V_\ell, \qquad \ell = 1, \dots, J,$$

where $(\cdot,\cdot)$ denotes the L^2-inner product on Ω. Let $\Phi_\ell = \{\phi_1^{(\ell)}, \dots, \phi_{n_\ell}^{(\ell)}\}$ denote the basis of the finite element space V_ℓ of dimension n_ℓ used in defining the stiffness and mass matrices. By the nestedness property $V_\ell \subset V_{\ell+1}$, there exists an $n_{\ell+1} \times n_\ell$ matrix $\mathcal{I}_\ell^{\ell+1}$ whose columns contain the coefficients of the basis Φ_ℓ in terms of the basis $\Phi_{\ell+1}$, so that, writing the bases as row vectors,

$$[\phi_1^{(\ell)}, \dots, \phi_{n_\ell}^{(\ell)}] = [\phi_1^{(\ell+1)}, \dots, \phi_{n_{\ell+1}}^{(\ell+1)}]\mathcal{I}_\ell^{\ell+1}.$$

The stopping criterion we shall use for the GMRES smoothing iterations is based on the representation of the residual r_ℓ of an approximate solution $\tilde{u}_\ell$ of the level-ℓ equation as the sum of differences of L^2-projections,

$$r_\ell = (I - Q_{\ell-1})r_\ell + (Q_{\ell-1} - Q_{\ell-2})r_\ell + \cdots + (Q_2 - Q_1)r_\ell + Q_1 r_\ell,$$

which we refer to as *residual sections.* The following result for coercive problems, which was proven in [32], shows that the error $u_\ell - \tilde{u}_\ell$ is small if each appropriately weighted residual section is small.

THEOREM 3.1. *Assume the underlying elliptic boundary value problem is H^1-elliptic and $H^{1+\alpha}$-regular with $\alpha > 0$. Then there exists a constant c independent of the level ℓ such that the $H^1(\Omega)$-norm of the error on level ℓ is bounded by*

$$\|u_\ell - \tilde{u}_\ell\|_1 \le c \left(\|Q_1 r_\ell\| + \sum_{j=2}^{\ell} h_j^\alpha \|(Q_j - Q_{j-1})r_\ell\| \right). \tag{3.1}$$

The boundary value problem (1.1)–(1.3) under consideration is not H^1-elliptic and therefore does not satisfy the assumptions of this theorem. We have found, however, that the bound (3.1) suggests a useful stopping criterion: terminate the GMRES smoothing iteration on level ℓ as soon as the residual section $(Q_\ell - Q_{\ell-1})r_\ell$ has become sufficiently small. To obtain a formula for the computation of these sections, assume the residual r_ℓ is represented by the coefficient vector $\boldsymbol{r}_\ell$ in terms of the dual basis of Φ_ℓ. The representation of $Q_{\ell-1}r_\ell$ with respect to the dual basis of $\Phi_{\ell-1}$ is then given by the coefficient vector $\mathcal{I}_\ell^{\ell-1}\boldsymbol{r}_\ell \in \mathbb{C}^{n_{\ell-1}}$, where $\mathcal{I}_\ell^{\ell-1} := (\mathcal{I}_{\ell-1}^{\ell})^\top$. Returning to the representation with respect to the basis $\Phi_{\ell-1}$ requires multiplication with $\boldsymbol{M}_{\ell-1}^{-1}$, so that we obtain

$$\|Q_{\ell-1}r_\ell\|^2 = (Q_{\ell-1}r_\ell, Q_{\ell-1}r_\ell) = (\mathcal{I}_\ell^{\ell-1}\boldsymbol{r}_\ell)^\top \boldsymbol{M}_{\ell-1}^{-1}\mathcal{I}_\ell^{\ell-1}\boldsymbol{r}_\ell.$$

If the sequence of triangulations underlying the finite element spaces V_ℓ is quasi-uniform, then the mass matrix of level ℓ is uniformly equivalent to the identity scaled by h^d, where d denotes the dimension of the domain. For the case $d = 2$ under consideration, this means that the Euclidean inner product on the coordinate space $\mathbb{C}^{n_\ell}$, denoted by $(\cdot,\cdot)_E$, when scaled by h_ℓ^2, is uniformly equivalent (with respect to the mesh size) to the L^2-inner product on V_ℓ. Therefore, the associated norms satisfy

$$ch_\ell^2\|\boldsymbol{v}_\ell\|_E^2 \le \|v_\ell\|^2 = \boldsymbol{v}_\ell^\top \boldsymbol{M}_\ell \boldsymbol{v}_\ell \le Ch_\ell^2\|\boldsymbol{v}_\ell\|_E^2 \qquad \forall v_\ell \in V_\ell,$$

where $\boldsymbol{v}_\ell$ is the coordinate vector of v_ℓ with respect to Φ_ℓ. Using this norm equivalence it is easily shown that

$$\begin{aligned} c\|(\mathcal{I} - h_\ell^2/h_{\ell-1}^2\mathcal{I}_{\ell-1}^{\ell}\mathcal{I}_\ell^{\ell-1})\boldsymbol{r}_\ell\|_E &\le h_\ell\|(I - Q_{\ell-1})r_\ell\| \\ &\le C\|(\mathcal{I} - h_\ell^2/h_{\ell-1}^2\mathcal{I}_{\ell-1}^{\ell}\mathcal{I}_\ell^{\ell-1})\boldsymbol{r}_\ell\|_E \end{aligned}$$

for some constants c and C uniformly for all levels ℓ. As a result, the residual sections may be computed sufficiently accurately without the need for inverting mass matrices.

In [32], it was suggested that the GMRES smoothing iteration for a full multigrid cycle be terminated as soon as the residual section on the given level is on the order of the discretization error on that level. For the problem under consideration here, we shall use the relative reduction of L^2-sections as a stopping criterion, so that roughly an equal error reduction for all modes is achieved in one V-cycle. On the first level on which GMRES smoothing is used, we have the additional difficulty that many eigenvalues may be badly approximated on the next-coarser level. For this reason, it is better to also smooth the oscillatory modes belonging to the next lower level and base the stopping criterion on the residual section $(I - Q_{\ell-2})r_\ell$ instead; we will use this "safer" choice on all levels. Numerical experiments with optimal smoothing schedules have shown the relative reduction of this residual section to scale like kh_ℓ, so that we arrive at the stopping criterion

$$\left\| \boldsymbol{r} - \frac{h_\ell^2}{h_{\ell-2}^2} \mathcal{I}_{\ell-1}^{\ell} \mathcal{I}_{\ell-2}^{\ell-1} (\mathcal{I}_{\ell-1}^{\ell} \mathcal{I}_{\ell-2}^{\ell-1})^\top \boldsymbol{r} \right\|_E \leq \gamma k h_\ell. \tag{3.2}$$

A complete description of the multigrid V-cycle algorithm starting on the finest level ℓ is as follows.

Algorithm 3.1. $\tilde{\boldsymbol{u}}_\ell = MG(\boldsymbol{u}_\ell^{(0)}, \boldsymbol{f}_\ell)$. *Multigrid V-cycle with GMRES smoothing on coarse levels*

```
if $\ell = 1$
    $\tilde{\boldsymbol{u}}_\ell := \boldsymbol{A}_\ell^{-1} \boldsymbol{f}_\ell$
else
    if $kh_\ell < 1/2$
        perform $m_1$ steps of damped Jacobi smoothing to obtain $\boldsymbol{u}_\ell^{(1)}$
    else
        perform 2 steps of GMRES smoothing to obtain $\boldsymbol{u}_\ell^{(1)}$
    endif
    $\boldsymbol{u}_\ell^{(2)} := \boldsymbol{u}_\ell^{(1)} + \mathcal{I}_{\ell-1}^{\ell} MG(0, \mathcal{I}_\ell^{\ell-1}(\boldsymbol{f}_\ell - \boldsymbol{A}_\ell \boldsymbol{u}_\ell^1))$
    if $kh_\ell < 1/2$
        perform $m_2$ steps of damped Jacobi smoothing to obtain $\tilde{\boldsymbol{u}}_\ell$
    else
        perform GMRES smoothing until stopping criterion (3.2) is satisfied
        or $m = m_{\max}$ to obtain $\tilde{\boldsymbol{u}}_\ell$
    endif
endif
```

In the multigrid V-cycle, Algorithm 3.1 is used recursively beginning with the finest level and iterated until the desired reduction of the relative residual is achieved on the finest level. In the FGMRES variant, Algorithm 3.1 represents the action of the inverse of a preconditioning operator being applied to the vector $\boldsymbol{f}_\ell$.

3.4. Experiments with automated stopping criterion. We now show how the multigrid solver and preconditioner perform with the automated stopping criterion for GMRES smoothing. Each method is applied to the two-dimensional Helmholtz problem on the unit square with a second order absorbing boundary condition and random right-hand side data. In these tests, we used $\gamma = 0.1$ in (3.2), and we also imposed an upper bound $m_{\max}$ on the number of GMRES smoothing steps, terminating the smoothing if the stopping criterion is not satisfied after $m_{\max}$ steps; we tested two

Table 3.5

Iteration counts for multigrid and multigrid-preconditioned FGMRES for various fine-grid sizes and wave numbers. In all cases, GMRES smoothing is performed on levels for which $kh > 1/2$ and the smoothing is terminated by the L^2-section stopping criterion or when $m_{\max}$ smoothing steps are reached. A dash denotes divergence.

	Multigrid						MG-preconditioned FGMRES					
$N\backslash k$	2π	4π	8π	16π	32π	64π	2π	4π	8π	16π	32π	64π
	$m_{\max} = 40$											
64	12	12	13				7	8	9			
128	12	12	13	16			7	8	9	13		
256	12	12	13	17	27		7	8	9	13	20	
512	12	12	13	16	27	78	7	8	9	13	21	36
	$m_{\max} = 20$											
64	12	12	13				7	8	9			
128	12	12	13	22			7	8	9	16		
256	12	12	13	21	201		7	8	10	16	37	
512	12	12	13	21	331	—	7	8	10	16	36	80

values, $m_{\max} = 40$ and $m_{\max} = 20$. At fine-grid levels, where damped Jacobi smoothing is used, the number of presmoothings and postsmoothings was $m_1 = m_2 = 2$.

We present three sets of results. Table 3.5 shows iteration counts for a variety of wave numbers and mesh sizes. Table 3.6 examines performance in more detail by showing the automatically generated smoothing schedules for two wave numbers, $k = 8\pi$ and $k = 32\pi$. Finally, to give an idea of efficiency, Table 3.7 shows an estimate for the operation counts (multiplications) required for the problems treated in Table 3.6.

Table 3.6

Smoothing schedules with automated stopping criterion for selected parameters.

$k = 8\pi$, $m_{\max} = 40$

	Grid	# levels	Smoothing schedule							Iterations
FGMRES	64×64	6	J	20	17	11	2	D		9
	128×128	7	J	J	19	16	11	2	D	9
MG	64×64	6	J	16	17	11	2	D		13
	128×128	7	J	J	18	16	11	2	D	13

$k = 8\pi$, $m_{\max} = 20$

	Grid	# levels	Smoothing schedule							Iterations
FGMRES	64×64	6	J	19	17	12	2	D		9
	128×128	7	J	J	18	16	11	2	D	9
MG	64×64	6	J	16	17	11	2	D		13
	128×128	7	J	J	18	16	11	2	D	13

$k = 32\pi$, $m_{\max} = 40$

	Grid	# levels	Smoothing schedule									Iterations
FGMRES	256×256	8	J	34	38	22	1	0	0	D		20
	512×512	9	J	J	33	38	22	1	0	0	D	21
MG	256×256	8	J	32	36	18	1	0	0	D		27
	512×512	9	J	J	32	37	19	1	0	0	D	27

$k = 32\pi$, $m_{\max} = 20$

	Grid	# levels	Smoothing schedule									Iterations
FGMRES	256×256	8	J	20	20	18	1	0	0	D		37
	512×512	9	J	J	20	20	18	1	0	0	D	36
MG	256×256	8	J	20	20	16	1	0	0	D		201
	512×512	9	J	J	20	20	16	1	0	0	D	331

TABLE 3.7

Operation counts (in millions) for selected parameters with $m_{\max} = 40$.

	$k = 8\pi$		$k = 32\pi$	
Grid	MG	FGMRES	MG	FGMRES
64×64	13.2	13.3		
128×128	24.0	22.1		
256×256	61.2	43.2	1091.2	971.1
512×512	196.6	148.1	1418.1	1377.8

We make the following observations on these results:

- For low wave numbers, the number of iterations of stand-alone multigrid is close to that for FGMRES. The difference increases as the wave number increases, especially for the case $m_{\max} = 20$. For large enough k, multigrid fails to converge, whereas MG-preconditioned FGMRES is robust. This behavior is explained by the results of section 2.2.2. For large wave numbers, the increased number of amplified modes eventually causes multigrid to fail; a larger number of smoothing steps mitigates this difficulty, presumably by eliminating some smooth errors. The (outer) FGMRES iteration handles this situation in a robust manner.
- The automated stopping criterion leads to smoothing schedules close to those obtained by hand tuning (see Table 3.4), and correspondingly similar outer iteration counts.
- The operation counts shown in Table 3.7 suggest that MG-preconditioned FGMRES is more efficient than stand-alone multigrid even when the latter method is effective. Operation counts for MG-preconditioned FGMRES with $m_{\max} = 20$ (not shown) indicate that the costs for this strategy, which uses more outer iterations but fewer smoothing steps than when $m_{\max} = 40$, are essentially the same, although the larger number of outer FGMRES steps requires more memory to store more fine-grid vectors.
- For fixed wave number, outer iteration counts are mesh independent, so that standard "multigrid-like" behavior is observed. The costs per unknown (for fixed k and smoothing schedule) also display textbook multigrid efficiency, i.e., they are constant. However, because Jacobi smoothing is less expensive than GMRES smoothing, during the initial stages of mesh refinement the operation counts grow at less than a linear rate.
- The growth in outer iteration counts with increasing wave number is slower than linear in k. The operation counts increase more rapidly, however, because of the increased number of smoothing steps required for larger wave numbers.

We also note that GMRES has nontrivial storage requirements; we expect other Krylov subspace methods with lower storage requirements (e.g., QMR [15] or Bi-CGSTAB [33]) to perform in a similar manner.

4. Application to an exterior problem. As a final example we apply the algorithm to an exterior scattering problem for the Helmholtz equation as given in (1.1)–(1.3). The domain Ω consists of the exterior of an ellipse bounded externally by a circular artificial boundary Γ_∞ on which we impose the exact nonlocal Dirichlet-to-Neumann (DtN) boundary condition (see [21]). The source function is $f = 0$; forcing is due to the boundary condition on the boundary Γ of the scatterer, given by

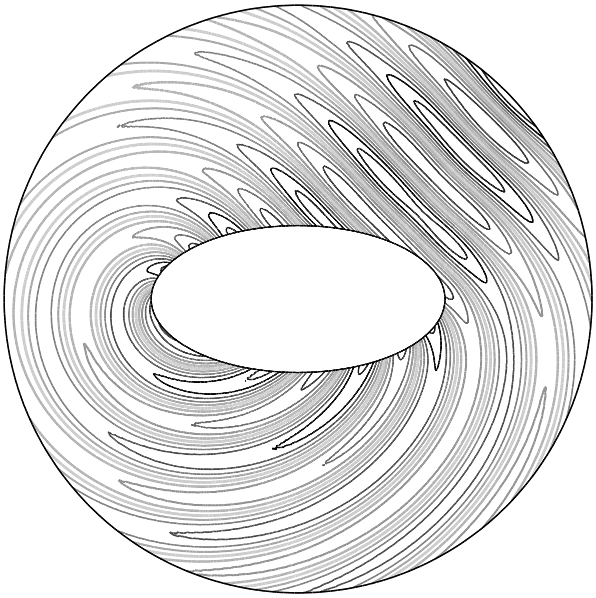


FIG. 4.1. *Contour plot of the solution of the Dirichlet problem with wave number $k = 8\pi$.*

$$u(x,y) = g(x,y) \quad \text{or} \quad \frac{\partial u(x,y)}{\partial n} = \frac{\partial g(x,y)}{\partial n}, \qquad (x,y) \in \Gamma,$$

with data $g(x,y) = -e^{ik(x\cos\alpha + y\sin\alpha)}$ representing a plane wave incident at angle α to the positive x-axis. The solution u represents the scattered field associated with the obstacle and incident field g; the resulting total field $u+g$ then satisfies a homogeneous Dirichlet or Neumann boundary condition on Γ, respectively. An angle of incidence $\alpha = \pi/4$ was chosen to avoid a symmetric solution. The problems were discretized using linear finite elements beginning with a very coarse mesh which is successively refined uniformly to obtain a hierarchy of nested finite element spaces. The finest mesh, obtained after five refinement steps, contains 32768 degrees of freedom. Several combinations of k and h were tested, where in each case $kh < 0.5$ on the finest mesh. Figure 4.1 shows a contour plot of the solution u of the Dirichlet problem for $k = 8\pi$. The computations make use of the PDE Toolbox of the Matlab 5.3 computing environment.

The problems were solved using both the stand-alone and FGMRES-accelerated versions of multigrid, with GMRES smoothing using the residual section stopping criterion with $\gamma = 0.1$, outer stopping criterion requiring residual reduction by a factor of 10^{-6} as in section 3, and zero initial guess. We used the maximal number of levels in all examples with the exception of the Dirichlet problem for $k = 8\pi$, where we also varied the number of levels from six down to two. The results are shown in Table 4.1. The table gives the wave number k and the length of the ellipse E in wavelengths $\lambda = 2\pi/k$. The third column gives the maximum value of kh on the finest mesh and the fourth column indicates the number of levels used in each computation. The last two columns list the iteration counts.

We observe that the preconditioned iteration performs well in all cases with a growth in number of iterations slower than linear in k. The stand-alone multigrid variant performs less well in comparison, requiring more than 100 steps to converge

TABLE 4.1

Iteration counts for the exterior scattering problem with Dirichlet or Neumann plane wave data on the boundary of an ellipse for various wave numbers, grid sizes, and numbers of levels. A dash denotes divergence.

k	Size $E[\lambda]$	$(kh_{\max})_{\text{fine}}$	# Levels	MG	FGMRES
		Dirichlet problem			
2π	1	.10	6	36	13
		.21	5	27	12
		.42	4	26	12
4π	2	.21	6	38	16
		.42	5	27	14
8π	4	.42	6	41	20
			5	—	28
			4	100	26
			3	41	16
			2	41	13
		Neumann problem			
2π	1	.10	6	—	21
		.21	5	37	15
		.42	4	21	12
4π	2	.21	6	—	28
		.42	5	53	21
8π	4	.42	6	—	32

in several cases and even diverging in one case. This is particularly the case for the Neumann problem, where the superiority of the preconditioned variant is even more pronounced. For the Neumann problems we also notice a slight growth in iteration counts for fixed k and decreasing h.

5. Conclusions. The results of this paper show that the addition of Krylov subspace iteration to multigrid, both as a smoother and as an outer accelerating procedure, enables the construction of a robust multigrid algorithm for the Helmholtz equation. GMRES is an effective smoother for grids of intermediate coarseness, in that it appears not to amplify any error modes and in addition tends to have a regularizing effect on the contribution to the coarse grid correction coming from smoothing on a given level. The combination of our multigrid algorithm as a preconditioner with FGMRES is effective in handling the deficiencies of standard multigrid methods for the Helmholtz equation, and the outer FGMRES acceleration is necessary, particularly for high wave numbers. In addition, results in the paper indicate that grids too coarse to result in a meaningful discretization of the Helmholtz equation may still provide some useful information for coarse grid corrections. Using an automated stopping criterion based on L^2-sections of the residual leads to smoothing cycles that are close to hand-tuned optimal smoothing schedules.

An important aspect of our algorithm is that it consists of familiar building blocks and is thus easily implemented. For very large wave numbers for which the discretization must not only keep kh but also k^3h^2 small, the grid hierarchy will contain more grids fine enough to use Jacobi smoothing, thus making the algorithm more efficient. The result is a multigrid method that appears to converge with a rate independent of the mesh size h and with only moderate dependence on the wave number k. Finally, the numerical results show that we are able to effectively solve Helmholtz problems with wave numbers of practical relevance.

REFERENCES

[1] A. Bamberger, P. Joly, and J. E. Roberts, *Second-order absorbing boundary conditions for the wave equation: A solution for the corner problem*, SIAM J. Numer. Anal., 27 (1990), pp. 323–352.

[2] R. E. Bank, *A comparison of two multilevel iterative methods for nonsymmetric and indefinite elliptic finite element equations*, SIAM J. Numer. Anal., 18 (1981), pp. 724–743.

[3] R. E. Bank and C. C. Douglas, *Sharp estimates for multigrid rates of convergence with general smoothing and acceleration*, SIAM J. Numer. Anal., 22 (1985), pp. 617–633.

[4] A. Bayliss, C. I. Goldstein, and E. Turkel, *An iterative method for the Helmholtz equation*, J. Comput. Phys., 49 (1983), pp. 443–457.

[5] A. Bayliss, C. I. Goldstein, and E. Turkel, *Preconditioned conjugate gradient methods for the Helmholtz equation*, in Elliptic Problem Solvers II, G. Birkhoff and A. Schoenstadt, eds., Academic Press, Orlando, FL, 1984, pp. 233–243.

[6] A. Bayliss, C. I. Goldstein, and E. Turkel, *The numerical solution of the Helmholtz equation for wave propagation problems in underwater acoustics*, Comput. Math. Appl., 11 (1985), pp. 655–665.

[7] A. Behie and P. Forsyth, *Comparison of fast iterative methods for symmetric systems*, IMA J. Numer. Anal., 3 (1983), pp. 41–63.

[8] F. A. Bornemann and P. Deuflhard, *The cascadic multigrid method for elliptic problems*, Numer. Math., 75 (1996), pp. 135–152.

[9] D. Braess, *On the combination of the multigrid method and conjugate gradients*, in Multigrid Methods II, W. Hackbusch and U. Trottenberg, eds., Lecture Notes in Math. 1228, Springer-Verlag, Berlin, 1986, pp. 52–64.

[10] J. H. Bramble, J. E. Pasciak, and J. Xu, *The analysis of multigrid algorithms for nonsymmetric and indefinite elliptic problems*, Math. Comp., 51 (1988), pp. 389–414.

[11] A. Brandt and I. Livshits, *Wave-ray multigrid method for standing wave equations*, Electron. Trans. Numer. Anal., 6 (1997), pp. 162–181.

[12] A. Brandt and S. Ta'asan, *Multigrid methods for nearly singular and slightly indefinite problems*, in Multigrid Methods II, W. Hackbusch and U. Trottenberg, eds., Lecture Notes in Math. 1228, Springer-Verlag, Berlin, 1986, pp. 99–121.

[13] W. L. Briggs, *A Multigrid Tutorial*, SIAM, Philadelphia, 1987.

[14] B. Enquist and A. Majda, *Absorbing boundary conditions for the numerical simulation of waves*, Math. Comp., 31 (1977), pp. 629–651.

[15] R. W. Freund and N. M. Nachtigal, *QMR: A quasi-minimal residual method for non-Hermitian linear systems*, Numer. Math., 60 (1991), pp. 315–339.

[16] D. Givoli, *Non-reflecting boundary conditions*, J. Comput. Phys., 94 (1991), pp. 1–29.

[17] C. I. Goldstein, *Multigrid preconditioners applied to the iterative solution of singularly perturbed elliptic boundary value problems and scattering problems*, in Innovative Numerical Methods in Engineering, Proceedings of the 4th International Symposium, Georgia Institute of Technology, Atlanta, 1986, pp. 97–102.

[18] W. Hackbush, *Multi-Grid Methods and Applications*, Springer-Verlag, Berlin, 1985.

[19] I. Harari and T. J. R. Hughes, *Finite element method for the Helmholtz equation in an exterior domain*, Comput. Methods Appl. Mech. Engrg., 87 (1991), pp. 59–96.

[20] F. Ihlenburg and I. Babuška, *Finite element solution to the Helmholtz equation with high wave numbers*, Comput. Math. Appl., 30 (1995), pp. 9–37.

[21] J. B. Keller and D. Givoli, *Exact non-reflecting boundary conditions*, J. Comput. Phys., 82 (1989), pp. 172–192.

[22] R. Kettler, *Analysis and comparison of relaxation schemes in robust multigrid and conjugate gradient methods*, in Multigrid Methods, W. Hackbusch and U. Trottenberg, eds., Lecture Notes in Math. 960, Springer-Verlag, Berlin, 1982, pp. 502–534.

[23] B. Lee, T. A. Manteuffel, S. F. McCormick, and J. Ruge, *First-order system least-squares (FOSLS) for the Helmholtz equation*, SIAM J. Sci. Comput., 21 (2000), pp. 1927–1949.

[24] S. F. McCormick, ed., *Multigrid Methods*, Frontiers Appl. Math. 3, SIAM, Philadelphia, 1987.

[25] C. W. Oosterlee and T. Washio, *An evaluation of parallel multigrid as a solver and a preconditioner for singularly perturbed problems*, SIAM J. Sci. Comput., 19 (1998), pp. 87–110.

[26] A. Ramage, *A multigrid preconditioner for stabilised discretisations of advection-diffusion problems*, J. Comput. Appl. Math., 110 (1999), pp. 187–203.

[27] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS, Boston, 1995.

[28] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[29] A. Schatz, *An observation concerning Ritz-Galerkin methods with indefinite bilinear forms*, Math. Comp., 28 (1974), pp. 959–962.

[30] V. V. Shaidurov, *Some estimates of the rate of convergence for the cascadic conjugate-gradient method*, J. Comput. Math. Appl., 31 (1996), pp. 161–171.

[31] Y. Shapira, *Multigrid techniques for highly indefinite equations*, in Proceedings of the 1995 Copper Mountain Conference on Multigrid Methods, Copper Mountain, CO, 1995; also available online from http://www.c3.lanl.gov/∼yairs/.

[32] G. Starke, *On the Performance of Krylov Subspace Iterations as Smoothers in Multigrid Methods*, manuscript, 1995.

[33] H. A. van der Vorst, *BI-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 631–644.

[34] J. Xu, *A new class of iterative methods for nonselfadjoint or indefinite problems*, SIAM J. Numer. Anal., 29 (1992), pp. 303–319.

[35] H. Yserentant, *On the multi-level splitting of finite element spaces for indefinite elliptic boundary value problems*, SIAM J. Numer. Anal., 23 (1986), pp. 581–595.

[36] H. Yserentant, *Preconditioning indefinite discretization matrices*, Numer. Math., 54 (1988), pp. 719–734.

SIAM J. Sci. Comput.
Vol. 23, No. 4, pp. 1316–1334

OPTIMIZING SCHRÖDINGER FUNCTIONALS USING SOBOLEV GRADIENTS: APPLICATIONS TO QUANTUM MECHANICS AND NONLINEAR OPTICS[*]

JUAN JOSÉ GARCÍA-RIPOLL[†] AND VÍCTOR M. PÉREZ-GARCÍA[†]

Abstract. In this paper we study the application of the Sobolev gradients technique to the problem of minimizing several Schrödinger functionals related to timely and difficult nonlinear problems in quantum mechanics and nonlinear optics. We show that these gradients act as preconditioners over traditional choices of descent directions in minimization methods and show a computationally inexpensive way to obtain them using a discrete Fourier basis and a fast Fourier transform. We show that the Sobolev preconditioning provides a great convergence improvement over traditional techniques for finding solutions with minimal energy as well as stationary states and suggest a generalization of the method using arbitrary linear operators.

Key words. Sobolev gradients, ground states, nonlinear Schrödinger equations

AMS subject classifications. 65K10, 35Q55, 78M50, 82D50

PII. S1064827500377721

1. Introduction.

The observation of nature reveals that many unforced continuous systems tend to accommodate into stationary configurations, in which the distributions of mass, charge, velocity, etc., do not change throughout time. In the language of mathematical modeling all configurations are represented by points of a certain space of functions, $\psi(\mathbf{x}) \in W$, while the tendency of the system to lie into any of these states is given by a functional, $E(\psi) : W \to \mathbb{R}$, *the energy*, whose minima are precisely those stationary "states." For this reason it is possible to see many physical problems written as variational principles of the type "find $\psi \in W$ such that $E(\psi) : W \to \mathbb{R}$ achieves a minimum on W." In most situations the functional to be minimized has a dependence on ψ of the form

$$E(\psi) = \int f\left(\nabla\psi(\mathbf{x}), \psi(\mathbf{x})\right) d^n x. \tag{1.1}$$

However, a complete analytical description of the minima of the functional $E(\psi)$ is usually not possible. In this paper we will introduce several techniques for performing this study numerically, focusing on the minimization of $E(\psi)$ subject to physical constraints.

From a practical point of view, this problem is similar to that of finding the minima of a real function defined over a finite-dimensional space, such as $\mathbb{R}^n$. First, a definition of derivative of the functional, $\nabla E(\psi)$, must be chosen. If the domain of the functional W is equipped with some scalar product, we may use the Fréchet derivative which is given by a first order expansion of the functional around a function ψ,

$$E(\psi + \delta) = E(\psi) + \langle \delta, \nabla E(\psi)\rangle + \langle \nabla E(\psi), \delta\rangle + O\left(\|\delta\|^2\right). \tag{1.2}$$

[*] Received by the editors September 7, 2000; accepted for publication (in revised form) March 2, 2001; published electronically December 18, 2001.
http://www.siam.org/journals/sisc/23-4/37772.html

[†] Universidad de Castilla-La Mancha, Departamento de Matemáticas, E.T.S.I. Industriales, Avd. Camilo José Cela, 3, Ciudad Real, E-13071, Spain (jjgarcia@ind-cr.uclm.es, vperez@ind-cr.uclm.es).

The *critical points,* ψ_c, are defined as the points where the first order variation of the functional vanishes for any perturbation δ. That is, the derivative vanishes in a weak sense:

$$\langle \delta, \nabla E(\psi_c)\rangle = 0 \quad \forall \delta. \tag{1.3}$$

Just like in the finite-dimensional case it is possible to show that any minimum of the functional must also be a critical point. Thus a common approach is to solve (1.3) and verify a posteriori which solutions are actually minima of the functional. If $W = L^2(\mathbb{R}^n)$, this procedure gives us the well-known Euler–Lagrange equations of the problem, which is a partial derivatives equation (PDE):

$$\frac{\partial f}{\partial \bar{\psi}} - \nabla \cdot \frac{\partial f}{\partial \nabla \bar{\psi}} = 0. \tag{1.4}$$

However, we have no guarantee of reducing the complexity of the problem, as it is by no means trivial to solve (1.4). We are also likely to obtain more solutions than we actually need, since not only minima, but also maxima and saddle node points, will satisfy the Lagrange equations.

To avoid these problems some other methods are used which aim at finding the minima of the functional directly, constructing minimizing sequences, $\{\psi_i\}$, whose limit is a minimum of the functional: $\psi_c = \lim_{i\to\infty} \psi_i$. These methods will be discussed in the following sections.

The outline of this paper is as follows. In section 2 we recall the definition of Sobolev gradients as given in [2, 3]. We derive a formal solution to the problem of finding these gradients which is based on the inversion of a positive Hermitian operator. In section 3 we derive an explicit expression for the Sobolev gradients in the trigonometric Fourier basis and comment on its implementation using fast Fourier transforms.

In sections 4 and 5 we apply the previous tools to two physical problems. Using descent techniques with Sobolev gradients over Fourier spaces we will find the ground states of a Bose–Einstein condensate in a rotating magnetic trap and the excited states for coupled laser beams propagating through a nonlinear medium. Both physical systems are modeled by nonlinear equations of Schrödinger type and present difficulties when traditional minimization techniques are used. We comment on the great improvements that are achieved using Sobolev gradients. Finally in section 6 we summarize our results and offer some conclusions.

2. Sobolev gradients.

2.1. Direct solutions of the variational problem.

There are two traditional approaches to the problem of finding the minima of a functional using a discrete basis. The first one expands the unknown solution using a Fourier basis $\{\phi_k\}$, $\psi = \sum_k c_k \phi_k$, and defines a new functional over the finite-dimensional space

$$E(\{c_k\}) \equiv E\left(\sum c_k \phi_k\right). \tag{2.1}$$

The functional is then minimized using methods which are well known from the domain of finite-dimensional problems, e.g., Newton's method or nonlinear conjugate gradient.

This procedure is quite straightforward and there is a huge amount of literature and tools which can be immediately applied to (2.1). However, for some types of problems one has to deal with highly nonlinear algebraic equations with many terms on each equation, something which is computationally too expensive to work with.

The second approach involves what is known as *descent techniques.* The idea is to manipulate the original functional (1.1) building an analytic equation for a minimizing trajectory in the target space W. This equation is then discretized and solved on a suitable basis. In the *continuous steepest descent* version, the trajectory $\psi(t) : \mathbb{R} \to W$ is continuous and defined by a PDE which involves the gradient of the functional (1.2),

$$\frac{\partial \psi}{\partial t} = -\nabla E. \tag{2.2}$$

The *discrete steepest descent* technique is computationally cheaper since instead of requiring an integrator for the PDE it constructs a sequence of estimates to the minimum, $\{\psi_{k+1} = \psi_k + \lambda_k \nabla E(\psi_k)\}$, by locally minimizing $E(\psi_k + \lambda \nabla E)$ with respect to the real parameter λ.

In this paper we deal only with descent techniques. The first and most important reason is that we will work with the definition of $\nabla E(\psi_k)$ trying to improve its convergence. As was already shown in [3], this work pays off: a good choice of the gradient improves convergence by several orders of magnitude. The second motivation is that by focusing on the gradient, we find that our algorithms will be essentially independent on the descent method, which leaves space for further improvement. For instance, one might apply these techniques to a nonlinear conjugate gradient method—which dynamically adjusts the search direction, $d_k \equiv \nabla E(\psi_k)$, with an estimate that takes into account the history of the evolution. Finally, by working with (2.2) or its discrete version we avoid the complex nonlinearities that arise in other methods, and we will have a more ample choice of Fourier basis to work with.

2.2. Boundary conditions. Before proceeding with the study of Sobolev gradients, we must make more precise the boundary conditions of our functional spaces. In this paper we will compare applications to several problems of physics in which the solutions concentrate on a narrow region of space and decrease exponentially as we move out of this region, that is,

$$|\psi(\mathbf{r})| = \mathcal{O}(e^{-|r|^2}), \quad |r| \to \infty.$$

Naturally, computers allow us to study only a small region of space, which means we have to introduce some reasonable boundary conditions and hope that the solutions of this simplified problem do not deviate much from the original one.

In our case we will perform our study with functions which are defined over a d-dimensional rectangular volume, $\Omega \equiv \{\mathbf{x} \in \Pi_i [a_i, b_i]\}$ (see section 2.13), and we will impose zero boundary conditions on the sides of this box.

2.3. Ordinary gradients. It is customary in the literature to work in spaces which are equipped with an L^2 scalar product and its corresponding norm

$$\langle \psi, \phi \rangle_{L^2} \equiv \int_\Omega \bar{\psi}\phi, \tag{2.3}$$

$$\|\psi\|^2_{L^2} \equiv \int_\Omega |\psi|^2. \tag{2.4}$$

If one does so and works with (1.2) as well as the boundary conditions, then the formal definition of the gradient is one of Lagrange's:

$$\nabla E(\psi) = \frac{\partial E}{\partial \bar{\psi}} - \nabla \frac{\partial E}{\partial \left(\nabla \bar{\psi}\right)}. \tag{2.5}$$

We will refer to this definition as the "ordinary" gradient to distinguish it from the different definitions that we will derive below.

2.4. Sobolev gradients. Following the ideas from [3] we will move our problem to a different space, which is the Sobolev space of functions, such that ψ and its derivatives, $\nabla\psi$, have a well-defined L^2-norm:

$$\mathbb{H}^1 \equiv \{\psi/\psi, \nabla\psi \in L^2\}. \tag{2.6}$$

This Sobolev space will also be equipped with a scalar product and a norm:

$$\langle \psi, \phi \rangle \equiv \int_\Omega \left[\bar{\psi}(\mathbf{x})\phi(\mathbf{x}) + \nabla\bar{\psi}(\mathbf{x}) \cdot \nabla\phi(\mathbf{x})\right] d^n x, \tag{2.7}$$

$$\|\psi\|^2 \equiv \int_\Omega \left[|\psi|^2 + |\nabla\psi|^2\right] d^n x. \tag{2.8}$$

To obtain a new explicit expression for the gradient of the functional in the Sobolev space we will follow a less rigorous derivation than in [2]. Performing a first order expansion of our functional around a trial state ψ we obtain

$$\begin{aligned} E(\psi + \varepsilon\delta) = E(\psi) &+ \varepsilon \int_\Omega \left[\bar{\delta}\frac{\partial E}{\partial \bar{\psi}} + \nabla\bar{\delta}\frac{\partial E}{\partial(\nabla\bar{\psi})}\right] \\ &+ \varepsilon \int_\Omega \left[\frac{\partial E}{\partial \psi}\delta + \frac{\partial E}{\partial(\nabla\psi)}\nabla\delta\right] + O(\varepsilon^2). \end{aligned} \tag{2.9}$$

Here the bars over δ and ψ denote complex conjugation.

We have to turn this expression into something like (1.2). This means that we have to find some function, $\phi \equiv \nabla_S E$, which is the gradient and satisfies

$$\int_\Omega \left[\bar{\delta}\frac{\partial E}{\partial \bar{\psi}} + \nabla\bar{\delta}\frac{\partial E}{\partial(\nabla\bar{\psi})}\right] = \int_\Omega \left[\bar{\delta}\phi + \nabla\bar{\delta}\nabla\phi\right] = \langle \delta, \phi \rangle, \tag{2.10}$$

$$\int_\Omega \left[\frac{\partial E}{\partial \psi}\delta + \frac{\partial E}{\partial(\nabla\psi)}\nabla\delta\right] = \int_\Omega \left[\bar{\phi}\delta + \nabla\bar{\phi}\nabla\delta\right] = \langle \phi, \delta \rangle. \tag{2.11}$$

If we use the boundary conditions to integrate by parts and then impose that any of these equalities be satisfied for all perturbations δ, the problem has a formal solution which is given by a Lagrange equation,

$$(1 - \triangle)\,\phi = \frac{\partial E}{\partial \bar{\psi}} - \nabla\frac{\partial E}{\partial(\nabla\bar{\psi})}. \tag{2.12}$$

In consequence, our formal expression for the Sobolev gradient of $E(\psi)$ finally reads

$$\nabla_S E \equiv (1 - \triangle)^{-1}\,\nabla E. \tag{2.13}$$

Here $\nabla_S E$ stands for the Sobolev gradient, ∇E is the ordinary one, and $(1 - \triangle)^{-1}$ represents the inverse of a linear and strictly positive definite operator.

3. Sobolev gradients on discrete Fourier spaces. As mentioned above, we will work with functions which are defined over a rectangular Ω with side lengths given by $L_i = b_i - a_i$. We customarily define an orthogonal set of basis functions, $\phi_n = e^{ik_n x}$, over Ω, where $\{k_n = 2\pi(\frac{n_1}{L_1}, \dots, \frac{n_d}{L_d}),\ n_i \in Z\}$.

It is well known that it is possible to expand any continuous function $f(x)$ with periodic boundary conditions using this basis:

$$f(x) = \sum_{n=-\infty}^{+\infty} \hat{f}_n \phi_n(x), \tag{3.1}$$

where

$$\hat{f}_n = \frac{1}{V} \int_\Omega \bar{\phi}_n(x) f(x). \tag{3.2}$$

In the previous formula $V = \Pi_i L_i$ is the volume of Ω and arises because of the lack of normalization of the basis functions, a common practice which saves some computation time.

To discretize the problem we will work within the set of functions sampled over a set of evenly spaced points from Ω, $\{x_n = (n_1 h_1, \dots, n_d h_d),\ n_i = 0, \dots, N_i - 1\}$. Here h_i represents the spacing along the ith dimension, n is a vector of nonnegative integers, and we denote a sampled function by an index, as in $f_n \equiv f(x_n)$.

Due to this procedure our previous Fourier basis is now redundant. We can choose finite subset of functions that represent any sampled function. These functions are given by $\{k_n = 2\pi(\frac{n_1}{L_1}, \dots, \frac{n_d}{L_d}),\ n_i = -M_i + 1, \dots, M_i\}$, where $M_i = [N_i/2]$ is the integer quotient of N_i divided by 2. In this basis a sampled function is given by an expansion which reads

$$f_m = \sum_n \hat{f}_n \phi_n(x_m) \tag{3.3}$$

and which makes use of the same coefficients as (3.2).

The advantage of the finite Fourier basis over other approaches is that it provides an approximant of any function whose error is of order $O(L_i/N_i)^p$, where p is the maximum differentiability of the sampled function. Furthermore, there is a numerically efficient method known as the fast Fourier transform (FFT), which allows one to compute the Fourier coefficients up from the sampled function, $f_n \to \hat{f}_m$, and vice versa [1].

Solving (2.13) numerically in a discrete Fourier basis is simple. Let us say that we have computed the ordinary gradient and that its sampled version has some Fourier coefficients

$$\nabla E(\mathbf{x}_n) = \sum \hat{e}_m \phi_m(x_n), \tag{3.4}$$

and let us assume that there exists a certain solution to (2.13) and that it has another discrete Fourier expansion

$$\nabla_s E(\mathbf{x}_n) = \sum \hat{s}_m \phi_m(x_n). \tag{3.5}$$

Then by virtue of (2.13)

$$\hat{s}_m = \frac{\hat{e}_m}{1 + k_m^2}; \tag{3.6}$$

that is, in the sampled space the Sobolev gradient represents a preconditioning of the ordinary gradient such that the most oscillating modes are more attenuated. Furthermore, due to this very simple expression, computing the Sobolev preconditioning is computationally cheap and involves only minor changes to existing computer codes based on Fourier transforms.

4. Applications to quantum mechanics.

4.1. The problem. In this section we apply the Sobolev gradients technique to a timely problem from quantum physics. The system that we will study is a dilute gas of bosonic atoms which are cooled down to ultralow temperatures at which their dynamics become synchronized. When the temperature is low enough, the gas or "condensate" may be described using a single complex wave function, $\psi(\mathbf{x}, t)$, which is ruled by the so-called Gross–Pitaevskii equation, a type of nonlinear Schrödinger equation that for a Bose gas in a rotating trap reads [4]

$$i\partial_t \psi(\mathbf{x}, t) = \left[-\frac{1}{2}\triangle + V(\mathbf{x}) + g|\psi(\mathbf{x}, t)|^2 - \Omega L_z\right]\psi(\mathbf{x}, t). \tag{4.1}$$

Here $g \in \mathbb{R}^+, \Omega \in \mathbb{R}$, $L_z = i\,(x_1\partial_2 - x_2\partial_1)$ is a Hermitian operator whose expected value $\langle L_z\rangle = \int \bar{\psi} L_z \psi$ represents the *angular momentum* of the condensate along an axis of the trap, and the equation has been properly adimensionalized.

There is a conserved quantity associated to (4.1) which is called the *energy functional* of the condensate

$$E(\psi) = \frac{1}{2}\int\left\{|\nabla\psi|^2 + \bar{\psi}\left[V(\mathbf{x}) + \frac{1}{2}g|\psi|^2 - \Omega L_z\right]\psi\right\} d^n x. \tag{4.2}$$

Our objective in this part of the work will be to *find the solutions which are the minima of the energy subject to a restriction of the L^2-norm*

$$\int |\psi|^2 \equiv N. \tag{4.3}$$

The particular value of N is imposed by the experimental conditions and remains constant throughout evolution. Furthermore, without this restriction the absolute minimum of the energy is always reached at the trivial solution $\psi = 0$.

In quantum mechanics the variational formulation of the problem is traditionally converted into a Lagrange equation (1.4), which is nothing but the Gross–Pitaevskii equation for the so-called *stationary states.* In short the word "stationary" refers to solutions of the type

$$\psi(\mathbf{x}, t) = e^{-i\mu t}\phi_\mu(\mathbf{x}). \tag{4.4}$$

The pair $\{\mu, \phi_\mu(\mathbf{x})\}$ satisfies a nonlinear eigenvalue problem

$$\mu\phi_\mu(\mathbf{x}) = \left[-\frac{1}{2}\triangle + V(\mathbf{x}) + g|\phi_\mu(\mathbf{x})|^2 - \Omega L_z\right]\phi_\mu(\mathbf{x}). \tag{4.5}$$

Due to the difficulty of solving problem (4.5) directly we will try to find a direct solution to the variational problem.

4.2. Numerical methods: Imaginary time evolution. We can search the minima of (4.2) using descent techniques modified to account for the restriction on the norm (4.3). The first way to do this is to use a version of the continuous steepest descent which is known as *imaginary time evolution.* We can summarize this method with the following set of equations:

$$\nu(\mathbf{x},\tau) = \sqrt{\frac{N}{\|\sigma\|^2_{L^2}}}\sigma(\mathbf{x},\tau), \tag{4.6}$$

$$\frac{\partial\sigma}{\partial\tau}(\mathbf{x},\tau) = -\nabla E(\nu), \tag{4.7}$$

$$\nabla E(\nu) = \left[-\frac{1}{2}\triangle + V(\mathbf{x}) + g|\nu|^2 - \Omega L_z\right]\nu. \tag{4.8}$$

Here we see that $\nu(\mathbf{x},\tau)$ evolves continuously maintaining a fixed L^2-norm N and following the direction of decreasing energy given by ∇E. Indeed it is easy to show that $\frac{\partial}{\partial\tau}\left[E(\nu(\mathbf{x},\tau))\right] \leq 0$. Hence, the limit given by

$$\phi(\mathbf{x}) = \lim_{\tau\to\infty}\nu(\mathbf{x},\tau) \tag{4.9}$$

is at least a critical point of the energy, if not a minimum.[1]

From a practical point of view, the simplest way to find the minimizer using imaginary time evolution is to repeatedly integrate (4.7) for a very short time, Δt, apply (4.6) for the newly found $\sigma(\mathbf{x}, t+\Delta t)$, and use the new estimate for ν to redefine $\sigma \equiv \nu$ and repeat the procedure until convergence. This way one avoids the problem that according to (4.7) the norm of σ may grow indefinitely. The same consideration applies to the remaining methods that we will present here.

Although the method from (4.7) was derived using an ordinary gradient, nothing prevents us from applying our Sobolev preconditioning and all results should still be valid. If we do so, our new equations are

$$\nu(\mathbf{x},\tau) = \sqrt{\frac{N}{\|\sigma\|^2_{L^2}}}\sigma(\mathbf{x},\tau), \tag{4.10}$$

$$\frac{\partial\sigma}{\partial\tau}(\mathbf{x},\tau) = -\nabla_S E(\nu), \tag{4.11}$$

$$\nabla_S E(\nu) = (1-\triangle)^{-1}\left[-\frac{1}{2}\triangle + V(\mathbf{x}) + g|\nu|^2 - \Omega L_z\right]\nu \tag{4.12}$$

and the critical point is still given by (4.9).

4.3. Numerical methods: Minimization of the free energy. While the imaginary time evolution is easy to understand and to implement, the fact that it performs the descent over a path of functions with a certain norm makes it too restricted and sometimes too slow. A different approach is to define a new functional called *free energy* with a Lagrange multiplier that takes care of the restriction on the norm. In quantum mechanics this free energy is usually defined as

$$F_{QM}(\psi) = E(\psi) - \mu N(\psi), \tag{4.13}$$

[1] As is common with these local minimization procedures, it remains the problem that iterations may be trapped in a critical point which is not a minimum. Linear stability analysis may then be applied to check the validity of the solution.

because it preserves the linear form of the equations. Since our equations are already nonlinear we define a free energy functional more conveniently as

$$F(\psi) = E(\psi) + \frac{1}{2}\left(N(\psi) - \lambda\right)^2. \tag{4.14}$$

First and most important, it is not difficult to show that any absolute or relative minimum of $F(\psi)$ is also a minimum of $E(\psi)$ subject to (4.3) with a nonlinear eigenvalue given by $\mu = N(\psi) - \lambda$.

Second, as we will prove in Appendix A, $F(\psi)$ must have at least one finite norm minimum, something which cannot be easily assured for F_{QM}.

The practical advantage of our new functional consists in that fixing λ and Ω we can perform a continuous descent over the whole domain of $F(\psi)$ without renormalizing the solution on each iteration—i.e., the search space is larger. The new equation that we must integrate is thus

$$\frac{\partial \nu}{\partial \tau}(\mathbf{x}, \tau) = -\nabla F(\nu) = -\left[-\frac{1}{2}\triangle + V(\mathbf{x}) + g|\nu|^2 - \Omega L_z\right]\nu. \tag{4.15}$$

Using Sobolev preconditioning, we derive the following from (4.15):

$$\frac{\partial \nu}{\partial \tau}(\mathbf{x}, \tau) = -\nabla_S F(\nu) = -\left(1 - \triangle\right)^{-1}\left[-\frac{1}{2}\triangle + V(\mathbf{x}) + g|\nu|^2 - \Omega L_z\right]\nu. \tag{4.16}$$

In both cases the actual solution is still given by the limit of (4.9).

4.4. Numerical results. Up to this point we have shown four different numerical methods, two of them incorporating Sobolev preconditioning ((4.11) and (4.16)) and two without it ((4.7) and (4.15)). We have compared the efficiency of these methods for several test situations. The details of our study are as follows.

All methods have been implemented using the discrete Fourier basis mentioned above. This applies both to the calculation of derivatives and to the application of the Sobolev preconditioning. We restricted our simulations to two-dimensional problems with a radially symmetric potential, $V(\mathbf{x}) = \frac{1}{2}|\mathbf{x}|^2$, over a grid of 128×128 points. Practical experience with more complex problems shows that our results generalize to higher dimensionality and denser grids.

Due to the requirements of the method, both variants of imaginary time evolution ((4.7) and (4.11)) are integrated using a Runge–Kutta–Fehlberg method, where the tolerance is adapted as the solution converges to its target. On the other hand, instead of performing a continuous descent for both variants of the free energy descent ((4.15) and (4.16)), we found it more convenient and faster to perform a discrete steepest descent.

All programs have been implemented using the tensor-algebra environment called Yorick [6], an interpreted environment which is capable of fast numerical computations and which can be equipped with the FFTW library [7]. Execution times are given for a Digital Personal workstation 500au, but the programs run equally well on modest personal computers with Pentium-II processors and less than 64 Mb of memory.

As a test case we have considered three situations. Case A is the simplest one, corresponding to a stationary trap $\Omega = 0$, an intense nonlinearity $g = 100$, and starting the minimization with a radially symmetric Gaussian of unit width, that is, $\psi_0 \propto e^{-|\mathbf{x}|^2}$, a shape which is similar to that of the true ground state.

Cases B and C involve a rotating condensate with $\Omega = 0.6$ and $g = 100$. Under these conditions the functional achieves both an absolute minimum, which is the same

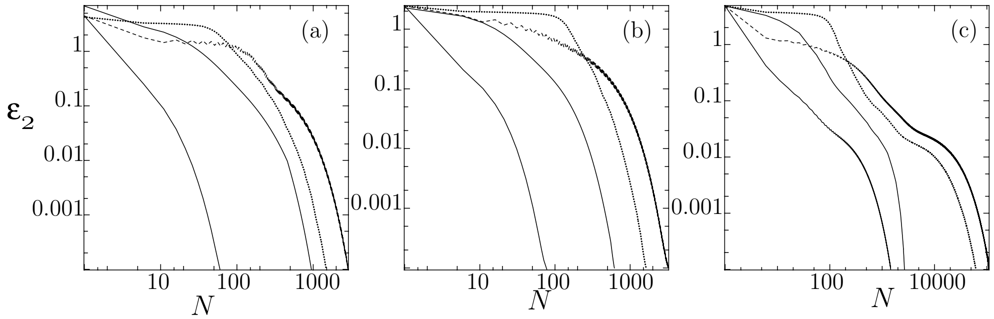


FIG. 4.1. *Evolution of error, $\varepsilon_2 \equiv \|\psi - \psi_{exact}\|_2$, through different minimization processes for continuous steepest descent with Sobolev preconditioning (lower solid line) and without it (dashed line) and for imaginary time evolution with Sobolev preconditioning (upper solid line) and without it (dotted line). Plots* (a) *to* (c) *correspond, respectively, to the cases* A, B, *and* C *described in the text. Both axes, error and number of iterations, are in a logarithmic scale.*

TABLE 4.1

Iterations and computation time for each minimization method: Imaginary time without (IT) and with (ITS) Sobolev preconditioning and free energy without (FE) and with (FES) Sobolev preconditioning. Shown are results for the initial data described in the text (cases A, B, *and* C*).*

Methods		IT	ITS	FE	FES
Case A	Iterations	1320	945	2850	55
	Time (s)	416	371	285	13
Case B	Iterations	1630	615	3210	320
	Time (s)	468	242	75	9
Case C	Iterations	64195	2665	108505	1455
	Time (s)	19863	1165	10861	168

as that found in case A, and a local minimum. The local minimum is a solution of vortex type, a topological defect, whose behavior near zero is $\psi \propto (x_1 + ix_2)/|\mathbf{x}|^2$.

For this reason we designed two test cases with different initial conditions. Case B starts with a Gaussian profile with a centered vortex or $\psi_0 \propto |\mathbf{x}|e^{-|\mathbf{x}|^2}(x_1 + ix_2)/|\mathbf{x}|^2$. In this case all methods are trapped on the local minimum with the centered vortex.

Finally, a third set of simulations, case C, uses the same parameters $\{\Omega = 0.6, g = 100\}$ but starts from a nonsymmetric initial state, $\psi_0 \propto |\mathbf{x}|e^{-|\mathbf{x}|^2}((x_1 - y_1) + i(x_2 - y_1))/|\mathbf{x} - \mathbf{y}|^2$ which is close to the vortex solution but belongs to the basin of attraction of the ground state. Here all minimization methods require more computational work since they must find a path out of the local minimum, and it is precisely in this case where the differences between methods are best shown.

In Figure 4.1 and Table 4.1 we summarize the results of the simulations. In case A it is apparent that the Sobolev preconditioning has a positive influence over convergence, with an astonishing result of 55 steps for the steepest descent with free energy. A similar behavior is found in case B.

In case C, the Sobolev preconditioning enhances convergence speed by two orders of magnitude. An intuitive explanation of why the steepest descent with a Sobolev gradient takes less steps to converge will be discussed in detail in Appendix B.

5. Applications to nonlinear optics.

5.1. The model. In this section we consider a model for a pair of incoherently interacting light beams. To be precise we will study the light field of each beam, $u(\mathbf{x}, t)$

and $w(\mathbf{x},t)$, propagating through a weakly nonlinear saturable optical medium. This system may be modeled by the Cauchy problem

$$i\partial_t u = -\triangle u + \frac{u}{1+\kappa(|u|^2+|w|^2)}, \tag{5.1}$$

$$i\partial_t w = -\triangle w + \frac{w}{1+\kappa(|u|^2+|w|^2)} \tag{5.2}$$

for the complex functions $u, w : \mathbb{R}^2 \times \mathbb{R}^+ \to \mathbb{C}$, which vanish at infinity and satisfy the initial data $u(\mathbf{x},0) = u_0(\mathbf{x})$ and $w(\mathbf{x},0) = w_0(\mathbf{x})$. Here $\kappa \in \mathbb{R}^+$, $-\triangle = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$ is the Laplacian operator which accounts for the diffraction of light, and the nonlinear term $(1+|u|^2+|w|^2)^{-1}$ models the saturable interaction among the beams.

Let us define the two-component vector

$$\tilde{U}(\mathbf{x},t) = \begin{pmatrix} u(\mathbf{x},t) \\ w(\mathbf{x},t) \end{pmatrix}; \tag{5.3}$$

then the energy functional for the system reads

$$E(\tilde{U}) = \int \left[-\tilde{U}^\dagger \triangle \tilde{U} + G(|\tilde{U}|^2) \right] d^n x, \tag{5.4}$$

where $G(\rho) = \frac{1}{\kappa^2}\left(\ln(1+\kappa\rho) - \kappa\rho\right)$. The analysis of this section may be generalized to more general nonlinearities with only minor changes to $G(\rho)$.

5.2. Stationary solutions. We are interested in stationary solutions, which are of the form

$$\tilde{U}(\mathbf{x},t) = \begin{pmatrix} e^{i\mu_u t} & 0 \\ 0 & e^{i\mu_w t} \end{pmatrix} U(\mathbf{x}) = e^{iMt} U(\mathbf{x}). \tag{5.5}$$

The equations for the stationary solutions now are of elliptic type,

$$MU = -\triangle U + G'(|U|^2)U, \tag{5.6}$$

with zero Dirichlet boundary conditions at infinity. Again this formulation poses a nonlinear eigenvalue problem for the pair $\{M, U\}$.

The stationary solutions are critical points of the energy functional subject to a constraint on the L^2-norm of each component. That is, defining

$$N_u = \int |u|^2 d^n x, \tag{5.7}$$

$$N_w = \int |w|^2 d^n x, \tag{5.8}$$

the first order variation of the energy around U_0 for fixed N_u and N_w must be zero:

$$\left.\frac{\delta E}{\delta U}\right|_{N_u,N_w} = 0. \tag{5.9}$$

The particular fixed values of $\{N_u, N_w\}$ represent the total intensity of each beam of light.

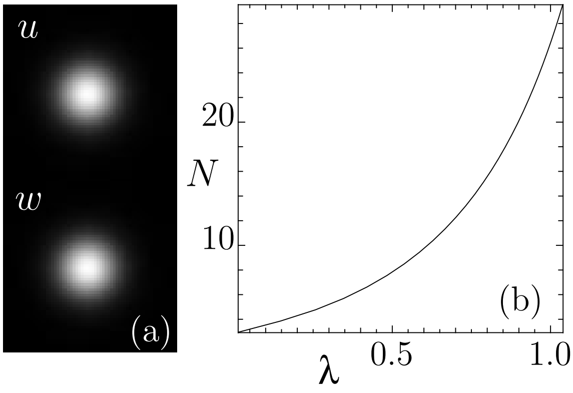


FIG. 5.1. (a) *Density profile of a ground state for* $N_u = N_w = 30$. (b) *Norms* $N_u = N_w = N$ *as a function of the nonlinear Lagrange multipliers* $\lambda_u = \lambda_w = \lambda$.

5.3. Ground states. In principle there are many different stationary solutions of (5.6). However, if we focus on the ground states (stationary solutions of minimal energy) we can apply some of the methods that we mentioned in section 4 with minor modifications to account for the higher dimensionality of the problem. For instance, one may define a free energy functional

$$F(U) = E(U) + \frac{1}{2}(N_u - \lambda_u)^2 + \frac{1}{2}(N_w - \lambda_w)^2 \tag{5.10}$$

and minimize it using a discrete steepest descent with Sobolev preconditioning.

By performing this minimization using different parameters $\{\lambda_u, \lambda_w\}$ one obtains nodeless localized solutions for u and w as shown in Figure 5.1. The precise values of λ_u and λ_w determine the norm of each component.

Let us remark that, up to a global factor, the ground state has the same shape in both envelopes. That is,

$$\begin{aligned} u_0(\mathbf{x}) &= N\sqrt{\xi}\,\rho(|\mathbf{x}|, N), \\ w_0(\mathbf{x}) &= N\sqrt{1-\xi}\,\rho(|\mathbf{x}|, N) \quad \forall \xi \in [0,1]. \end{aligned} \tag{5.11}$$

The common shape ρ depends on the total intensity $N = N_u + N_w$, and it is the one that fixes the values of μ_u and μ_w. For this reason if we had chosen the traditional definition of the free energy $F_{QM}(U) = E(U) + \mu_u N_u + \mu_w N_w$ we would have found an infinite degeneracy which prevents convergence to the desired values of $\{N_u, N_w\}$. This problem is removed by the use of our nonlinear Lagrange multipliers and the $\{\lambda_u, \lambda_w\}$ parameters.

5.4. Excited states. In the field of guided light waves there is a great interest on the properties of solutions of (5.6) which are not ground states, the so-called *excited states*. Minimization methods based on the energy functional (see section 4) cannot be applied to this task since excited states are not necessarily local minima of the energy. Instead, a variational principle somehow equivalent to (5.6) must be defined.

Let us rewrite (5.6) as the application of a nonlinear operator

$$f(U) \equiv \left[-\triangle - M + I(|U|^2)\right] U = 0, \tag{5.12}$$

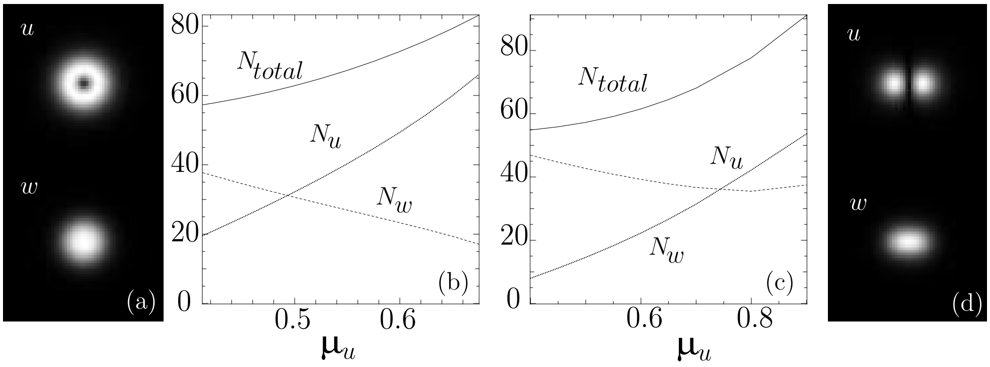


FIG. 5.2. (a) *Vortex-mode vector solitons and* (b) *their norms N_u, N_w, $N_{total} = N_u + N_w$ as a function of μ_u for $\mu_w = 1$.* (d) *Dipole-mode vector solitons and* (c) *their norms N_u, N_w, $N_{total} = N_u + N_w$ as a function of μ_u for $\mu_w = 1$.*

define the error functional

$$F(U) \equiv \int f(U)^\dagger f(U) \geq 0, \tag{5.13}$$

and take our variational principle to be *find U_0 such that F reaches an absolute minimum $F(U_0) = 0$.*

With this principle and Sobolev preconditioning our descent technique becomes

$$\frac{\partial \nu(\mathbf{x},\tau)}{\partial \tau} = -\nabla_S F(\nu) = -\left(1 - \triangle\right)^{-1} \nabla F, \tag{5.14}$$

and it may be proven that $U_0(\mathbf{x}) = \lim_{\tau\to\infty} \nu(\mathbf{x},\tau)$. The ordinary gradient ∇F reads

$$G \equiv (-\triangle + I(|U|^2))U, \tag{5.15}$$

$$\nabla F = (-\triangle + I(|U|^2))G + I'(|U|^2)\left(U^\dagger G + G^\dagger U\right) U. \tag{5.16}$$

Its calculation using a discrete Fourier basis is straightforward.

There are several advantages to this approach. The first one is that $F(U)$ makes no distinction between ground and excited states: any stationary state with the right eigenvalues $\mu_i = M_{ii}$ is a local minimum of this new functional. The second one is that we do not need to add any Lagrange multipliers to $F(U)$ since they are already present in the M operators. Finally we expect the minima of $F(U)$ to be finite, discrete, and separated so as to avoid problems with descent methods being trapped on critical points that are not minima. Indeed, it is very easy for teach the computer to know when this accidental trapping happens since any absolute minimum of F must also be a zero of it, $F(U_0) = 0$.

A remarkable feature of the method that we have outlined above is that to distinguish among the different excited states we have to change both the ad hoc eigenvalues, $\{\mu_u, \mu_w\}$, and the initial data of the minimization method.

We have concentrated on two types of *excited states* of particular interest for applications: the first one is called the *vortex vector soliton* [8], its features being summarized in Figures 5.2(a)–(b). Choosing different initial data for the minimization process, we obtain a second type of excited states which are called *dipole-mode vector solitons* [9]. An example of these asymmetric solutions is depicted in Figure 5.2(d).

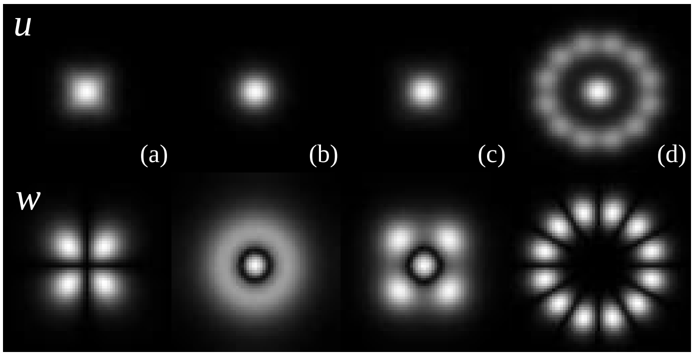


FIG. 5.3. *Different unstable multisolitonic configurations arising from* (5.14) *with a change of the initial conditions for fixed* $\mu_u = -1, \mu_w = -0.3, \kappa = 0.5$.

Depending on the parameters $\mu_{u,w}$ we find different norms of the solution. The fact that our method allows the finding of these nonsymmetric stationary states is very interesting since conventional approaches to the problem have severe difficulties. In fact, the application of the Sobolev preconditioning not only greatly enhances the convergence but is necessary for obtaining convergence.

Nevertheless the method works equally well for more complicated stationary solutions. In Figure 5.3 we show several exotic solutions that are obtained by solving (5.14) with different initial conditions. All of those states are dynamically unstable and could have not been found with traditional minimization methods.

5.5. Performance and grid refinement. In this section we want to analyze the performance of the method from section 5.4 and to introduce a multigrid-like technique to improve convergence rates while looking for more accurate solutions.

The idea of the multigrid technique is to use solutions from coarse-grain grids to calculate better approximations on finer grids [10]. Roughly the algorithm consists of setting a coarse-grain initial data, solving the equation or the variational principle with this initial data until the error is small enough, interpolating the result over a finer grid, and iterating using this new grid until both the error and the spatial discretization of the solution are the ones we desire.

Since we are already working with Fourier modes over discrete grids the logical choice for our algorithm is indeed Fourier interpolation. The idea is to use the expansion from (3.3) *outside of the original grid*, that is,

$$\psi^{(new)}(x) \equiv \sum_n c_n e^{i\mathbf{k}_n \mathbf{x}} \quad \forall \mathbf{x} \in \Omega. \tag{5.17}$$

This approximation is then discretized over a finer grid, resulting in an expansion with a larger number of modes,

$$\psi_m^{(new)} = \sum_n c^{(new)} e^{i\mathbf{k}_\mathbf{n}^{(\mathbf{new})} \mathbf{x}_m}, \tag{5.18}$$

which may be used as the starting point for further iteration.

In our case we have used only two different grids, one with 32×32 points (*coarse grid*) and another with 64×64 points (*fine grid*). We have used two different initial data for our minimization method: either a pair of Hermite modes which resemble

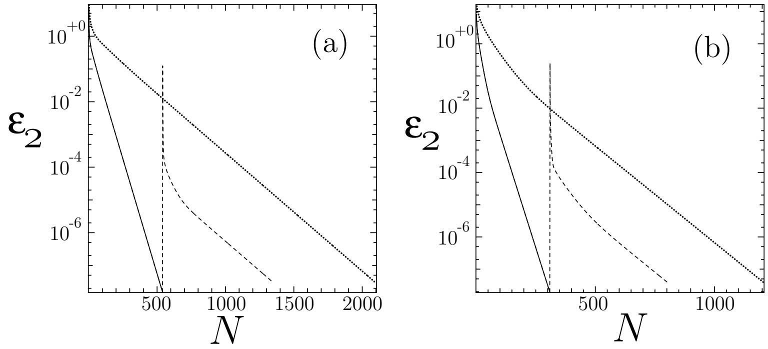


FIG. 5.4. *Dependence of the L^2 error on the number of iterations when looking for* (a) *a* vortex-mode vector soliton *or* (b) *a* dipole-mode vector soliton. *We show results for a grid with* 32 × 32 *points (solid),* 64 × 64 *points (dotted), and* 64 × 64 *points starting from an interpolation of the* 32 × 32 *solution (dashed).*

TABLE 5.1
Results for the search of stationary solutions of (5.1)–(5.2) *with and without grid refinement. Shown are the results for different initial data. The initial data named "interpolated solution" corresponds to taking as initial data the approximated solution on the* 32 × 32 *grid.*

	Initial data type					
	Hermite modes for vortex solitons		Interpolated solution for vortex	Hermite modes for dipole solitons		Interpolated solution for dipole
Grid	32×32	64×64	64×64	32×32	64×64	64×64
Iterations	540	2101	812	496	1206	496
Time (s)	106	2006	757	60	1141	466

the desired shape or the solution of the coarse grid interpolated over the finer grid. As both the evolution of the error in Figure 5.4 and the computation times in Table 5.1 show, interpolation saves a significant amount of time. Indeed, the interpolated solution with only 32×32 Fourier modes is already a good approximation for the fine grid, as it shows the small change of the error in Figure 5.4

6. Conclusions.

In this paper we have shown a numerically efficient way to improve the convergence of several minimization methods using the so-called Sobolev gradients and applied it to different problems which involve nonlinear Schrödinger equations. We have also proven that these gradients represent a preconditioning over the traditional definition of gradients on L^2. In Appendix B we suggest a generalization of this method to different vector spaces and scalar products.

We have presented two different methods for solving our minimization problems: a traditional one, the *imaginary time evolution*, and a new one, *the minimization of a nonlinear free energy*. Both methods have been shown to be suitable for Sobolev preconditioning, giving us two new methods that we call *preconditioned imaginary time evolution* and a *preconditioned free energy*.

We have implemented all four methods using a discrete Fourier basis and an FFT. With these tools we have shown that the Sobolev preconditioning becomes an inexpensive additional step over existing methods. The four resulting solvers have been applied to several realistic problems, and in all tests the nonlinear free energy with the Sobolev preconditioning showed the best convergence rates. Indeed the Sobolev preconditioning has an important effect on convergence, which can be as good as gaining two orders of magnitude over the traditional techniques. Furthermore, in contrast to what happens with finite differences [2], the preconditioning may be applied without significant computational cost.

We have derived a new method for finding excited states of coupled nonlinear Schrödinger equations. This method introduces a new variational principle which is not based on an energy functional but on finding the zeros of a nonlinear operator which corresponds to the equation to be solved. We have also shown how to improve convergence using Fourier interpolation and a two-grid method as a source for better initial approximations of the iterative method. This approach may be extended to more sophisticated multigrid methods.

Appendix A. Existence of minimizers for the nonlinear free energy functional. In this appendix we want to prove the existence of minimizers for the free energy (4.14). Let us write the free energy functional in the following form:

$$F(\psi) = \int \bar{\psi} A_\Omega \psi d^n x + \int \frac{g}{2}|\psi|^4 d^n x + \frac{1}{2}\left(N-\lambda\right)^2, \tag{A.1}$$

where $N = \int |\psi|^2 d^n r$ and

$$A_\Omega = -\frac{1}{2}\triangle + V(\mathbf{x}) - \Omega L_z, \quad \Omega \in [0,1), \tag{A.2}$$

is a positive Hermitian operator $A_\Omega \geq \mu_{min} > 0$.

Let us also define the following spaces of functions defined over $\mathbb{R}^n$. We will work in L^p spaces:

$$L^p = \left\{ \psi : \mathbb{R}^2 \to \mathbb{C} \;/\; \|\psi\|_p := \left(\int |\psi(\mathbf{x})|^p d^n x \right)^{1/p} < +\infty \right\}. \tag{A.3}$$

Also of interest will be the space of functions over which A_Ω is well defined:

$$H_\Omega = \left\{ \psi : \mathbb{R}^2 \to \mathbb{C} \;/\; \|\psi\|_\Omega := \left(\int \bar{\psi}(\mathbf{x}) A_\Omega \psi(\mathbf{x}) \right)^{1/2} < +\infty \right\}. \tag{A.4}$$

Let us remark that the dimensionality of the space, $\mathbb{R}^2$, is important, since for $n \leq 2$ it follows that $H_\Omega \subset L^4$.

With the preceding notation we will state the following relevant theorem.

Theorem A.1. *The free energy functional F given by* (A.1) *has at least one absolute minimum in the set given by the inequalities*

$$0 < \|\psi\|_\Omega^2 \leq \lambda \|\psi\|_2^2 \leq \lambda(\lambda - \mu_{min}).$$

Proof. The first step of the proof will be to show that the domain of F is indeed the whole space H_Ω. Using the positivity of the A_Ω operator we show that

$$\|\psi\|_2 \leq \sqrt{\mu_{min}} \|\psi\|_\Omega, \tag{A.5}$$

which means that $H_\Omega \subset L^2$. We need Sobolev's inequality [11]

$$\|\psi\|_k \leq \|\psi\|_2^{1-d} \|\nabla \psi\|_2^d, \tag{A.6}$$

where $d = n/2 - n/k$, n is the dimensionality of the space, and for us $d = 1/2$. Applying this inequality we obtain the following bound:

$$\|\psi\|_4 \leq \sqrt{\|\psi\|_2 \|\nabla\psi\|_2} \leq \mu_{min}^{\frac{1}{4}} \|\psi\|_\Omega, \tag{A.7}$$

which means that $H_\Omega \subset L^4$. Since $\|\cdot\|_2$, $\|\cdot\|_4$, $\|\cdot\|_\Omega < +\infty$ inside H_Ω we conclude that F as given by (A.1) is well defined over the whole space.

F is also continuous. To prove it let us rewrite the free energy functional as

$$F(\psi) = \|\psi\|_\Omega^2 + \frac{g}{2}\|\psi\|_4^4 + \frac{1}{2}(\|\psi\|_2^2 - \mu)^2, \tag{A.8}$$

and using the bounds (A.5) and (A.7) one shows that

$$|F(\psi) - F(\xi)| \leq \delta(\varepsilon) \quad \forall \xi \ : \ \|\psi - \xi\|_\Omega \leq \epsilon, \tag{A.9}$$

where the constant $\delta(\varepsilon)$ is given by ε and $\|\psi\|_4$.

We will also need to show that F is coercive:

$$\lim_{\|\psi\|\to\infty} \frac{F(\psi)}{\|\psi\|} \geq \alpha > 0. \tag{A.10}$$

Using (A.8) one may show that indeed

$$\frac{F(\psi)}{\|\psi\|_\Omega} \geq \|\psi\|_\Omega, \tag{A.11}$$

and thus the quotient tends to infinity as the norm grows. The continuity and coercitivity of F allow us to use Theorem 1 of [12, section 1.2], which states the existence of at least one minimizer $\{\inf F(u) \ : \ u \in X\}$ of F provided it is a weakly lower semicontinuous and coercive application $F : X \to \mathbb{R}$ in the reflexive Banach space X.

So we know that there is at least one minimum, but we do not know where to look for it. Let us now show how the λ parameter allows us to select different targets for the minimization problem. To do so we define a real one-dimensional function for any given direction $\psi \in H_\Omega$,

$$f(k) \equiv F(k\psi). \tag{A.12}$$

This function is nothing but a polynomial over k,

$$f(k) = kNa_\psi + k^2 N^2 \frac{u_\psi}{2} + \frac{1}{2}(kN - \lambda)^2, \tag{A.13}$$

where $a_\psi = \int \bar{\psi} A \psi / N$ and $u_\psi = \frac{g}{2} \int |\psi|^4 / N^2$ are constants that depend only on the precise direction ψ.

By differentiating the polynomial and imposing $k = 0$ we find that $f'(0) < 0$ at the origin for all possible directions. This means that, as we mentioned above, our Lagrange "multiplier" λ allows us to avoid the useless solution, $\psi = 0$.

Furthermore, we can restrict the location of the minimizer to a surface of a certain norm. By differentiating $f(k)$ we reach

$$(a_\psi - \lambda) + Nk(u_\psi + 1) = 0. \tag{A.14}$$

This equation has a single solution which gives us the norm of minimal energy along the ψ-direction,

$$N_{min}(\psi) = \max\left\{0, \frac{\lambda - a_\psi}{u_\psi + 1}\right\} \leq \lambda - \mu_{min}, \tag{A.15}$$

a value which is bounded above by our Lagrange multiplier λ.

From (A.14) it also follows that for any absolute minimum of the functional ψ_{min}, the expected value of the A_Ω operator must be bounded by the L^2-norm

$$\int \bar{\psi}_{min} A \psi_{min} = \|\psi_{min}\|_\Omega^2 \leq \lambda N(\psi_{min}). \tag{A.16}$$

Otherwise the trivial solution $\psi = 0$ would have less energy than ψ_{min}. We can thus, instead of working with an unknown surface, delimit the location of the minimum to a set which is given by two inequalities, (A.15) and (A.16):

$$W = \{\psi \in H_\Omega \;:\; 0 < N(\psi) \leq \lambda - \mu_{min}, \|\psi\|_\Omega^2 \leq \lambda N(\psi)\}. \qquad \square \tag{A.17}$$

There are several practical consequences of this theorem. First, it states that the problem of finding the minima of $E(\psi)$ subject to fixed norm has one solution, i.e., there exists at least one ground state. Second, but equally important, it proves our Lagrange penalizer $\frac{1}{2}(N - \mu)^2$ to be specially well suited for this problem since it avoids both the useless solution $\psi = 0$ and those with too large norm. And finally it gives us a bounded set in which the minimizer must be. Indeed we can extend this result by proving that we can restrict our search space to a compact superset of W.

THEOREM A.2. *Any absolute minimum, ψ, of the functional F given by* (A.1) *lays inside the compact set of L^2,*

$$\bar{U} = \{\psi \in L^2 \;:\; \|\psi\|_\Omega^2 \leq \lambda \|\psi\|_2^2 \leq \lambda(\lambda - 1)\}.$$

Proof. For the type of spaces that we work with, a set is compact iff it is closed and we can build an ε-net for any positive number ε. The ε-net is a finite set $K_\varepsilon = \{\nu_1, \ldots, \nu_k\}$ such that for each $\psi \in \bar{U}$ there is an element $\nu \in K_\epsilon$ verifying $\|\psi - \nu\|_\Omega < \varepsilon$. Thus compactness is equivalent to the possibility of building an arbitrarily good approximation of our minimizer using a finite but sufficiently large basis of functions.

It is evident that the set $\bar{U}$ is closed in the subspace H_Ω of L^2. Let $\{u_n\} \in \bar{U}$ be a convergent sequence and let u be their limit. Since for each element of the sequence

$$\|u_n\|_\Omega \leq \sqrt{\lambda}\|u_n\|_2 < \lambda(\lambda - 1) \tag{A.18}$$

it is also obvious that

$$\|u\|_\Omega \leq \sqrt{\lambda}\|u\|_2 \leq \lambda(\lambda - 1), \tag{A.19}$$

thus also the limit belongs to $\bar{U}$.

The compactness of the closed set $\bar{U}$ essentially follows from the fact that the eigenstates of A_Ω form a complete basis of H_Ω, and that the eigenvalues of A_Ω form a monotonously growing unbounded set of positive numbers. These eigenstates are of the form

$$\phi_{n,l} = P_n^l(|\mathbf{x}|)\frac{x_1 + ix_2}{|\mathbf{x}|^2} e^{-|\mathbf{x}|^2/2}, \tag{A.20}$$

where P_n^l are Laguerre's generalized polynomials and the corresponding eigenvalues are

$$A_\Omega \phi_{n,l} = \mu_{n,l}\phi_{n,l} = (2n + (1-\Omega)l + 1)\phi_{n,l}, \quad n, l \in \mathbb{N} \cup \{0\}. \tag{A.21}$$

Let us choose any natural number k such that $k+1 > \lambda$. We can split the whole space as a direct sum $H_\Omega = H_0^{k+1} \oplus H_{k+1}^\infty$, where

$$H_j^k = \mathrm{lin}\{\phi_{n,l} \ : \ j \leq 2n + l < k\}. \tag{A.22}$$

The important point is that since H_0^{k+1} is isomorph to $\mathbb{R}^m$ for some natural number m, and $\bar{U}_k = \bar{U} \cap H_0^{k+1}$ lays inside a compact m-dimensional ball of radius $\sqrt{\lambda - 1}$, then we can always find an ε-net for $\bar{U}_k$. Furthermore, by separating

$$\psi = \psi_a + \psi_b, \quad \psi_a \in \bar{U}_k, \psi_b \in H_{k+1}^\infty, \tag{A.23}$$

and using the definition of $\bar{U}_k$ we show that the projection of ψ outside of $\bar{U}_k$ can be made arbitrarily small:

$$\|\psi_b\|_2^2 \leq \frac{\lambda}{k+1} N. \tag{A.24}$$

A direct consequence of this is that for $k > \lambda N/\varepsilon$, an ε-net of $\bar{U}_k$ is also an ε-net of $\bar{U}$, which proves the compactness.

Finally, by inspecting the eigenvalues of A_Ω and using (A.15)–(A.16) it is not difficult to see that the absolute minimum of F must lay in $\bar{U}$. □

Appendix B. Extending the Sobolev gradients. We have shown that redefining the gradient turns out to be a kind of preconditioning over the original choice of the direction of descent. Let us assume that our functional has the following form:

$$E(\psi) = \int \bar{\psi} A \psi + f(|\psi|^2, \mathbf{x}), \tag{B.1}$$

where A is a nonnegative Hermitian operator that may involve some derivatives. Let us also assume that H is a suitable space equipped with the following scalar product:

$$\langle \psi, \phi \rangle = \int \bar{\psi}\,(1 + A)\,\phi, \tag{B.2}$$

which is indeed a scalar product because $A|_H \geq 0$.

Let us rewrite the energy functional in the following way:

$$E(\psi) = \langle \psi, \psi \rangle + \int f(|\psi|^2, \mathbf{x}) - |\psi|^2. \tag{B.3}$$

The Sobolev gradient in H is

$$\nabla_A E = \psi + (1 + A)^{-1}\left[\partial_1 f - \psi\right], \tag{B.4}$$

while the so-called ordinary gradient is $\nabla E = A\psi + \partial_1 f$. Hence the preconditioning nature of the method is recovered:

$$\nabla_A E = (1 + A)^{-1}\,\nabla E. \tag{B.5}$$

We can thus think that the new choice of the scalar product aims at making the linear part of the energy functional close to some quadratic form, $\langle \psi, B\psi \rangle$, such that the new operator B is almost the unity. In our example, indeed, $B\psi = \psi$. We believe that this preconditioning will both enhance the directions of decreasing energy and have a smoothing effect on the nonlinear part. However, the problem is complicated and we know of no proof that these arguments are of general applicability.

REFERENCES

[1] J. M. Sanz-Serna, *Fourier techniques in numerical methods for evolutionary problems*, in Proceedings of the Third Granada Seminar on Computational Physics, 1994, P. L. Garrido and J. Marro, eds., Lecture Notes in Phys. 448, Springer-Verlag, Berlin, 1995, pp. 145–200.

[2] J. W. Neuberger and R. J. Renka, *Sobolev gradients and the Ginzburg–Landau functional*, SIAM J. Sci. Comput., 20 (1998), pp. 582–590.

[3] J. W. Neuberger, *Sobolev Gradients and Differential Equations*, Springer-Verlag, Berlin, 1997.

[4] J. J. García-Ripoll and V. M. Pérez-García, *Stability of vortices in inhomogeneous Bose–Einstein condensates subject to rotation: A three dimensional analysis*, Phys. Rev. A, 60 (1999), pp. 4864–4874.

[5] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer Ser. Oper. Res., Springer-Verlag, New York, 1999.

[6] The Yorick environment for scientific computing is available from ftp://ftp-icf.llnl.gov/pub/Yorick/doc/index.html.

[7] M. Frigo and S. G. Johnson, *FFTW: An adaptive software architecture for the FFT*, in Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, IEEE, Seattle, WA, 1998, pp. 1381–1384.

[8] Z. H. Musslimani, M. Segev, D. N. Christodoulides, and M. Soljacić, *Composite multihump vector solitons carrying topological charge*, Phys. Rev. Lett., 84 (1999), pp. 1164–1167.

[9] J. J. García-Ripoll, V. M. Pérez-García, E. A. Ostrovskaya, and Y. S. Kivshar, *Dipole–mode vector solitons*, Phys. Rev. Lett., 85 (2000), pp. 82–85.

[10] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial*, SIAM, Philadelphia, 2000.

[11] L. Hörmander, *The Analysis of Linear Partial Differential Operators* I, Springer-Verlag, Berlin, 1983.

[12] B. Dacorogna, *Direct Methods in the Calculus of Variations*, Springer-Verlag, Berlin, 1989.

SIAM J. SCI. COMPUT.
Vol. 23, No. 4, pp. 1335–1351

A MODIFIED FINITE VOLUME APPROXIMATION OF SECOND-ORDER ELLIPTIC EQUATIONS WITH DISCONTINUOUS COEFFICIENTS[*]

R. EWING[†], O. ILIEV[‡], AND R. LAZAROV[§]

Abstract. A modified finite difference approximation for interface problems in R^n, $n = 1, 2, 3$, is presented. The essence of the modification falls in the simultaneous discretization of any two normal components of the flux at the opposite faces of the finite volume. In this way, the continuous normal component of the flux through an interface is approximated by finite differences with second-order consistency. The derived scheme has a minimal $(2n + 1)$-point stencil for problems in R^n. Second-order convergence with respect to the discrete H^1-norm is proved for a class of interface problems. Second-order pointwise convergence is observed in a series of numerical experiments with one-dimensional (1-D), two-dimensional (2-D), and three-dimensional (3-D) interface problems. The numerical experiments presented demonstrate advantages of the new scheme compared with the known schemes which use arithmetic and harmonic averaging of the discontinuous diffusion coefficient.

Key words. finite volume method, interface problems, finite differences, elliptic problems with variable coefficients

AMS subject classifications. 65N10, 65F30

PII. S1064827599353877

1. Introduction.

Elliptic problems with discontinuous coefficients (often called interface problems) arise naturally in mathematical modeling processes in heat and mass transfer, diffusion in composite media, flows in porous media, etc. These processes are described by the model diffusion equation

$$-\nabla(K\nabla u) = f(x) \quad \text{for } x \in \Omega, \tag{1.1}$$

subject to various boundary conditions. Here $\Omega \subset R^n$ is a bounded polyhedra, and $K(x)$ is a symmetric and uniformly positive definite matrix in Ω which may have a jump discontinuity across a given surface Γ. Due to the nature of the processes, often the fluxes across Γ, defined as $-K\nabla u \cdot \mathbf{n}$, where $\mathbf{n}$ is the normal unit vector to Γ, are smooth, although the coefficients and the derivatives of the solution are discontinuous. Often the surfaces of discontinuity of the coefficient matrix $K(x)$ are called interfaces. The assumption that the solution and the normal component of the flux are continuous through the interface is physical and is often used to close the mathematical problem. In this paper we derive a new class of finite difference schemes for second-order elliptic equations with diagonal coefficient matrix $K(x) = \mathrm{diag}(k_1(x), \ldots, k_n(x))$. The derived schemes are based on finite volume techniques

[*] Received by the editors April 9, 1999; accepted for publication June 18, 2001; published electronically December 18, 2001. This work has been partially supported by U.S. National Science Foundation grants DMS-9626567 and DMS-9973328 and Bulgarian Fund for Scientific Research grant MM-811.

http://www.siam.org/journals/sisc/23-4/35387.html

[†] Institute for Scientific Computation, Texas A&M University, College Station, TX 77843-3404 (richard-ewing@tamu.edu).

[‡] ITWM, University of Kaiserlautern, Erwin-Schrodinger-Strasse, D-67663 Kaiserlautern (iliev@itwm.uni-kl.de) and Institute of Mathematics, BAS, Acad. G. Bonchev Street bl. 8, BG-1113 Sofia, Bulgaria (oleg@math.bas.bg).

[§] Department of Mathematics, Texas A&M University, College Station, TX 77843-3368 (lazarov@math.tamu.edu).

and use two main assumptions: (1) both the right-hand side $f(x)$ and the normal components of the flux across the interface are smooth enough; (2) the interfaces Γ (i.e., the surfaces where the coefficients $k_i(x)$ have jumps) are parallel to the grid planes (lines).

In the one-dimensional (1-D) case, the first assumption reduces just to the smoothness of the right-hand side $f(x)$. In the multidimensional case, these two assumptions are much more complicated and restrictive. First, the interfaces have to be planes parallel to the coordinate planes. Second, the smoothness properties of the solution will depend on the smoothness of the boundary of the domain, the smoothness of the interface Γ, and the ratio of the coefficient jumps in a pretty complicated manner. Some particular results in this direction can be found in the fundamental work of Kondratiev [11]. In general, when the normal component of the flux is smooth it makes sense to use schemes with better approximation properties away from the corners and the points of intersection of the interface Γ with itself or with the boundary $\partial\Omega$.

Finite difference schemes obtained from discretization of the balance equation over a finite number of control volumes have been widely used in computational practice for differential equations. In the early stages, these were finite difference schemes on rectangular meshes with quite complicated treatment of the coefficients and the right-hand side (see, for example, the classical books [14, 16] and references therein). In [20, 21], Tikhonov and Samarskii derived an $O(h^{2m+1})$-accurate finite difference scheme, where $m \geq 0$ is an arbitrary integer, for two-point boundary value problems. The coefficients of the scheme are, in general, certain nonlinear functionals of the differential equation coefficients, which were assumed to be piecewise smooth.

Further, in [18] Shashkov has extended the balance equation approximation idea to a large class of differential operators (including divergence, gradient, and curl) on quite general quadrilateral grids (see also [7]). This new approach has produced discrete operators which approximate the corresponding differential operators and have the same properties as the continuous ones. For example, the discrete gradient is adjoint in a special inner product to the discrete divergence.

In recent years, the finite volume approach has been combined with finite element method techniques in a new development which is capable of producing accurate approximations on general triangular and quadrilateral grids (see, e.g., [2, 3, 4, 10, 12, 15]). The main advantages of the method are compactness of the discretization stencil, good accuracy, and local discrete conservation. In all of these discretization methods, it is assumed that the possible jumps of the diffusion coefficient are aligned with the finite element partitioning. This means that inside each finite element the diffusion coefficient is sufficiently smooth, and the jumps may occur only at the finite element boundaries.

A straightforward application of the finite volume method to a generic interface problem results in a scheme which uses harmonic averaging of the coefficient. This is particularly important in the case of discontinuous coefficients (see, for example, [16]). Inspecting these schemes, one easily sees that the normal component of the flux at the interface is discretized with a local truncation error $O(h)$. In this paper we present a modification of the classical finite volume method so the normal component of the flux in the new scheme has $O(h^2)$-local truncation for interface problems with smooth normal flux. Note that we do not suppose that the interfaces are aligned with finite volume surfaces. However, we assume that the interfaces are orthogonal to the coordinate axes. Our approach can be viewed as a defect correction of the standard scheme with harmonic averaging of the coefficient, since it takes into account the next term in the Taylor expansion of the flux. This correction does preserve the standard $(2n+1)$-point overall stencil and uses data only from the neighboring 2^n cells. We were able to increase the order of the local truncation error and at the same time preserve the standard stencil by discretizing the normal components of the flux at the opposite sides of the finite volume as a couple.

Recently, Il'in [9] and LeVeque and Li [13] have derived second-order finite difference approximations of the two-dimensional (2-D) interface problem using similar assumptions about the normal flux through the interface. However, in order to get a second-order scheme, Il'in [9] uses a larger stencil than the compact $(2n+1)$-point stencils for problems in R^n, while LeVeque and Li [13] use Taylor expansions for the solution around the interface. The latter paper does not consider the discretization of the fluxes, and this can be viewed as a disadvantage when the problem requires their accurate reconstruction, e.g., with velocities in porous media or the heat fluxes in thermal problems. Below we propose a homogeneous difference scheme for a class of 1-D, 2-D, and three-dimensional (3-D) elliptic problems with variable discontinuous coefficients with arbitrarily located interfaces. The coefficients of the scheme are obtained from the coefficients of the differential equation by a simple formula. The approach of LeVeque and Li from [13] requires solving small systems of linear equations for determining these coefficients at each point near the interface. Moreover, our scheme is easily extendible to fine-scale inhomogeneities of the coefficients (finer than the grid size). However, our approach deals only with interfaces orthogonal to the coordinate axes, while the approach from [13] can treat arbitrarily located interfaces.

Below we summarize advantages and disadvantages of the new scheme in comparison with known ones for grids not aligned with the diffusion coefficient jump. On the positive side are the following features of the new scheme: (1) the scheme has $O(h^2)$-local truncation error for the normal component of the flux; recall that the standard schemes with arithmetic and harmonic averaging of the coefficient at the interface have, in general, local truncation error $O(1)$ and $O(h)$, respectively; (2) the proposed scheme is algebraically equivalent to a scheme which is second-order consistent with the interface differential problem; (3) the numerical experiments for problems with large jumps of the diffusion coefficient demonstrate that the new scheme is orders of magnitude more accurate than the scheme which uses harmonic averaging.

On the negative side are the following two main disadvantages: (1) in general, the scheme is only asymptotically (for $h \to 0$) locally conservative; (2) the corresponding matrix is a nonsymmetric M-matrix; this will add some costs to the solution method for the algebraic problem. However, the numerical experiments on a wide class of problems with discontinuous coefficients show that the scheme is so accurate that these two disadvantages cannot diminish the value of the method.

We have run several numerical experiments in order to validate the new scheme and to compare it with the known schemes. These experiments include solving 1-D, 2-D, and 3-D interface problems with known analytical solutions, as well as solving a 2-D problem with a singular solution. Also, we considered problems where the interfaces are aligned with the finite volume surfaces, as well as problems with arbitrarily located interfaces, orthogonal to the coordinate axes. Pointwise second-order convergence is observed in numerical experiments. Note that the accuracy of the new scheme observed in our experiments is almost uniform with respect to the jump of coefficients, and it is comparable with the accuracy of the solution of the Poisson equation with a constant diffusion coefficient. What is even more interesting is that this conclusion is valid not only for the case of interfaces aligned with the finite volume boundaries but also for the nonaligned case. Meanwhile, for problems with large jumps of the coefficients, the accuracy of the scheme with harmonic averaging is very sensitive to the jump size, and its accuracy is orders of magnitude less than the accuracy of the new scheme. Numerical experiments for the so-called thin lenses problem are especially interesting. In this case our scheme provides very accurate results even on very coarse grids, in considerable contrast with the other known schemes.

The paper is organized as follows. Section 2 is devoted to the derivation and study of modified finite volume schemes for 1-D problems. Section 3 contains the formulation of the new finite volume scheme for multidimensional interface problems. Finally, section 4 summarizes and discusses the results of the numerical experiments of a series on interface problems in R^n, $n = 1, 2, 3$.

2. Modified finite volume discretization for 1-D problems. In order to illustrate our approach, we shall first consider the 1-D case and rewrite (1.1) into its mixed form: find $u(x)$ such that

$$\frac{\partial W}{\partial x} = f(x), \quad W = -k(x)\frac{\partial u}{\partial x}, \quad 0 < x < 1, \quad u(0) = 0, \quad u(1) = 0. \tag{2.1}$$

Here $k = k(x) > k_0$ is a known diffusion coefficient, $W(x)$ is the flux dependent variable, and $f(x)$ is the given source term. Conditions for continuity of the function and of the flux through interface points ξ are added:

$$[u] = [W] = 0 \quad \text{for } x = \xi. \tag{2.2}$$

Here $[u]$ denotes the difference of the right and left limits of u at the point of discontinuity. The main assumptions for this problem are (1) the coefficient $k(x)$ has a finite number of jump discontinuities and in the closed intervals between the jumps $k(x)$ is twice continuously differentiable; (2) the right-hand side $f(x)$ is continuous and has the continuous first derivative on the closed interval $[0, 1]$.

We introduce a standard uniform cell-centered grid $x_0 = 0$, $x_1 = h/2$, $x_i = x_{i-1} + h$, $i = 2, \ldots, N$, $x_{N+1} = 1$, where $h = 1/N$. Note that the endpoints $x = 0$ and $x = 1$ are part of the grid, but they are at $h/2$ distance from their neighboring grid points. This type of shifted grid is slightly inconvenient for Dirichlet boundary conditions, but it is natural and very convenient for computations when the boundary condition involves the flux W. The internal grid points can be considered as centered around the volumes $V_i = (x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}})$, where $x_{i+\frac{1}{2}} = x_i + \frac{1}{2}h$, $x_{i-\frac{1}{2}} = x_i - \frac{1}{2}h$. The values of a function f defined at the grid points x_i are denoted by f_i. Nonuniform grids can be treated in a similar way. A reason to work with cell-centered grids is that they are widely used, say, in the computational fluid dynamics. Considering, for example, nonisothermal fluid-structure interaction problems, one has to solve problems close to the one considered here. However, our approach is defined locally, at a particular finite volume level, and it can work with standard vertex-based grids as well.

The finite volume method exploits the idea of writing the balance equation over the finite volume V_i, i.e., integrating (2.1) over each volume V_i:

$$W_{i+\frac{1}{2}} - W_{i-\frac{1}{2}} = h\ \varphi_i, \quad \varphi_i = \frac{1}{h}\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} f(x)dx, \quad i = 1, 2, \ldots, N. \tag{2.3}$$

Next, we rewrite the flux equation in the form

$$-\frac{\partial u}{\partial x} = \frac{W(x)}{k(x)}$$

and integrate this expression over the interval (x_i, x_{i+1}):

$$-(u_{i+1} - u_i) = -\int_{x_i}^{x_{i+1}} \frac{\partial u}{\partial x} dx = \int_{x_i}^{x_{i+1}} \frac{W(x)}{k(x)} dx. \tag{2.4}$$

We assume that flux $W(x)$ is two times continuously differentiable on the interface so it can be expanded around the point $x_{i+\frac{1}{2}}$ in Taylor series

$$W(x) = W_{i+\frac{1}{2}} + (x - x_{i+\frac{1}{2}}) \frac{\partial W_{i+\frac{1}{2}}}{\partial x} + \frac{(x - x_{i+\frac{1}{2}})^2}{2} \frac{\partial^2 W(\eta)}{\partial x^2}, \quad \eta \in (x_i, x_{i+1}). \tag{2.5}$$

After replacing the first derivative of the flux at $x_{i+\frac{1}{2}}$ by a two-point backward difference, we get the following approximation of (2.4):

$$\begin{aligned} -(u_{i+1} - u_i) = W_{i+\frac{1}{2}} &\int_{x_i}^{x_{i+1}} \frac{dx}{k(x)} \\ &+ \frac{W_{i+\frac{1}{2}} - W_{i-\frac{1}{2}}}{h} \int_{x_i}^{x_{i+1}} \frac{(x - x_{i+\frac{1}{2}})}{k(x)} dx + O(h^3). \end{aligned} \tag{2.6}$$

Finally, we rewrite this equation in the following basic form:

$$-k^H_{i+\frac{1}{2}} \frac{u_{i+1} - u_i}{h} = W_{i+\frac{1}{2}} + a_{i+\frac{1}{2}}(W_{i+\frac{1}{2}} - W_{i-\frac{1}{2}}) + \psi_i, \tag{2.7}$$

where

$$k^H_{i+\frac{1}{2}} = \left(\frac{1}{h} \int_{x_i}^{x_{i+1}} \frac{dx}{k(x)} \right)^{-1}, \quad a_{i+\frac{1}{2}} = k^H_{i+\frac{1}{2}} \frac{1}{h^2} \int_{x_i}^{x_{i+1}} \frac{x - x_{i+\frac{1}{2}}}{k(x)} dx, \quad \psi_i = O(h^2). \tag{2.8}$$

Here $k^H_{i+\frac{1}{2}}$ is the well-known harmonic averaging of the coefficient $k(x)$ over the cell (x_i, x_{i+1}), which has played a fundamental role in deriving accurate schemes for discontinuous coefficients (see, e.g., [14, 16]). This presentation of the flux $W(x)$ is a starting point for our discretization. Since we have assumed that the flux is smooth, then the consecutive terms in the right-hand side in (2.7) are $O(1)$, $O(h)$, and $O(h^2)$, respectively. Truncation of this sum after the first term produces the well-known scheme of Samarskii [16] with harmonic averaging of the coefficient. This scheme is $O(h)$-consistent at the interface points and second-order accurate in the discrete H^1-norm. Further, we call this scheme the *harmonic averaging* (HA) scheme. The scheme we shall derive takes the two terms of the presentation (2.7) and disregards the ψ_i-term. Let $F_{i+\frac{1}{2}}$ and $F_{i-\frac{1}{2}}$ denote the approximation to the exact fluxes $W_{i+\frac{1}{2}}$ and $W_{i-\frac{1}{2}}$, respectively, and let y_i denote the approximate values of the exact solution $u(x_i)$. Thus we get the following relations:

$$-k^H_{i+\frac{1}{2}} \frac{y_{i+1} - y_i}{h} = F_{i+\frac{1}{2}} + a_{i+\frac{1}{2}}(F_{i+\frac{1}{2}} - F_{i-\frac{1}{2}}), \tag{2.9}$$

$$-k^H_{i-\frac{1}{2}} \frac{y_i - y_{i-1}}{h} = F_{i-\frac{1}{2}} + a_{i-\frac{1}{2}}(F_{i+\frac{1}{2}} - F_{i-\frac{1}{2}}). \tag{2.10}$$

The two relations above allow us to derive the final expression for $F_{i+\frac{1}{2}} - F_{i-\frac{1}{2}}$, which is needed in the balance equation (2.3): subtract (2.10) from (2.9) to get

$$(1 + a_{i+\frac{1}{2}} - a_{i-\frac{1}{2}})(F_{i+\frac{1}{2}} - F_{i-\frac{1}{2}}) = -k^H_{i+\frac{1}{2}} \frac{y_{i+1} - y_i}{h} + k^H_{i-\frac{1}{2}} \frac{y_i - y_{i-1}}{h}. \tag{2.11}$$

Since the grid points x_0 and x_{N+1} are shifted by $h/2$ from their neighbors, the finite difference equations at x_1 and x_N have to be modified. Combining (2.11) with (2.3), we get the following finite difference approximation of the differential problem (2.1):

$$L_h y_i = \varphi_i \quad \text{for } i = 1, \ldots, N, \tag{2.12}$$

where

$$L_h y_i \equiv \begin{cases} -\frac{4}{3}\frac{1}{h}\left(k^H_{\frac{3}{2}}\frac{y_2 - y_1}{h} - k^H_{\frac{1}{2}}\frac{2y_1}{h}\right) & \text{for } i = 1, \\ -\left(1 + a_{i+\frac{1}{2}} - a_{i-\frac{1}{2}}\right)^{-1}\frac{1}{h}\left(k^H_{i+\frac{1}{2}}\frac{y_{i+1} - y_i}{h} - k^H_{i-\frac{1}{2}}\frac{y_i - y_{i-1}}{h}\right), & i \neq 1, N, \\ -\frac{4}{3}\frac{1}{h}\left(-k^H_{N+\frac{1}{2}}\frac{2y_N}{h} - k^H_{N-\frac{1}{2}}\frac{y_N - y_{N-1}}{h}\right) & \text{for } i = N. \end{cases} \tag{2.13}$$

Here in the first and last difference equations, we have explicitly imposed the homogeneous boundary conditions $y_0 = 0$ and $y_{N+1} = 0$. Further, in the text we refer to this approximation as a scheme using the *improved harmonic averaging* (IHA) scheme.

REMARK 2.1. *For $a_{i+\frac{1}{2}} = a_{i-\frac{1}{2}} = 0$, the operator L_h is the well-known finite difference operator corresponding to harmonic averaging of the diffusion coefficient, which is second-order accurate in the discrete H^1-norm* (*see, e.g.,* [16]).

REMARK 2.2. *Near the boundary the discretization* (2.13) *has a special form due to the use of a cell-centered grid. The well-known discretization for such grids uses the factor* 1 *instead of* $\frac{4}{3}$. *Although the difference scheme with the factor* 1 *is not consistent with the differential problem at the points x_1, x_N, it is proven* (*see, for example,* [23]) *that this does not influence the order of convergence of the scheme. We prefer the factor* $\frac{4}{3}$ *because the fluxes at $x = 0$ and $x = 1$ are $O(h^2)$-accurate in this case. Moreover, the numerical experiments of numerous test problems with continuous and discontinuous coefficients showed that the constant in the convergence was smaller when the discretization* (2.12)–(2.13) *was used.*

REMARK 2.3. *Alternative ways for deriving three-point approximations of* 1*-D problems in the framework of the finite element method are discussed, for example, in* [1, 5, 22]. *In the latter work, the finite element spaces involve the local solutions of the problem* (2.1), *while in the former work the schemes are derived by the dual mixed hybrid formulation.*

In our computations for comparison of the performance of the new scheme, we also use the scheme (2.12) in which $a_{i+\frac{1}{2}} = a_{i-\frac{1}{2}} = 0$ and $k^H_{i+\frac{1}{2}}$ is replaced by $0.5(k_i + k_{i+1})$. This scheme is referred to as the *arithmetic averaging* (*AA*) scheme.

Accurate computations of the fluxes are needed in many applications. The following expression approximates the continuous flux with second-order accuracy (even at the interface):

$$F_{i+\frac{1}{2}} = \frac{-k^H_{i+\frac{1}{2}}\frac{y_{i+1} - y_i}{h}\left(1 - a_{i-\frac{1}{2}}\right) - k^H_{i-\frac{1}{2}}\frac{y_i - y_{i-1}}{h}\left(a_{i+\frac{1}{2}}\right)}{\left(1 + a_{i+\frac{1}{2}}\right)\left(1 - a_{i-\frac{1}{2}}\right) + a_{i+\frac{1}{2}}a_{i-\frac{1}{2}}} = W_{i+\frac{1}{2}} + O(h^2).$$

The new scheme approximates the fluxes to second-order accuracy, *independent* of the positions of the discontinuity of the coefficient $k(x)$. The price we paid is the necessity to evaluate the expressions $k^H_{i+\frac{1}{2}}$ and $a_{i+\frac{1}{2}} - a_{i-\frac{1}{2}}$ with an error no larger than $O(h^2)$. We shall assume that any point ξ where the coefficient $k(x)$ is discontinuous is known and can be presented in the form $\xi = x_i + \theta h$ for some i and $0 \leq \theta \leq 1$.

Obviously, if $\theta = 0$ or $\theta = 1$ (i.e., when the interfaces are aligned with grid nodes), then $a_{i+\frac{1}{2}} - a_{i-\frac{1}{2}} = O(h^2)$. Thus, disregarding this term and taking into account that $k^H_{i+\frac{1}{2}} = k(x_{i+\frac{1}{2}}) + O(h^2)$, we end up with a scheme which is the same as those obtained from finite difference or linear finite element approximations. Further, if the diffusion coefficient $k(x)$ has jump discontinuities and $0 < \theta < 1$, but the flux is still smooth, the second term in the right-hand side in (2.6) will be essential to derive a better approximation. *Note that accounting for this second term is the main difference between our approach and the standard finite volume discretization of interface problems.* In short, this term does not affect the order of convergence, but it allows us to improve the constant of convergence in the case of discontinuous coefficients and to derive more accurate difference schemes.

Now we consider some particular realizations of this scheme. If the point of discontinuity ξ is in the subinterval $[x_i, x_{i+1}]$: $\xi = x_i + \theta h$, where $0 \leq \theta \leq 1$, then the approximation of the integral in $k^H_{i+\frac{1}{2}}$ is done by splitting it into integrals over (x_i, ξ) and (ξ, x_{i+1}) and then applying the trapezoidal or midpoint rule for each integral. This approach will produce accurate enough evaluation of $k^H_{i+\frac{1}{2}}$. The following formula will be only $O(h)$-accurate:

$$k^H_{i+\frac{1}{2}} \approx \left(\frac{\theta}{k_i} + \frac{1-\theta}{k_{i+1}} \right)^{-1}.$$

Note that in the case of piecewise constant coefficients, this formula is exact. However, it is based on the use of left and right rectangular quadrature formulas, and it might not be accurate enough in the general case. Second-order accurate evaluation of the integrals is given by

$$k^H_{i+\frac{1}{2}} \approx \left[\frac{\theta}{2} \left(\frac{1}{k_i} + \frac{1}{k_{\xi-0}} \right) + \frac{1-\theta}{2} \left(\frac{1}{k_{i+1}} + \frac{1}{k_{\xi+0}} \right) \right]^{-1}. \tag{2.14}$$

Note that $k_{\xi-0}, k_{\xi+0}$ are known from the second interface condition (2.2).

Further, we continue with the second integral in (2.8). Our aim is to obtain a second-order approximation for the flux $W(x)$. We again split the integral into two integrals and apply the following trapezoidal rule for each of the two integrals:

$$\begin{aligned} \frac{1}{h^2} \int_{x_i}^{x_{i+1}} \frac{(x - x_{i+1})}{k(x)} dx &= \frac{1}{h^2} \int_{x_i}^{\xi} \frac{(x - x_{i+\frac{1}{2}})}{k(x)} dx + \frac{1}{h^2} \int_{\xi}^{x_{i+1}} \frac{(x - x_{i+\frac{1}{2}})}{k(x)} dx \\ &= \frac{\theta}{2} \left(\frac{\theta - 0.5}{k_{\xi-0}} - \frac{0.5}{k_i} \right) + \frac{1-\theta}{2} \left(\frac{0.5}{k_{i+1}} + \frac{\theta - 0.5}{k_{\xi+0}} \right) + O(h^2). \end{aligned} \tag{2.15}$$

The case of piecewise constant coefficient $k(x)$ is very important for the applications. In this case the formulas presented above are exact and reduce to

$$k^H_{i+\frac{1}{2}} = \left(\frac{\theta}{k_i} + \frac{1-\theta}{k_{i+1}} \right)^{-1} \quad \text{and} \quad a_{i+\frac{1}{2}} = \frac{1}{2} \frac{\theta(1-\theta)(k_i - k_{i+1})}{(1-\theta)k_i + \theta k_{i+1}}. \tag{2.16}$$

Obviously, if the point of discontinuity ξ is a midpoint of the grid, i.e., $\xi = x_{i+\frac{1}{2}}$, then $\theta = 1/2$ and

$$k^H_{i+\frac{1}{2}} = 2 \left(\frac{1}{k_i} + \frac{1}{k_{i+1}} \right)^{-1} \quad \text{and} \quad a_{i+\frac{1}{2}} = \frac{1}{4} \left(\frac{k_i - k_{i+1}}{k_i + k_{i+1}} \right).$$

It is reasonable to assume that the step size h is so small that if there is a jump in the coefficient $k(x)$ in the interval (x_i, x_{i+1}), then $k(x)$ is smooth at the two neighboring intervals (x_{i-1}, x_i) and (x_{i+1}, x_{i+2}). Thus

$$1 + a_{i+\frac{1}{2}} - a_{i-\frac{1}{2}} = 1 + \frac{\theta(1-\theta)}{2}\left(\frac{k_i - k_{i+1}}{(1-\theta)k_i + \theta k_{i+1}}\right) + O(h^2) \geq 1/2.$$

Similarly, we also have an estimate from above: $1 + a_{i+\frac{1}{2}} - a_{i-\frac{1}{2}} \leq 3/2$. These two estimates will guarantee that the finite difference scheme is well conditioned.

The following result is valid.

Proposition 1. *Assume that the coefficient $k(x)$ is a piecewise C^1-function and has a finite number of jump discontinuities, the grid is such that the discontinuities are at the points $x_{i+\frac{1}{2}}$, and the source term $f(x)$ is a C^1-function on* $(0,1)$. *Then the finite difference scheme* (2.12), (2.8), (2.14), (2.15) *is second-order accurate in the discrete H^1-norm; i.e., the error $e_i = u(x_i) - y_i$ satisfies the estimate*

$$||e||_{H^1} \equiv ||y-u||_{H^1} \equiv \left(\sum_{i=1}^{N} k^H_{i-1/2}(e_i - e_{i-1})^2/h\right)^{1/2} \leq M\ h^2.$$

The second-order of accuracy in H^1 follows from the second-order of discretization for the fluxes by using the classical technique for deriving a priori estimates for the solution of the finite difference scheme (see, e.g., [16, 17]).

Remark 2.4. *Note that if $f(x) \equiv 1$, then $W''(x) \equiv 0$ and the local truncation error is identically zero. This means that the IHA scheme is exact (i.e., it reproduces exactly the solution at the grid points) for problems with a piecewise constant diffusion coefficient and a constant right-hand side, while the HA scheme is exact only for homogeneous problems. Thus the HA scheme reproduces exactly piecewise linear solutions, while the IHA scheme reproduces exactly piecewise quadratic solutions.*

3. Modified finite volume discretization for 3-D problems. In this section we shall introduce the finite difference scheme for the equation (1.1) in R^3 with homogeneous Dirichlet boundary conditions on a rectangular domain Ω. Now we introduce the flux $\mathbf{W} = -K(x)\nabla u$. If the diffusion coefficient is discontinuous on a certain surface (so-called interface denoted by Γ), then two conditions for continuity of the solution and the normal component of the flux through the interface are added:

$$[u] = 0, \quad [\mathbf{W}\cdot\mathbf{n}] = 0, \quad x \in \Gamma, \tag{3.1}$$

where $[g]$ denotes the difference of the limit values of the function g from both sides of Γ and $\mathbf{n}$ is the unit vector normal to Γ.

In this paper we consider multidimensional problems that can be discretized in a coordinatewise way. This means that the interfaces are parallel to the faces of the finite volumes, and the diffusion coefficient matrix $K(x)$ is a diagonal. Thus the discretization of a 3-D problem is obtained as a tensor product discretization of three 1-D problems (like the one investigated in the preceding section).

The finite volume approach is used for discretizing the above equation on cell-centered grids which are tensor products of grids in each direction. The grid sizes and the number of the nodes in the x_i-direction is h_i and N_i for $i = 1,2,3$. The grid points are denoted by $(x_{1,i}, x_{2,j}, x_{3,k})$, where $0 \leq i \leq N_1$, $0 \leq j \leq N_2$, $0 \leq k \leq N_3$. The values of the unknown function are related to the volumes' centers. The discretization at the internal points is based on the local flux balance for the finite volume around the point. For the finite volume V_P corresponding to node P this balance is

$$\int_{\partial V_P} \mathbf{W}\cdot\mathbf{n}ds = h_1h_2h_3\varphi_P, \quad \varphi_P = \frac{1}{h_1h_2h_3}\int_{V_P} f(x)dx. \tag{3.2}$$

Here $\mathbf{n}$ is the unit outward normal to the volume boundary ∂V_P. Next, we approximate the integrals over the volume faces by the midpoint rule to get

$$h_2h_3\left(W_e - W_w\right) + h_3h_1\left(W_n - W_s\right) + h_1h_2\left(W_t - W_b\right) = h_1h_2h_3\varphi_P + O(h^2). \tag{3.3}$$

Subscripts with capital letters W, E, S, N, B, T are used to denote the values at the west, east, south, north, bottom, and top neighboring grid points; and the subscript P is used for the center of the stencil, while w, e, s, n, b, t stand for the respective values in the center points of the control volume faces. For example, $W_e = -k_1\frac{\partial u}{\partial x_1}|_{x_e}$ is the flux through a face perpendicular to the axis x_1 at the point $x_e = (x_{1,i+\frac{1}{2}}, x_{2,j}, x_{3,k})$, and the grid point P is denoted by $x_P = (x_{1,i}, x_{2,j}, x_{3,k})$. The grid point east of P is denoted by $x_E = (x_{1,i+1}, x_{2,j}, x_{3,k})$, while that north of P is $x_N = (x_{1,i}, x_{2,j+1}, x_{3,k})$, etc. Further, we approximate the differences $W_e - W_w$, $W_n - W_s$, and $W_t - W_b$ as 1-D fluxes in the directions x_1, x_2, and x_3, correspondingly, using formula (2.11) in each direction.

In the particular case when the diffusion coefficient is a constant within any finite volume and the interfaces are aligned with finite volume surfaces (i.e., $\theta = 0.5$), the finite volume scheme, approximating the 3-D problem and preserving second-order of discretization for the normal components of the fluxes through interfaces, can be written as

$$\begin{aligned} &h_2h_3\mu_1^{-1}\left[k_e^H\,\frac{y_E - y_P}{h_1} - k_w^H\frac{y_P - y_W}{h_1}\right] + h_3h_1\mu_2^{-1}\left[k_n^H\frac{y_N - y_P}{h_2} - k_s^H\frac{y_P - y_S}{h_2}\right] \\ &\quad + h_1h_2\mu_3^{-1}\left[k_u^H\frac{y_T - y_P}{h_3} - k_d^H\frac{y_P - y_B}{h_3}\right] = h_1h_2h_3\varphi_P, \end{aligned} \tag{3.4}$$

where

$$\begin{aligned} \mu_1 &= \left[1 + \frac{1}{4}\left(\frac{k_{1,P} - k_{1,E}}{k_{1,P} + k_{1,E}} + \frac{k_{1,P} - k_{1,W}}{k_{1,P} + k_{1,W}}\right)\right], \\ \mu_2 &= \left[1 + \frac{1}{4}\left(\frac{k_{2,P} - k_{2,N}}{k_{2,P} + k_{2,N}} + \frac{k_{2,P} - k_{2,S}}{k_{2,P} + k_{2,S}}\right)\right], \\ \mu_3 &= \left[1 + \frac{1}{4}\left(\frac{k_{3,P} - k_{3,T}}{k_{3,P} + k_{3,T}} + \frac{k_{3,P} - k_{3,B}}{k_{3,P} + k_{3,B}}\right)\right]. \end{aligned}$$

Here k_e^H stands for harmonic averaging of $k_1(x)$ in the direction east from P, i.e., over the interval $(x_{1,i}, x_{1,i+1})$, $k_{1,P}$ is its value at the point P, etc. These finite difference equations are written for all internal points except those for which the $(2n + 1)$-stencil includes points at the boundary. For these points, essentially one has to add the modification of the approximation at the direction of the neighboring boundary point. Such modification has been introduced for 1-D problems in section 2 (see formulas (2.12)–(2.13)). To close the system to this set of finite difference equations, we add the equations accounting for the Dirichlet boundary conditions.

Note that in the case when the interfaces are not aligned with the finite volume surfaces (but are orthogonal to the axes), the multidimensional θ-HA scheme, as well as the multidimensional θ-IHA scheme, are derived as tensor products of the respective 1-D schemes.

It is obvious that the finite difference scheme can be written as a linear system of algebraic equations with a nonsymmetric M-matrix. If the coefficients $k_i(x)$, $i = 1,\dots,n$, are C^2-functions in the whole domain, then the factors μ_1, μ_2, and μ_3 are all of the order $O(h^2)$ and the nonsymmetry is negligible. On the other hand, $\mu_i > 1/2$, $i = 1,2,3$, for grids with jump discontinuities of the coefficient $k_i(x)$ parallel to the grid faces, regardless of the size of the jump. Therefore, although the condition number of the linear system will depend on the size of the jump, this dependence will be the same as in the case of arithmetic or harmonic averaging. The finite difference scheme (3.4) is the IHA scheme that has been used in our numerical experiments for both 2-D and 3-D problems, in the case when the interfaces are aligned with the finite volume surfaces. The multidimensional θ-HA scheme and θ-IHA scheme are used in the nonaligned case.

4. Numerical experiments.

A series of computational experiments were performed in order to experimentally study the accuracy and the convergence rate of the new scheme and to compare it with known schemes for 1-D, 2-D, and 3-D interface problems. Two kind of problems were solved in 1-D and 2-D cases. The first one is a problem with the known analytical solution with the right-hand side being calculated from the known solution. The second one is a problem with the right-hand side identically equal to 1. Note that in the 1-D case this problem also has an analytical solution. Only problems with known analytical solutions are solved in the 3-D case.

The relative discrete maximum norm (denoted as C-norm) of the solution error is calculated as $\max|u-y|/\max|u|$. Also, the relative discrete L_2-norm of the solution error is computed as $(\sum_V \operatorname{meas}(V)(u-y)^2)^{\frac{1}{2}}/\max|u|$. Here the operations max and the summation are considered over all grid nodes. The relative C- and L_2-norms of the error are reported in the tables below for cases when the analytical solution is known. In all tables we have used the following shorthand notations: B stands for the problem when $k(x) \equiv 1$; i.e., we solve the Poisson equation; AA stands for schemes with arithmetic averaging; HA stands for schemes with harmonic averaging; and, finally, IHA is used for a heading with the results obtained by the new scheme which uses improved harmonic averaging.

4.1. Numerical experiments for 1-D problems.

Results from the problem computation with exact solution $u^{ex} = \frac{1}{k}\sin(\frac{\pi x}{2})(x-\frac{1}{2})(1+x^2)$ and diffusion coefficient equal to 1 for $0 < x < 0.5$ and equal to 10^{-4} for $0.5 < x < 1$ are presented in Tables 4.1 and 4.2.

The results from solving the Dirichlet problem with the right-hand side identically equal to 1 are presented in Tables 4.3 and 4.4. The diffusion coefficient in this case is 1 for $0 < x < 0.4$, 10^{-3} for $0.4 < x < 0.7$, and 10 for $0.7 < x < 1$. Note that this problem has a piecewise quadratic solution.

The results from numerical experiments in the 1-D case demonstrate that the new scheme has a much smaller constant of convergence than the scheme based on harmonic averaging. Both schemes asymptotically converge with second-order, as predicted by the theory. Tables 4.3 and 4.4 confirm the theory that the IHA scheme is exact for interface problems with piecewise quadratic solutions.

4.2. Numerical experiments for 2-D problems.

Here we consider an isotropic case, i.e., $k_i(x) = k(x)$, $i = 1,\dots,n$. First of all, a 2-D interface problem with a different coefficient in four subregions and with a known analytical solution is solved.

Table 4.1
1-D problem with $u^{ex} = \frac{1}{k}\sin(\frac{\pi x}{2})(x-\frac{1}{2})(1+x^2)$; $k = \{1, 10^{-4}\}$ *in two subregions, respectively; the relative* C*-norms of the error and their ratios.*

	$k = \{1, 10^{-4}\}$					
Nodes	Case AA		Case HA		Case IHA	
12	4.95d-2	–	4.49d-3	–	5.41d-4	–
22	2.34d-2	*2.12*	1.28d-3	*3.51*	1.98d-4	*2.73*
42	1.13d-2	*2.07*	3.24d-4	*3.95*	5.90d-5	*3.35*
82	5.60d-3	*2.02*	8.14d-5	*3.98*	1.60d-5	*3.69*
162	2.78d-3	*2.01*	2.04d-5	*3.99*	4.17d-6	*3.84*

Table 4.2
1-D problem with $u^{ex} = \frac{1}{k}\sin(\frac{\pi x}{2})(x-\frac{1}{2})(1+x^2)$; $k = \{1, 10^{-4}\}$ *in two subregions, respectively; the relative* L_2*-norms of the error and their ratios.*

	$k = \{1, 10^{-4}\}$					
Nodes	Case AA		Case HA		Case IHA	
12	2.25d-2	–	2.39d-3	–	2.58d-4	–
22	1.01d-2	*2.23*	6.20d-4	*3.85*	9.50d-5	*2.72*
42	4.78d-3	*2.11*	1.58d-4	*3.92*	2.91d-5	*3.26*
82	2.32d-3	*2.06*	4.00d-5	*3.95*	8.02d-6	*3.63*
162	1.14d-3	*2.04*	1.00d-5	*4.00*	2.10d-6	*3.82*

Table 4.3
1-D problem $-(ku')' = 1$, $u(0) = 0$, $u(1) = 1$; $k = \{1, 10^{-3}, 10\}$ *in two subregions, respectively; the relative* C*-norms of the error and their ratios.*

	$k = \{1, 10^{-3}, 10\}$					
Nodes	Case AA		Case HA		Case IHA	
12	5.39d-1	–	1.06d-1	–	3.6d-15	*exact*
22	3.02d-1	*1.78*	2.70d-2	*3.93*	3.3d-15	*exact*
42	1.55d-1	*1.95*	6.63d-3	*4.07*	5.1d-15	*exact*
82	7.89d-2	*1.96*	1.65d-3	*4.02*	5.3d-15	*exact*
162	3.98d-2	*1.98*	4.13d-4	*3.99*	1.0d-14	*exact*

exact ≡ the difference scheme is exact for this problem.

Table 4.4
1-D problem $-(ku')' = 1$, $u(0) = 0$, $u(1) = 1$; $k = \{1, 10^{-3}, 10\}$ *in three subregions, respectively; the relative* L_2*-norms of the error and their ratios.*

	$k = \{1, 10^{-3}, 10\}$					
Nodes	Case AA		Case HA		Case IHA	
12	2.89d-1	–	5.79d-2	–	1.3d-15	*exact*
22	1.62d-1	*1.78*	1.48d-2	*3.91*	1.4d-15	*exact*
42	8.34d-2	*1.94*	3.63d-3	*4.08*	1.1d-15	*exact*
82	4.24d-2	*1.97*	9.04d-4	*4.02*	1.2d-15	*exact*
162	2.14d-2	*1.98*	2.26d-4	*4.00*	2.7d-15	*exact*

exact ≡ the difference scheme is exact for this problem.

We compute the solution using schemes obtained from HA and IHA averaging of the diffusion coefficient. The notations HA and IHA are preserved for the case when $\theta = \frac{1}{2}$, while notations θ-HA and θ-IHA are used for other values of θ. Two sets of values for the diffusion coefficient in the four subregions are used in order to demonstrate the influence of the size of the jump discontinuity on the accuracy of the schemes. The results from these computations are presented in Tables 4.5 and 4.6 for the first set and in Tables 4.7 and 4.8 for the second set.

Table 4.5

2-D problem with $u^{ex} = \frac{1}{k}\sin(\frac{\pi x}{2})(x - x_\xi)(y - y_\xi)(1 + x^2 + y^2)$ *and* $k = \{10^{-2}, 1, 10^{-4}, 10^{+6}\}$ *in four subregions with interfaces at* $x = x_\xi$ *and* $y = y_\xi$*; the relative* C*-norms of the error and their ratios.*

	$k(x,y) \equiv 1$		$x_\xi = \frac{1}{2}$, $y_\xi = \frac{1}{2}$ (aligned)				$x_\xi = \frac{1}{3}$, $y_\xi = \frac{1}{3}$ (nonaligned)			
Grid	Case B		HA scheme		IHA scheme		θ-HA scheme		θ-IHA scheme	
12x12	2.09d-4	–	1.75d-2	–	3.34d-4	–	1.91d-2	–	4.48d-4	–
22x22	5.74d-5	*3.6*	5.97d-3	*2.9*	7.64d-5	*4.4*	9.56d-3	*2.0*	1.45d-4	*3.1*
42x42	1.75d-5	*3.7*	1.80d-3	*3.3*	1.94d-5	*3.9*	2.10d-3	*4.6*	4.02d-5	*3.6*
82x82	4.11d-6	*3.8*	5.03d-4	*3.6*	4.97d-6	*3.9*	7.03d-4	*3.0*	1.10d-5	*3.7*
162x162	1.06d-6	*3.9*	1.36d-4	*3.7*	1.26d-6	*3.9*	1.64d-4	*4.3*	2.80d-6	*3.9*

Table 4.6

2-D problem with $u^{ex} = \frac{1}{k}\sin(\frac{\pi x}{2})(x - x_\xi)(y - y_\xi)(1 + x^2 + y^2)$ *and* $k = \{10^{-2}, 1, 10^{-4}, 10^{+6}\}$ *in four subregions with interfaces at* $x = x_\xi$ *and* $y = y_\xi$*; the relative* L_2*-norms of the error and their ratios.*

	$k(x,y) \equiv 1$		$x_\xi = \frac{1}{2}$, $y_\xi = \frac{1}{2}$ (aligned)				$x_\xi = \frac{1}{3}$, $y_\xi = \frac{1}{3}$ (nonaligned)			
Grid	Case B		HA scheme		IHA scheme		θ-HA scheme		θ-IHA scheme	
12x12	7.79d-5	–	2.80d-3	–	8.05d-5	–	3.66d-3	–	9.72d-5	–
22x22	2.16d-5	*3.6*	7.98d-4	*3.5*	1.71d-5	*4.7*	1.41d-3	*2.6*	3.50d-5	*2.8*
42x42	5.93d-6	*3.6*	2.09d-4	*3.8*	4.30d-6	*4.0*	2.90d-4	*4.9*	9.48d-6	*3.7*
82x82	1.58d-6	*3.8*	5.34d-5	*3.9*	1.12d-6	*3.8*	8.12d-5	*3.6*	2.66d-6	*3.6*
162x162	4.09d-7	*3.9*	1.35d-5	*4.0*	2.87d-7	*3.9*	1.92d-5	*4.2*	6.82d-7	*3.9*

Table 4.7

2-D problem with $u^{ex} = \frac{1}{k}\sin(\frac{\pi x}{2})(x - x_\xi)(y - y_\xi)(1 + x^2 + y^2)$ *and* $k = \{10, 10^{-1}, 10^{3}, 1\}$ *in four subregions with interfaces at* $x = x_\xi$ *and* $y = y_\xi$*; the relative* C*-norms of the error.*

	$k(x,y) \equiv 1$		$x_\xi = \frac{1}{2}$, $y_\xi = \frac{1}{2}$ (aligned)				$x_\xi = \frac{1}{3}$, $y_\xi = \frac{1}{3}$ (nonaligned)			
Grid	Case B		HA scheme		IHA scheme		θ-HA scheme		θ-IHA scheme	
12x12	2.09d-4	–	2.32d-3	–	2.32d-4	–	8.47d-4	–	2.84d-4	–
22x22	5.74d-5	*3.6*	7.72d-4	*3.0*	6.88d-5	*3.4*	2.14d-4	*4.0*	6.20d-5	*4.6*
42x42	1.75d-5	*3.7*	2.33d-4	*3.3*	1.92d-5	*3.6*	8.98d-5	*2.4*	1.10d-5	*5.6*
82x82	4.11d-6	*3.8*	6.50d-5	*3.6*	5.09d-6	*3.8*	2.08d-5	*4.3*	4.30d-6	*2.6*
162x162	1.06d-6	*3.9*	1.74d-5	*3.7*	1.31d-6	*3.9*	6.60d-6	*3.2*	6.40d-7	*6.7*

Table 4.8

2-D problem with $u^{ex} = \frac{1}{k}\sin(\frac{\pi x}{2})(x - x_\xi)(y - y_\xi)(1 + x^2 + y^2)$ *and* $k = \{10, 10^{-1}, 10^{3}, 1\}$ *in four subregions with interfaces at* $x = x_\xi$ *and* $y = y_\xi$*; the relative* L_2*-norm of the error.*

	$k(x,y) \equiv 1$		$x_\xi = \frac{1}{2}$, $y_\xi = \frac{1}{2}$ (aligned)				$x_\xi = \frac{1}{3}$, $y_\xi = \frac{1}{3}$ (nonaligned)			
Grid	Case B		HA scheme		IHA scheme		θ-HA scheme		θ-IHA scheme	
12x12	7.79d-5	–	4.11d-4	–	6.18d-5	–	1.70d-4	–	6.37d-5	–
22x22	2.16d-5	*3.6*	1.20d-4	*3.4*	1.65d-5	*3.8*	4.30d-5	*4.0*	1.99d-5	*3.2*
42x42	5.93d-6	*3.6*	3.31d-5	*3.6*	4.75d-6	*3.5*	1.14d-5	*3.8*	3.32d-6	*6.0*
82x82	1.58d-6	*3.8*	8.34d-6	*4.0*	1.26d-6	*3.8*	2.84d-6	*4.0*	1.25d-6	*2.7*
162x162	4.09d-7	*3.9*	2.11d-6	*4.0*	3.31d-7	*3.8*	7.24d-7	*3.9*	2.10d-7	*6.0*

Let us discuss the results presented in Tables 4.5–4.8. The scheme with harmonic averaging of $k(x)$ is $O(h^2)$-accurate. The new scheme also converges with second-order, but the constant in front of the convergence factor is two orders of magnitude smaller for the example with large jumps of the coefficients. This means that in practical computations the new scheme allows computations on significantly coarser grids in comparison with known schemes. The accuracy of the new scheme is almost uniform with respect to the size of the jump discontinuity, as one can observe from the tables, and it is comparable with the accuracy of computing the Poisson equation with a constant coefficient (denoted as Case B in the tables). An interesting fact is that the IHA scheme preserves this behavior even in the case when interfaces are not aligned with the finite volume boundaries. At the same time, the accuracy of the HA scheme depends on the jump discontinuity. Note that the larger the jump of the coefficient, the better the advantages of the IHA scheme are seen.

The results in the nonaligned case need special discussion. It was observed in the experiments that the θ-schemes converge uniformly (with respect to refinement of the grid) with second-order only if the grid is refined in a such a way that θ remains constant. In our experiments θ varies from one grid to the next, and this is the reason for the nonmonotone values obtained for the ratios of the norms of the error on consecutive grids. A possible explanation of such a behavior is that the reminder term is different for the cases $0 \leq \theta \leq 0.5$ and $0.5 \leq \theta \leq 1$. This phenomenon needs further detailed investigations.

As a second 2-D example, an interface problem with the right-hand side identically equal to 1 is considered. Dirichlet boundary conditions on the east and west sides and zero Neumann boundary conditions on the north and south sides are prescribed. The computed problem is also known as a thin lenses problem: the diffusion coefficient is very small within two thin lenses and is equal to 1 elsewhere. In our computations the lenses are $\{0.4 < x_1 < 0.9;\ 0.2 < x_2 < 0.25\}$ and $\{0.2 < x_1 < 0.8;\ 0.7 < x_2 < 0.75\}$. The diffusion coefficient of the lenses has value 10^{-4}. The analytical solution of the problem is not known. The solution of the problem has singularities around corners of the lenses, and $u \in W_2^{1+\beta}$, where $\beta \sim \frac{1}{3}$ for our examples. (For details see [19].) The computed solutions are presented in Figure 4.1 for the AA scheme, in Figure 4.2 for the HA scheme, and in Figure 4.3 for the IHA scheme. The left plot on any figure presents the solution on a coarse 22×22 grid, while the right plot presents the solution on a fine 162×162 grid. Note that only one layer of grid cells in the x_2 direction is laying inside a lens on the coarse grid.

The maximum values of the computed numerical solutions are presented in Table 4.9. In this case we refine the grid by tripling the number of nodes in any direction so we have nested grids, and we can monitor the value of the numerical solution at a fixed grid point on the plane.

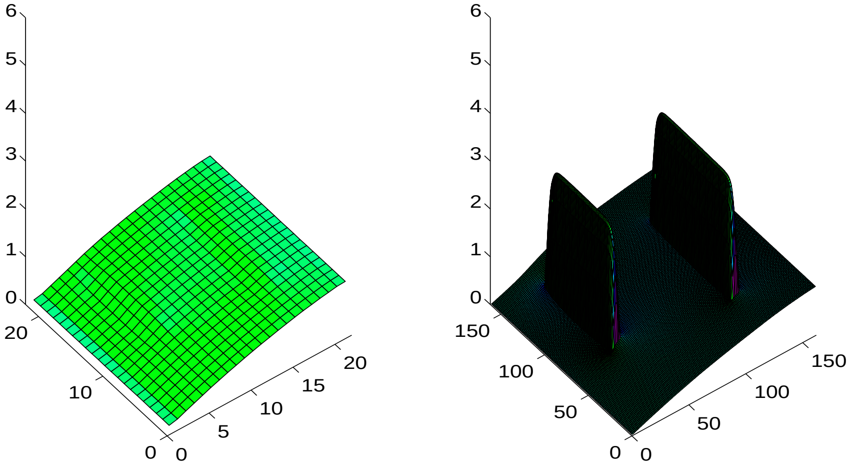


FIG. 4.1. *2-D thin lenses computations with the AA scheme. Left: grid* 22×22. *Right: grid* 162×162.

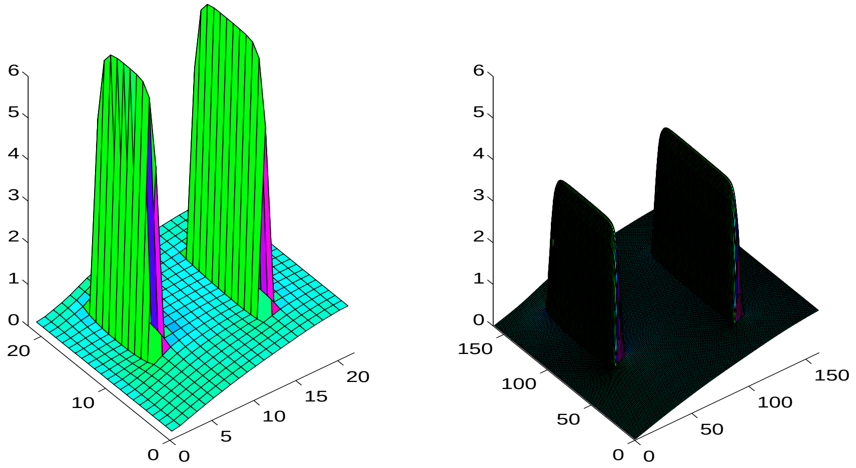


Fig. 4.2. *2-D thin lenses computations with the HA scheme. Left: grid* 22 × 22*. Right: grid* 162 × 162*.*

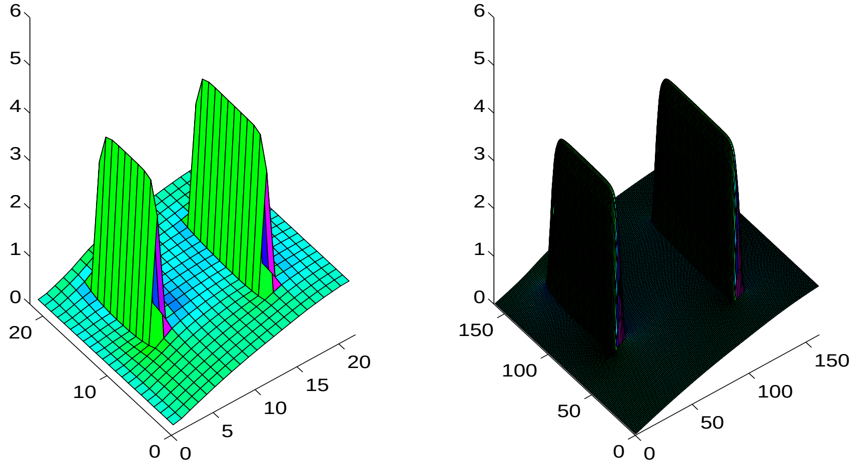


Fig. 4.3. *2-D thin lenses computations with the IHA scheme. Left: grid* 22 × 22*. Right: grid* 162 × 162*.*

Table 4.9
2-D thin lenses problem; the maximum value of the numerical solution.

Grid	AA	HA	IHA
22 × 22	1.0000	7.1186	3.9940
62 × 62	2.2620	4.3444	3.9952
182 × 182	3.3417	4.0356	3.9961
542 × 542	3.7970	4.0012	3.9965

Solutions computed by the IHA scheme are very close to the exact solution even when coarse grids are used. (The first three digits of the maximum value of the solution are correct even on the coarsest grid.) This fact is confirmed by the plots on Figure 4.3, as well as by the data in Table 4.9. At the same time, the HA scheme produces rough approximation to the solution on the coarse grid. The AA scheme is practically unusable for coarse grids and produces inaccurate solutions even on a very fine grid.

It should be noted that, in addition to the thin lenses problem, we have also computed the above example in the cases when the diffusion coefficient takes value 10^{-4} in larger domains (say, in an internal square which cover several grid nodes in each direction, etc.). In all cases the IHA scheme produces much better results than the HA scheme. However, improved harmonic averaging seems to be especially efficient for the thin lenses problems. A possible explanation for this phenomenon is that in the case of the thin lenses problems, the solution behaves in some subregions as a function of one variable; therefore, the scheme reduces to a scheme for a 1-D problem. As we know, for $f(x) \equiv 1$, this scheme is exact.

4.3. Numerical experiments for 3-D problems. We solved a 3-D problem (suggested in [6]) with nonhomogeneous Dirichlet boundary conditions and the known solution $u^{ex} = \frac{1}{k}(x_1 - 0.5)(x_2 - 0.5)(x_3 - 0.5)\sin(\frac{\pi x_1}{2})(1 + x_1^2 + x_2^2 + x_3^2)$, where k is a constant over each of the eight corners of this cube. More precisely, $k = \{10^2, 10^3, 10^7, 10^8, 10^{-2}, 10^{-1}, 10^3, 10^4\}$ in the eight corners, counting from the left to the right, and from the bottom to the top. Interfaces are aligned with the finite volume surfaces in this case, i.e., $\theta = 0.5$. The results from the numerical experiments are presented in Tables 4.10 and 4.11. In all cases we use the discretization of the boundary conditions reported in [8]. Tables 4.10 and 4.11 show that the numerical solution of the interface problem, obtained with the new scheme (3.4), is at least two orders more accurate than the numerical solutions, computed with the other two schemes. Table 4.10 shows that the AA scheme does not have a satisfactory accuracy, especially in the maximum norm. The HA scheme is much better, and the new IHA scheme produces the best results. We note that the solution computed with the new scheme on the coarsest grid with 18^3 points is more accurate than the solution computed by the HA scheme on the finest grid. The same observation can be made when comparing the arithmetic and harmonic averaging schemes. The HA scheme will need approximately 16^{11} nodes to produce the solution with the accuracy achieved by the new scheme on an 18^3-node grid. It can be also observed from Tables 4.10 and 4.11 that the constant of convergence of the new scheme does not depend on the jump of discontinuity in this example, and it is practically equal to the constant from the convergence rate of the scheme for Poisson's equation.

TABLE 4.10
3-D problem with eight subregions; the relative C-norm of the error.

Grid	B	AA	HA	IHA
18^3	7.35d-5	2.05d-1	6.13d-3	8.43d-5
34^3	1.51d-5	1.13d-1	2.10d-3	1.62d-5
66^3	2.29d-6	5.90d-2	6.53d-4	3.71d-6

TABLE 4.11
3-D problem with eight subregions; the relative L_2-norm of the error.

Grid	B	AA	HA	IHA
18^3	8.26d-6	1.01d-2	3.70d-4	9.40d-6
34^3	2.18d-6	4.59d-3	1.03d-4	1.86d-6
66^3	5.93d-7	2.19d-3	2.71d-5	4.31d-7

5. Conclusions. A family of new difference schemes for self-adjoint second-order elliptic equations with discontinuous coefficients is derived via a finite volumes approach. A new scheme, based on improved harmonic averaging of the coefficient, has second-order accuracy under the following assumptions: (1) the diffusion coefficient matrix $K(x)$ is diagonal; (2) the interfaces are planes perpendicular to the coordinate axes; (3) the normal (to the boundaries of a given finite volume) component of the flux is continuously differentiable at the finite volume boundaries. Second-order convergence of the new scheme in the maximum-norm is observed in various numerical experiments for problems in R^n, $n = 1, 2, 3$. The numerical experiments also demonstrate that the new scheme is much more accurate than the known schemes in solving interface problems, especially in the cases of large jumps of the coefficient. The advantages of the new scheme are better seen in solving multidimensional problems with many interfaces and the thin lenses problems.

Acknowledgments. The authors gratefully thank Prof. P. N. Vabishchevich and Prof. P. S. Vassilevski for their valuable suggestions and discussions.

REFERENCES

[1] O. Axelsson and V. Barker, *Finite Element Solution of Boundary Value Problems: Theory and Computations*, Academic Press, Orlando, FL, 1984.

[2] Z.Q. Cai, *On the finite volume element method*, Numer. Math., 58 (1991), pp. 713–735.

[3] Z. Cai, J. Mandel, and S. McCormick, *The finite volume element method for diffusion equations on general triangulations*, SIAM J. Numer. Anal., 28 (1991), pp. 392–402.

[4] S.H. Chou and P.S. Vassilevski, *A general mixed co-volume framework for constructing conservative schemes for elliptic problems*, Math. Comp., 68 (1999), pp. 991–1011.

[5] R.S. Falk and J.E. Osborn, *Remarks on mixed finite element methods for problems with rough coefficients*, Math. Comp., 62 (1994), pp. 1–19.

[6] R.E. Ewing, J. Shen, and P.S. Vassilevski, *Vectorizable preconditioners for mixed finite element solution of second-order elliptic problems*, Internat. J. Comput. Math., 44 (1992), pp. 313–327.

[7] J.M. Hyman and M. Shashkov, *Adjoint operators for the natural discretizations of the divergence, gradient and curl on logically rectangular grids*, Appl. Numer. Math., 25 (1997), pp. 413–442.

[8] O.P. Iliev, *On second order accurate discretization of* 3-*D elliptic problems with discontinuous coefficients and its fast solution with pointwise multi-grid solver*, IMA J. Numer. Anal., to appear.

[9] V.P. Il'in, *High order accurate finite volumes discretizations for Poisson equation*, Siberian Math. J., 37 (1996), pp. 151–169 (in Russian).

[10] H. Jianguo and X. Shitong, *On the finite volume element method for general self-adjoint elliptic problems*, SIAM J. Numer. Anal., 35 (1998), pp. 1762–1774.

[11] V.A. Kondratiev, *Boundary value problems for elliptic equations in domains with conical and angular points*, Trudy Mosk. Matemat. Obschestva, 16 (1967), pp. 209–292 (in Russian).

[12] R. Li, Z. Chen, and W. Wu, *Generalized Difference Method for Differential Equations: Numerical Analysis of Finite Volume Methods*, Marcel Dekker, New York, 2000.

[13] R.J. LeVeque and Z. Li, *Erratum: The immersed interface method for elliptic equations with discontinuous coefficients and singular sources*, SIAM J. Numer. Anal., 32 (1995), p. 1704.

[14] G.I. Marchuk, *Methods of Computational Mathematics*, Nauka, Moscow, 1980.

[15] I.D. Mishev, *Finite volume methods on Voronoi meshes*, Numer. Methods Partial Differential Equations, 14 (1998), pp. 193–212.

[16] A.A. Samarskii, *Theory of Difference Schemes*, Nauka, Moscow, 1977.

[17] A.A. Samarskii and V.B. Andreev, *Difference Methods for Elliptic Equations*, Nauka, Moscow, 1976 (in Russian).

[18] M. Shashkov, *Conservative Finite-Difference Methods on General Grids*, CRC Press, Boca Raton, FL, 1996.

[19] G. Strang and G. Fix, *An Analysis of Finite Element Methods*, Prentice–Hall, Englewood Cliffs, NJ, 1980.

[20] A.N. Tikhonov and A.A. Samarskii, *Homogeneous finite difference schemes*, Zh. Vychisl. Mat. Mat. Fiz., 1 (1961), pp. 5–63 (in Russian).

[21] A.N. Tikhonov and A.A. Samarskii, *Homogeneous finite difference schemes of high accuracy on non-uniform grids*, Zh. Vychisl. Mat. Mat. Fiz., 1 (1961), pp. 425–440 (in Russian).

[22] R.S. Varga, *Functional Analysis and Approximation Theory in Numerical Analysis*, CBMS-NSF Reg. Conf. Ser. in Appl. Math., SIAM, Philadelphia, 1971.

[23] P. Wesseling, *An Introduction to Multigrid Methods*, Wiley, New York, 1991.

SIAM J. SCI. COMPUT.
Vol. 23, No. 4, pp. 1352–1375

A MULTILEVEL ALGORITHM FOR WAVEFRONT REDUCTION*

Y. F. HU† AND J. A. SCOTT‡

Abstract. A multilevel algorithm for reordering sparse symmetric matrices to reduce the wavefront and profile is described. The algorithm is a combinatorial algorithm that uses a maximal independent vertex set for coarsening the adjacency graph of the matrix and an enhanced version of the Sloan algorithm on the coarsest graph. On a range of examples arising from practical applications, the multilevel algorithm is shown to produce orderings that are better than those produced by the Sloan algorithm and are of comparable quality to those obtained using the hybrid Sloan algorithm. Advantages over the hybrid Sloan algorithm are that the multilevel approach requires no spectral information and less CPU time.

Key words. sparse matrices, wavefront and profile reduction, multilevel algorithm, row ordering, frontal method

AMS subject classifications. 65F05, 65F50

PII. S1064827500377733

1. Introduction.

We consider a multilevel algorithm for ordering sparse symmetric matrices for the small wavefront and profile. The resulting ordering may be used to construct a row order for use with the row-by-row frontal method applied to a matrix with a symmetric sparsity pattern (see [27]). Since we are primarily concerned with matrices that are positive definite, we work only with the pattern of the matrix and do not take into account permutations needed for stability. In cases where the matrix is nondefinite, or is symmetric only in its sparsity pattern, the actual factorization may be more expensive and require more storage.

Minimizing the profile of a matrix is known to be an NP-complete problem [23]. A number of heuristic algorithms have been proposed, including the Cuthill–McKee [4], reverse Cuthill–McKee [9, 24], Gibbs–King [12], Gibbs–Poole–Stockmeyer [11], and Sloan [31] algorithms (see also [5, 22, 27]). More recently, spectral orderings based on the Fiedler vector of the Laplacian matrix associated with a matrix have been developed [1, 25, 26]. Kumfert and Pothen [22] propose combining an enhanced version of the second phase of the Sloan algorithm with the spectral ordering. The resulting hybrid Sloan algorithm (hereafter referred to as the *Hybrid algorithm*) has been shown to give significantly better orderings for large problems than either the spectral method or the Sloan method alone. This has been confirmed by Reid and Scott [27], who provide efficient implementations of the Sloan and Hybrid algorithms within the HSL 2000 [15] code MC60.

One reason for the success of the Hybrid algorithm is that the spectral algorithm takes a global view of the graph of the matrix. This global view is fed into the Sloan algorithm as a priority vector, and the Sloan algorithm then performs local refinement.

The spectral algorithm has also been used in the area of graph partitioning [14, 30]. More recently, researchers have found that, for large graphs, a multilevel approach

*Received by the editors September 7, 2000; accepted for publication (in revised form) June 22, 2001; published electronically December 18, 2001.
http://www.siam.org/journals/sisc/23-4/37773.html

†Computational Science and Engineering Department, Daresbury Laboratory, Warrington WA4 4AD, England (yifan@altavista.net). Current address: Wolfram Research Inc., 100 Trade Center Drive, Champaign, IL 61820.

‡Atlas Centre, Rutherford Appleton Laboratory, Oxon OX11 0QX, England (sct@letterbox.rl.ac.uk). The work of this author was funded by EPSRC grant GR/M78502.

[2, 13, 18, 33] can also provide good partitionings, while being much faster, because the calculation of the Fiedler vector of a large matrix is avoided. A number of efficient and high-quality graph partitioning codes based on the multilevel approach have been developed [14, 18, 33]. The success of the multilevel approach in graph partitioning motivated the work reported in this paper.

In this paper, we describe a multilevel Sloan algorithm for the ordering of sparse symmetric matrices. Numerical tests on a range of problems illustrate that the multilevel algorithm yields orderings that are better than those from the Sloan algorithm and are of comparable quality to the Hybrid algorithm. The main advantage of the multilevel approach over Hybrid is that it does not require any spectral information. Moreover, it is less expensive than the Hybrid algorithm.

The paper is organized as follows. In section 2, definitions and terminology are introduced, and the Sloan and Hybrid algorithms are recalled. In section 3, our multilevel approach is presented. Numerical results comparing our method with the Sloan and the Hybrid algorithms are given in section 4. Section 5 summarizes our findings and considers possible future directions for research.

We remark that while the writing of the present paper was in progress, our attention was brought to a report by Boman and Hendrickson [3]. In this report, Boman and Hendrickson propose using a multilevel method for envelope reduction. Their algorithm combines a multilevel approach with a 1-sum local refinement procedure (see section 3.1). Boman and Hendrickson found that the implementation of the Sloan algorithm developed by Kumfert and Pothen [22] often produced better envelopes and required less CPU time than their multilevel approach. This led Boman and Hendrickson to conclude that "Since the Sloan algorithm has recently been shown to be a fast and good algorithm for envelope reduction, we expect that the multilevel algorithm can be improved by replacing our 1-sum local refinement with a modified version of the Sloan algorithm." Although we were originally unaware of it, this paper takes up their challenge of combining the idea of a multilevel approach with the Sloan algorithm.

2. Background.

2.1. Definitions. We first need to introduce some nomenclature and notation. Let $A = \{a_{ij}\}$ be an $n \times n$ symmetric matrix. At the ith step of the factorization of A, row k is said to be *active* if $k \geq i$ and there exists a column index $l \leq i$ such that $a_{kl} \neq 0$. The ith *wavefront* f_i of A is defined to be the number of rows that are active during the ith step of the factorization. The *maximum* and *root-mean-squared* (RMS) wavefronts are, respectively,

$$F(A) = \max_{1 \leq i \leq n} \{f_i\}$$

and

$$\mathrm{rmsf}(A) = \left(\frac{\sum_{i=1}^{n} f_i^2}{n} \right)^{\frac{1}{2}}.$$

The *profile* of A is the total number of entries in the lower triangle when any zero ahead of the first entry in its row is excluded, that is,

$$P(A) = \sum_{i=1}^{n} \max_{a_{ij} \neq 0} \{i + 1 - j\}. \tag{2.1}$$

It is straightforward to show that

$$P(A) = \sum_{i=1}^{n} f_i.$$

The matrix envelope Env(A) is $P(A) - n$. For a frontal solver these statistics are important because

- the memory needed to store the frontal matrix is F^2;
- P is the total storage needed for the factorized matrix;
- the number of floating-point operations when eliminating a variable is proportional to the square of the current wavefront size.

Our goal therefore is to construct an efficient ordering algorithm that reduces the above quantities.

It is often convenient when developing ordering algorithms to treat the matrix A in terms of its adjacency graph. An undirected graph $\mathcal{G}$ is defined to be a pair (V, E), where V is a finite set of *vertices* (or *nodes*) and E is a finite set of *edges* defined as unordered pairs of distinct vertices. In a *weighted* graph, each vertex and edge has a weight associated with it. The adjacency graph $\mathcal{G}(A)$ of the square symmetric matrix A comprises the vertices $V(A) = \{1, 2, \ldots, n\}$ and the edges

$$E(A) = \{(i,j) \mid a_{ij} \neq 0,\ i > j\}.$$

Two vertices i and j are said to be *neighbors* (or to be *adjacent*) if they are connected by an edge. The notation $i \leftrightarrow j$ will be used to show that i and j are neighbors. The *adjacency* set for i is the set of its neighbors, that is,

$$\mathrm{adj}(i) = \{j \mid j \leftrightarrow i,\ i, j \in V\}.$$

The *degree* of $i \in V$ is $\deg(i) = |\mathrm{adj}(i)|$, the number of neighbors. If X is a subset of V, its adjacency set is defined to be

$$\mathrm{adj}(X) = \bigcup_{j \in X} \mathrm{adj}(j) \backslash X.$$

Observe that, given the graph representation of a symmetric matrix, the ith wavefront can be defined as the vertex i plus the set of vertices adjacent to the vertex set $\{1, 2, \ldots, i\}$, that is,

$$f_i = \mathrm{adj}\left(\{1, 2, \ldots, i\}\right) \cup \{i\}.$$

A *path* of length k in $\mathcal{G}$ is an ordered set of distinct vertices $\{v_1, v_2, \ldots, v_k, v_{k+1}\}$ with $v_i \leftrightarrow v_{i+1}$ $(1 \leq i \leq k)$. Two vertices are *connected* if there exists a path between them. A graph $\mathcal{G}$ is connected if each pair of distinct vertices is connected. The *distance*, $\mathrm{dist}(u, v)$, between two vertices u and v in $\mathcal{G}$ is the length of the shortest path connecting them. The *eccentricity* of a vertex u is defined to be

$$\mathrm{ec}(u) = \max\{\mathrm{dist}(u, v) \mid v \in \mathcal{G}\}.$$

The *diameter* of $\mathcal{G}$ is then

$$\delta(\mathcal{G}) = \max\{\mathrm{ec}(u) \mid u \in \mathcal{G}\}.$$

A vertex u is a *peripheral* vertex if its eccentricity is equal to the diameter of the graph, that is, $\mathrm{ec}(u) = \delta(\mathcal{G})$. A *pseudoperipheral* vertex u is defined by the condition that, if v is any vertex for which $\mathrm{dist}(u, v) = \mathrm{ec}(u)$, then $\mathrm{ec}(v) = \mathrm{ec}(u)$. The pair u, v of pseudoperipheral vertices define a *pseudodiameter*.

Throughout our discussion, it is assumed that the matrix A of interest is irreducible so that its adjacency graph $\mathcal{G}(A)$ is connected. Disconnected graphs can be treated by considering each component separately.

2.2. The Sloan algorithm. The Sloan algorithm [31] is widely used for profile and wavefront reduction. The algorithm, which uses the adjacency graph of the matrix, has the following two distinct phases:

1. selection of a start vertex s and an end vertex e,
2. vertex reordering.

The first phase looks for a pseudodiameter of the graph and chooses s and e to be the endpoints of this pseudodiameter. A pseudodiameter may be computed using a modification of the Gibbs–Poole–Stockmeyer algorithm. (See [27] for details of an efficient approach.) In the second phase, the chosen start vertex is numbered first, and a list of vertices that are eligible to be numbered next is formed. At each stage of the numbering, the list of eligible vertices comprises the neighbors of the vertices that have already been renumbered and their neighbors. The next vertex to be numbered is selected from the list of eligible vertices by means of a priority function. The priority of vertex i is given by $P(i)$, where

$$P(i) = -W_1 \mathrm{inc}(i) + W_2 \mathrm{dist}(i, e) \tag{2.2}$$

and (W_1, W_2) are positive weights. The first term (the "local" term), $\mathrm{inc}(i)$, is the amount by which the wavefront will increase if vertex i is ordered next. The second term (the "global" term), $\mathrm{dist}(i, e)$, is the distance between i and the end vertex e. Thus, a balance is maintained between the aim of keeping the wavefront small and bringing in vertices that have been left behind (far away from e). A vertex has a high priority if it causes either no increase or only a small increase to the current front size and is at a large distance from the end vertex e. The best choice for the weights (W_1, W_2) is problem dependent, but Sloan suggested that weights $(2, 1)$ usually gave satisfactory results for his test problems. Once an ordering for the vertices of the weighted graph has been obtained, an ordering for A can be constructed.

Duff, Reid, and Scott [5] extended Sloan's algorithm to vertex-weighted graphs obtained from finite element meshes and used the resulting orderings to generate element assembly orderings for the frontal method. The vertex-weighted graph is derived from the unweighted graph by "condensing" vertices to form supervertices. Vertices i and j are condensed into a supervertex if

$$i \cup \mathrm{adj}(i) = j \cup \mathrm{adj}(j). \tag{2.3}$$

The weight of a supervertex is the number of unweighted vertices it represents. The use of condensing can sometimes reduce the size of the graph considerably, thus reducing the time required for reordering.

The implementation of Sloan's algorithm was further enhanced by Kumfert and Pothen [22] in a number of ways.

- Sloan [32] used a simple search method to manage the priority queue but noted that, for large problems, using a binary heap would be the method of choice. Kumfert and Pothen implemented the use of a binary heap and were able to achieve a considerable efficiency gain. They also analyzed the time complexity of their efficient implementation.
- They applied the Sloan algorithm to the vertex-weighted graph so that it mimics what the algorithm would do on the corresponding unweighted graph. This resulted in smaller wavefront sizes.
- They divided their test problems into two classes and showed that, by using different weights for each class, for some problems the wavefront sizes obtained

by the Sloan algorithm could be substantially reduced. However, they were not able to predict a priori to which class a given problem belongs.

Most recently, Reid and Scott [27] have provided a Fortran implementation of Sloan's algorithm in HSL 2000 [15] as routine MC60. This code uses and extends the ideas of Kumfert and Pothen. MC60 optionally uses the vertex-weighted graph. For efficiency on problems of all sizes, when managing the priority queue, the code starts by using the simple search used by Sloan and switches to using a binary heap if the number of eligible vertices exceeds a given threshold. By default, MC60 tries both the pairs of weights $(2,1)$ and $(16,1)$. Alternatively, the user can supply the weights. The code also allows the user to provide a global priority vector to be used in place of the distance $\text{dist}(i,e)$ in the priority function. This enables MC60 to be used to implement the Hybrid algorithm of Kumfert and Pothen [22], which we discuss in the next section.

Throughout the rest of this paper, when referring to the Sloan algorithm, we mean the enhanced version of the algorithm as implemented within MC60.

2.3. The Hybrid algorithm. The first term in (2.2) affects the priority function in a local way by giving higher priority to vertices that will result in a small (or negative) increase to the current wavefront. This is done in a greedy fashion without consideration of the long-term effect. The second term acts in a more global manner, ensuring vertices lying far away from the end vertex are not left behind. The second phase of the Sloan algorithm can therefore be viewed as an algorithm that refines the ordering implied by the distance function $\text{dist}(i,e)$.

The distance function in (2.2) can be replaced by other orderings that provide a global view. In particular, Kumfert and Pothen [22] proposed using a spectral ordering. The spectral algorithm associates a Laplacian matrix $L = \{l_{ij}\}$ with the symmetric matrix A as follows:

$$l_{ij} = \begin{cases} -1 & \text{if } i \neq j \text{ and } i \leftrightarrow j, \\ \deg(i) & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases} \tag{2.4}$$

An eigenvector corresponding to the smallest positive eigenvalue of the Laplacian matrix is called a *Fiedler vector* [7, 8]. The spectral algorithm orders the vertices of $\mathcal{G}(A)$ by sorting the components of the Fiedler vector into monotonic order. The same permutation is applied to the original matrix to obtain the spectral ordering. This approach has been found to produce small profiles and wavefronts [1], although it is much more computationally expensive. (This is illustrated in [22].) An analysis of the spectral method for envelope reduction has been presented by George and Pothen [10].

The spectral algorithm has also been used in the context of graph partitioning [14], where it has been found that results can be improved by incorporating a local refinement step. This refinement performs local optimizations and smoothes out local oscillations that may be present. In the context of wavefront reduction, the Hybrid algorithm of Kumfert and Pothen [22] combines the spectral algorithm with a modified version of the second phase of the Sloan algorithm. It appears that it is this combination of global and local ordering algorithms that accounts for the good performance of the Hybrid algorithm, particularly for very large problems.

The Hybrid algorithm, as presented in [27], chooses as the start vertex s the first vertex in the spectral ordering and replaces (2.2) with the priority function

$$P(i) = -W_1 \text{inc}(i) - W_2\, \nu\, p(i). \tag{2.5}$$

Here ν is a normalizing factor and $p(i)$ is the position of vertex i in the spectral ordering, also referred to as its global priority value. ν is chosen so that the factor for W_2 varies up to $\text{dist}(s, e)$, as in (2.2). On the basis of their numerical experimentation, Reid and Scott [27] propose the pairs of weights $(1, 2)$ and $(16, 1)$. Note that $(1, 2)$ is recommended instead of the pair $(2, 1)$ used by default in their implementation of the Sloan algorithm. Reid and Scott argue that the global priority based on the spectral ordering has been found to be better than that obtained from a pseudodiameter, justifying a larger value for W_2 in this case. Numerical experiments [22, 27] have shown that, for large problems, in terms of the quality of the orderings produced, the Hybrid method can significantly outperform the Sloan algorithm. The Hybrid method does, however, require significantly more CPU time because it is more expensive to compute the Fiedler vector than it is to find a pseudodiameter for A. This is illustrated in section 4.

Through the use of (2.5), the modified second phase of Sloan's algorithm locally refines the spectral ordering. We therefore refer to this phase as Sloan refinement and denote by *SloanRefine*$(\mathcal{G}, p)$ the algorithm that takes the graph $\mathcal{G}$ together with a global priority vector p and uses (2.5) to return a refined ordering for $\mathcal{G}$.

3. The multilevel ordering algorithm.

The matrix ordering algorithm proposed in this paper is based on a multilevel approach. Given the adjacency graph $\mathcal{G}(A)$, a series of graphs is generated, each coarser than the preceding one. The coarsest graph is then ordered. This ordering is recursively prolonged to the next finer graph, local refinement is performed at each level, and the final ordering on the finest graph gives an ordering for A.

3.1. The multilevel approach.

In the context of graph partitioning, the multilevel approach generates a series of coarser and coarser graphs [2, 13, 19, 33]. The aim is for each successive graph to encapsulate the information needed to partition its "parent," while containing fewer vertices and edges. The coarsening continues until a graph with only a small number of vertices is reached. This can be partitioned cheaply. The partitions on the coarse graphs are recursively prolonged (usually by injection) to the finer graphs with further refinement at each level.

One of the first uses of a multilevel approach for the partitioning of undirected graphs was reported by Barnard and Simon [2]. Motivated by the need to reduce the time for computing the Fiedler vector, Barnard and Simon combined a multilevel approach with a spectral bisection algorithm. It was soon realized [13, 20, 33] that the multilevel approach can be used to advantage with a good local optimizer. In graph partitioning, the Kernighan–Lin algorithm [21] is used and, combined with the multilevel approach, has proved very successful at rapidly computing high quality partitions.

Boman and Hendrickson [3] propose adopting a multilevel approach for profile and wavefront reduction. They introduce a *weighted* 1-*sum* metric

$$\sigma_1(A) = \sum_{i=1}^{n} \sum_{j \leftrightarrow i,\ j<i} w_{ij}(i-j),$$

where w_{ij} are edge weights, and aim to minimize $\sigma_1(A)$. The edge weights are all one on the finest graph; on the coarser graphs, edge weights are assigned as edges are collapsed. (See [3] for details.) Although Boman and Hendrickson's objective is to minimize the matrix envelope of A, they report finding it more efficient and effective to work with the 1-sum. Boman and Hendrickson propose combining a multilevel approach with a refinement algorithm that is similar to the Kernighan–Lin algorithm and is based on swapping consecutive vertices. The gain in swapping each such pair of vertices k and $k+1$ is calculated initially and then updated during the refinement. The gain from a swap is measured using the weighted 1-sum. Boman and Hendrickson compare their approach with an implementation of the Sloan algorithm that incorporates the enhancements of Kumfert and Pothen [22] (referred to by Boman and Hendrickson as the fast Sloan). Boman and Hendrickson conclude that "The fast Sloan algorithm operates in the same performance range and often produces better envelopes in less time than the multilevel algorithm, but not always." Since, in turn, the Hybrid algorithm significantly outperforms Sloan for large problems, the multilevel algorithm of Boman and Hendrickson is not competitive with the Hybrid algorithm in terms of ordering quality.

As recognized by Boman and Hendrickson in their concluding remarks, because the reordering phase of the Sloan algorithm provides a good local refinement algorithm, it is of interest to try and use it directly with the multilevel approach. This combining of Sloan with the multilevel approach forms the basis of our multilevel wavefront reduction algorithm. The algorithm has three distinct phases: coarsening, coarsest graph ordering, and, finally, prolongation and refinement. We discuss each of these phases and then, in section 3.5, we outline our multilevel Sloan algorithm.

3.2. The coarsening phase. There are a number of ways to coarsen an undirected graph. The most popular method in graph partitioning is based on edge collapsing [13, 20], in which pairs of adjacent vertices are selected and each pair is coalesced into one new vertex. Because of its success for graph partitioning, coarsening using edge collapsing was the first strategy we employed when developing our multilevel wavefront reduction algorithm. However, we found that, while we were able to improve on the wavefronts obtained using the Sloan algorithm, the results were of a poorer quality than those given by the Hybrid algorithm. Full details and comparisons are given in the report [17]. This led us to consider alternative coarsening strategies.

In [2], a *maximal independent vertex set* of a graph is chosen as the vertices for the coarse graph. An independent set of vertices is a subset of the vertices such that no two vertices in the subset are connected by an edge in the graph. An independent set is maximal if the addition of an extra vertex always destroys the independence. An algorithm for constructing a maximal independent set is discussed below. Edges of the coarse graph are formed through a process based on the Galerkin product (see section 3.5 for details), which effectively links two vertices in the maximal independent vertex set by an edge if their distance apart is no greater than three. Figure 3.1 illustrates a graph $\mathcal{G}$ with 788 vertices, together with two levels of coarsening using this method, giving graphs with 332 and 94 vertices, respectively.

The coarsening process is applied recursively until one of the following is achieved:

- The number of levels exceeds a preset limit.
- The number of vertices in the coarsest graph is less than a preset number (chosen to be 100 in this study, but see section 4.4 for further discussions).
- The ratio of the number of vertices in two successive graphs exceeds a preset constant (0.8 in this study).

The last condition is necessary to avoid a slow reduction between fine and coarse graph sizes that can lead to a high algorithmic complexity for the multilevel algorithm.

3.2.1. Maximal independent vertex set algorithm. When used for the multilevel spectral algorithm, a maximal independent set is chosen in a greedy fashion

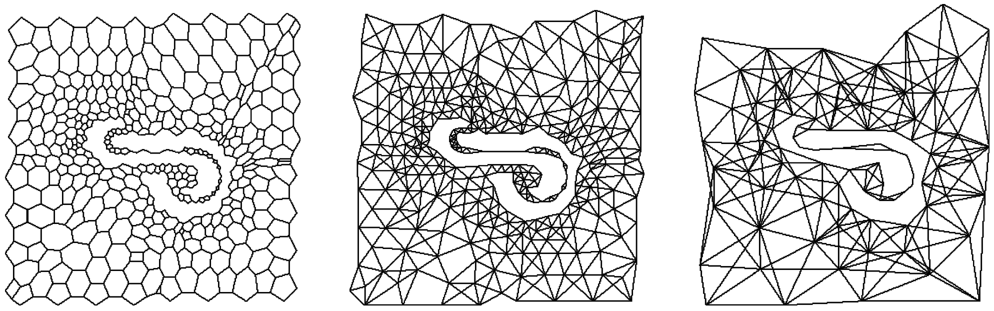


FIG. 3.1. *Graph coarsening based on the maximal independent vertex set: the original graph* $\mathcal{G}$ *with* 788 *vertices (left);* $\mathcal{G}_1$ *with* 332 *vertices (center);* $\mathcal{G}_2$ *with* 94 *vertices (right).*

by picking unmatched vertices at random and, when a vertex is picked, masking its neighboring vertices as matched [1, 2]. Hereafter this method of coarsening is referred to as the simple greedy approach.

We adopt a more sophisticated approach that yields comparable quality of orderings but requires less CPU time. (See section 4.3 for a comparison.) Our algorithm is based on that of Ruge and Stüben [28], which has been used successfully in the field of algebraic multigrid. This algorithm was designed for unsymmetric matrices; we have modified it for symmetric matrices. At each step, each vertex in V lies in one of three sets: it is either uncolored (V_U), it is in the maximal independent set (V_C), or it is not a candidate for the maximal independent set (V_F). Each vertex has a gain value associated with it, indicating the preference for this vertex to belong to V_C. Initially, each vertex i is uncolored (lies in V_U) and is assigned a gain value gain(i) equal to its degree. The gains are held in a priority queue. At each step, an uncolored vertex with the highest gain is removed from the queue and is moved into V_C. Its neighbors are then moved into V_F. They are also removed from the queue. For each such new vertex in V_F, the gain values of its uncolored neighbors are increased by one. The procedure is repeated until the queue is empty (that is, until all the vertices belong to either V_C or V_F).

In this algorithm, the gain of an uncolored vertex is always equal to the number of neighbors in V_U plus twice the number of neighbors in V_F. An uncolored vertex with a large number of neighbors in V_F is therefore more likely to be moved into V_C. This ensures that V_F vertices are well "covered" by V_C vertices, yielding a more uniform distribution of V_C vertices and allowing a more aggressive coarsening compared with the simple greedy approach. Our maximal independent vertex set algorithm is outlined below.

MAXIMAL INDEPENDENT VERTEX SET ALGORITHM.

```
• initialization:
    – $V_C = \emptyset$
    – $V_U = V$
    – for each vertex $i \in V$
        $\mathrm{gain}(i) = \deg(i)$
• do while $V_U \neq \emptyset$
    – $i_{\max} : \mathrm{gain}(i_{\max}) = \max_{j \in V_U}(\mathrm{gain}(j))$
    – $V_U = V_U \backslash \{i_{\max}\}$
    – $V_C = V_C \cup \{i_{\max}\}$
    – for each vertex $j \in \mathrm{adj}(i_{\max}) \cap V_U$
    $V_U = V_U \backslash \{j\}$
    for each vertex $k \in \text{adj}(j) \cap V_U$
      $\text{gain}(k) = \text{gain}(k) + 1$
```

3.3. The coarsest graph ordering. Because the coarsest graph has a small number of vertices and edges, it can be reordered quickly using any standard profile reduction algorithm. We have used both the Sloan and the Hybrid algorithms and present results for both approaches (using the MC60 implementation of these algorithms) in section 4.

3.4. The prolongation and refinement phase. During the prolongation phase, the vertices of the fine graph are given global priority values by mapping the coarse graph ordering onto the fine graph. This mapping can be represented by a prolongation matrix P. If the coarse and fine graphs have n_c and n_f vertices, respectively, the prolongation matrix is of order $n_f \times n_c$.

When coarsening is based on a maximal independent vertex set, the coarse graph vertices comprise the maximal independent set of the fine graph. The global priority value of a vertex in the fine graph that belongs to the maximal independent set is defined as the position of this vertex in the coarse graph ordering. The global priority value of a fine graph vertex not in the maximal independent set is calculated by averaging the global priority values of its neighbors that belong to the maximal independent set. (By definition there is at least one such neighbor.) For each coarse graph vertex j, let fine(j) be the index of this vertex in the fine graph. For each fine graph vertex i, define mdeg(i) to be the number of neighboring fine graph vertices that belong to the maximal independent set. The prolongation matrix $P = \{p_{ij}\}$ has entries

$$p_{ij} = \begin{cases} 1 & \text{if } i = \text{fine}(j), \\ \frac{1}{\text{mdeg}(i)} & \text{if } i \leftrightarrow \text{fine}(j),\ i \neq \text{fine}(j), \\ 0 & \text{otherwise}. \end{cases} \tag{3.1}$$

The global priority values are refined using the second phase of the enhanced Sloan algorithm (that is, by using the priority function (2.5)) to give the final ordering for the fine graph.

3.5. The multilevel algorithm. We can now formulate our multilevel wavefront reduction algorithm. For this it is convenient to introduce some further notation. The subscripts f and c are used to represent fine and coarse graph quantities, respectively. For example, $\mathcal{G}_f$ denotes the fine graph with n_f vertices and $\mathcal{G}_c$ is the graph with n_c vertices obtained after coarsening ($n_c < n_f$). We will associate with $\mathcal{G}_f$ an $n_f \times n_f$ matrix A_f which has zero diagonal entries and nonzero off-diagonal entries a_{ij} if and only if vertices i and j are adjacent in $\mathcal{G}_f$. A_c is defined analogously.

If P denotes an $n_f \times n_c$ prolongation matrix, the coarse graph may be expressed as the Galerkin product

$$A_c \leftarrow P^T\ A_f\ P.$$

This expression means that the matrix product $P^T A_f P$ is computed and the matrix A_c is obtained by setting the diagonal entries of the resulting matrix to zero.

The *global priority vector* p of a graph is a vector with entries $p(i)$, where $p(i)$ is the global priority value of vertex i. This vector indicates the preferred ordering of the vertices. For the Hybrid algorithm, the global priority vector is obtained by ordering the vertices based on the values of the Fiedler vector. Note that the global priority vector need *not* be a permutation vector. We let $\mathit{CoarsestOrder}(\mathcal{G})$ be an algorithm that returns an ordering for the coarsest graph $\mathcal{G}$ and recall that $\mathit{SloanRefine}(\mathcal{G}, p^0)$ denotes the algorithm that takes the graph $\mathcal{G}$, and a global priority vector p^0, and returns a refined ordering for $\mathcal{G}$ using (2.5) with $p = p^0$.

With this notation, if *MinSize* is the preset number of vertices beyond which there is no further coarsening, our multilevel wavefront reduction algorithm can be formally presented as follows. The starting point is the fine graph $\mathcal{G}_f$ and associated matrix A_f.

```
FUNCTION MultilevelOrder($\mathcal{G}_f$).
  • If $n_f$ < MinSize, then
      – $p_f$ = CoarsestOrder($\mathcal{G}_f$)
      – return $p_f$
  • The coarsening phase:
      – set up the $n_f \times n_c$ prolongation matrix $P$
      – $A_c \leftarrow P^T A_f P$
      – $p_c$ = MultilevelOrder($\mathcal{G}_c$)
  • The prolongation and refinement phase:
      – $p_f^0 = P p_c$
      – $p_f$ = SloanRefine($\mathcal{G}_f, p_f^0$)
      – return $p_f$
```

Figure 3.2 illustrates the multilevel algorithm applied to the test problem bcsstk11 (see the appendix). Notice that on the coarsest level, the lower right-hand part of the matrix after reordering (bottom right) has no nonzero entries. This is because the graph corresponding to the bcsstk11 matrix has nine components, eight of which are small, and coarsening gives eight isolated vertices on the coarsest level. Since diagonal elements are not displayed, after reordering there is an 8 × 8 null matrix in the lower right-hand part of the coarsest matrix.

Figure 3.3 presents a simple example of the two-level ordering algorithm using a maximal independent vertex set. The vertices $v3$, $v6$ are chosen to form the maximal independent set. Vertex $v1$ has one neighbor in the maximal independent set; therefore $\mathrm{mdeg}(v1) = 1$. Similarly, $\mathrm{mdeg}(v4) = 1$, and $\mathrm{mdeg}(v2) = \mathrm{mdeg}(v5) = 2$. Thus from (3.1), the prolongation matrix is

$$P = \begin{pmatrix} 1 & 0 \\ \frac{1}{2} & \frac{1}{2} \\ 1 & 0 \\ 1 & 0 \\ \frac{1}{2} & \frac{1}{2} \\ 0 & 1 \end{pmatrix}.$$

It follows that the Galerkin product is

$$P^T A_f P = \frac{1}{4} \begin{pmatrix} 2 & 0 \\ 1 & 1 \\ 2 & 0 \\ 2 & 0 \\ 1 & 1 \\ 0 & 2 \end{pmatrix}^T \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 2 & 0 \\ 1 & 1 \\ 2 & 0 \\ 2 & 0 \\ 1 & 1 \\ 0 & 2 \end{pmatrix} = \begin{pmatrix} 8.5 & 3.5 \\ 3.5 & 2.5 \end{pmatrix}.$$

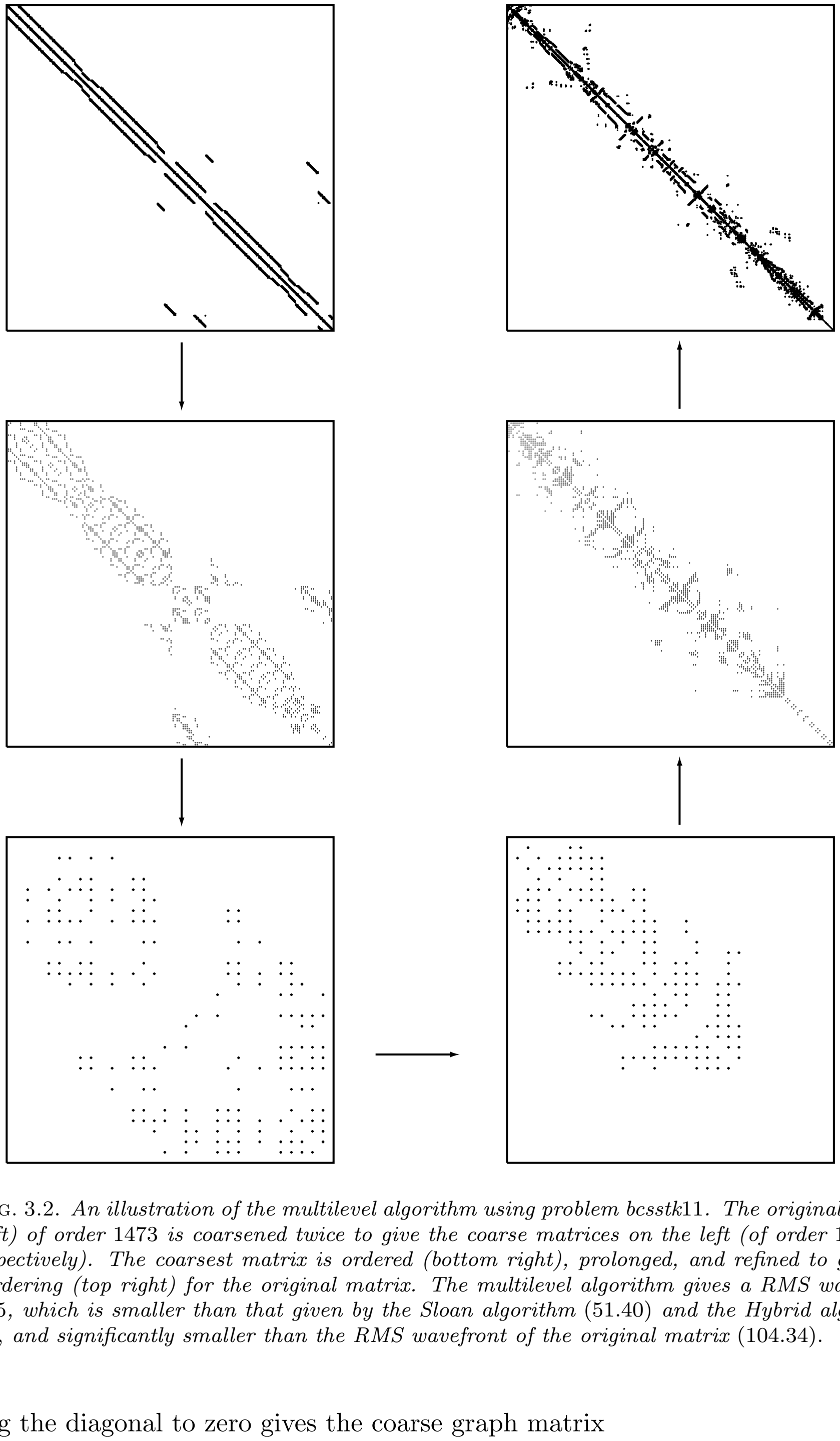


FIG. 3.2. *An illustration of the multilevel algorithm using problem bcsstk*11*. The original matrix (top left) of order* 1473 *is coarsened twice to give the coarse matrices on the left (of order* 162 *and* 30*, respectively). The coarsest matrix is ordered (bottom right), prolonged, and refined to give the final ordering (top right) for the original matrix. The multilevel algorithm gives a RMS wavefront of* 46.55*, which is smaller than that given by the Sloan algorithm* (51.40) *and the Hybrid algorithm* (47.76)*, and significantly smaller than the RMS wavefront of the original matrix* (104.34).

Setting the diagonal to zero gives the coarse graph matrix

$$A_c = \begin{pmatrix} 0 & 3.5 \\ 3.5 & 0 \end{pmatrix},$$

which corresponds to the coarse graph in Figure 3.3 (middle).

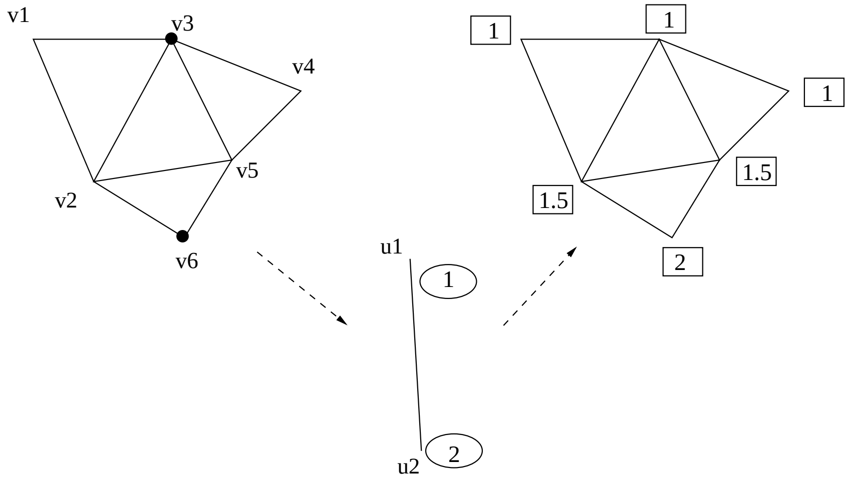


FIG. 3.3. *A graph (top left) is coarsened using the maximal independent vertex set (shaded and circled) to give the coarse graph (middle). The coarse graph is ordered, and this ordering is prolonged to give the priority vector for the fine graph (top right). Numbers in ellipses are the coarse graph ordering, and numbers in squares are global priority values.*

Assuming the coarse graph is ordered as $p_c(u1) = 1$, $p_c(u2) = 2$, the global priority vector for the fine graph is

$$p_f^0 = Pp_c = \frac{1}{2}\begin{pmatrix} 2 & 0 \\ 1 & 1 \\ 2 & 0 \\ 2 & 0 \\ 1 & 1 \\ 0 & 2 \end{pmatrix}\begin{pmatrix} 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 1 \\ 1.5 \\ 1 \\ 1 \\ 1.5 \\ 2 \end{pmatrix}. \tag{3.2}$$

This is shown on the right-hand side of Figure 3.3.

4. Numerical results. In this section, our multilevel approach is compared with the Sloan and Hybrid algorithms on a large set of test problems. All the codes used to obtain the reported results are written in Fortran. The experiments are performed on a COMPAQ computer with a 300 MHz Alpha EV5 processor, using the DIGITAL Fortran 90 V5.2 compiler.

The MC60 code of Reid and Scott [27] is used in the experiments for the Sloan algorithm, for ordering the coarsest graph, and for subsequent refinement. The spectral ordering needed for the Hybrid algorithm is computed using a multilevel Fiedler vector code written by the first author. The algorithm implemented by this Fiedler code is similar to that described in [2]. Coarsening based on heavy edge collapsing is used, and, on the coarsest graph, vertex weights and edge weights are not used in the computation of the Fiedler vector.

The coarsest graph ordering algorithm *CoarsestOrder* is taken to be either the Sloan or the Hybrid algorithm. We denote by *Sloan*(MIV, K) (and *Hybrid*(MIV, K)) the multilevel algorithm that uses the Sloan (respectively, Hybrid) algorithm on the coarsest graph together with the *MIV* (maximal independent vertex set) coarsening scheme on up to K levels. Thus *Sloan*$(MIV, 1)$ is the Sloan algorithm and

Hybrid$(MIV, 1)$ the Hybrid algorithm. Similarly, *Sloan*$(MIV, 3)$ is the multilevel algorithm, with maximal independent vertex set coarsening and a maximum of three levels, with the Sloan algorithm used on the coarsest graph.

Our suite of 101 test problems is listed in alphabetical order in the appendix. The problems are all symmetric and range in order from 66 (dwt66) to 224,617 (Halfb). We have included all the nontrivial symmetric problems of order at least 1000 from MatrixMarket (http://math.nist.gov/MatrixMarket). In addition, we have the Everstine test set [6], which, although small by today's standards, is widely used as a test set for profile and wavefront reduction algorithms. The Everstine problems are available from the Harwell–Boeing Sparse Matrix Collection (http://www.cse.clrc.ac.uk/Activity/SparseMatrices). We also have all the test problems used by Kumfert and Pothen [22], together with some additional finite element problems supplied by Christian Damhaug of Det Norske Veritas, Norway. Included in the appendix are the initial RMS wavefronts (rmsf) for each matrix and the ratio, ρ, between the RMS wavefronts before and after reordering with the Hybrid algorithm. In general, the Hybrid algorithm substantially improves the ordering (although there are a small number of exceptions, notably problems bscctk13, bcsstk17, and bcsstm13).

4.1. Multilevel Sloan algorithm. Figure 4.1 compares the RMS wavefront for the Sloan and the Hybrid algorithms with the multilevel algorithm *Sloan*(MIV, K).

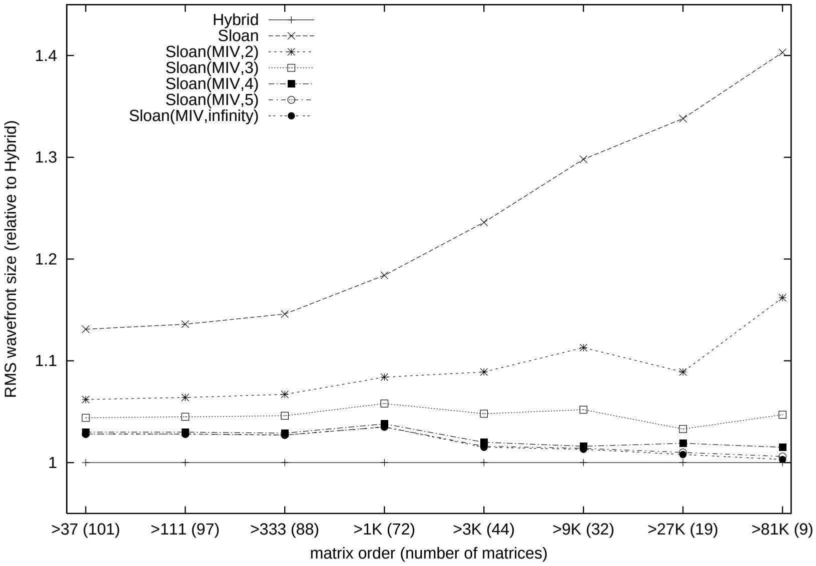


FIG. 4.1. *A comparison of the RMS wavefronts for the Sloan, the Hybrid, and the Sloan*(MIV, K) *algorithms.*

In this and subsequent figures, comparisons are given with respect to the Hybrid algorithm so that the RMS wavefront for each algorithm is divided by the corresponding RMS wavefront for the Hybrid algorithm and geometrically averaged over the test cases to give a relative score for the algorithm. The smaller the score, the better the algorithm. With this metric, the Hybrid algorithm always has a score of one. To show the effect of matrix order, the scores for each algorithm for matrices of order greater than $37 \times 3^k (1 \leq k \leq 8)$ are plotted separately in the figure with the number of matrices over the threshold printed in brackets. A log scale is used for the x-axis (matrix order).

A number of interesting features can be observed. The first observation is that, relative to the Hybrid algorithm, the RMS wavefront given by the Sloan algorithm deteriorates as the order of the matrix increases. Overall, the RMS wavefront for the Sloan algorithm is about 13% greater than for the Hybrid algorithm, while for the largest 10 matrices, it is about 40% more. This deterioration further confirms the earlier findings reported in [22, 27].

The second observation is that, as the number of levels increases, the multilevel orderings improve. The multilevel algorithm without a preset maximum number of levels, *Sloan*(MIV, ∞), produces orderings of comparable quality (within 3.5%) to the Hybrid algorithm and, in terms of CPU time (Figure 4.2), is substantially faster in our experiments, requiring about half the time of the Hybrid algorithm. Since *Sloan*(MIV, ∞) is generally no more expensive in terms of CPU time than *Sloan*(MIV, K) with $K > 2$, and it produces the smallest RMS wavefronts, we recommend not imposing a maximum number of levels on the multilevel algorithm. Thus we have a combinatorial algorithm for wavefront reduction that performs as well as the Hybrid algorithm in less time.

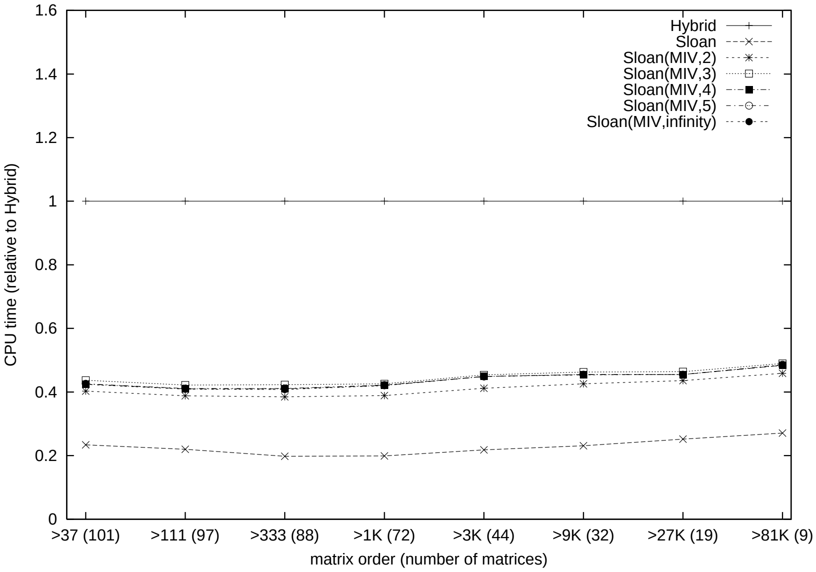


FIG. 4.2. *A comparison of the CPU times for the Sloan, the Hybrid, and the Sloan(MIV, K) algorithms.*

The above results were obtained using the modified version of the Ruge and Stüben [28] algorithm for selecting a maximal independent set. As discussed in section 3.2.1, a simple greedy algorithm can be used instead. The Ruge and Stüben algorithm selects coarse vertices by maximizing the number of neighbors in V_F and V_U. In general, this gives a more aggressive coarsening and fewer dense matrices on the coarse graphs. The result is that a multilevel algorithm based on the Ruge and Stüben approach requires less CPU time than the simple greedy algorithm. This is illustrated in Figure 4.3, where *Sloan*(MIV, ∞) is compared with *Sloan*$(MIVG, \infty)$, with the latter denoting the multilevel algorithm using the simple greedy approach for selecting the maximal independent set. *Sloan*(MIV, ∞) clearly takes less CPU time while yielding orderings of comparable quality.

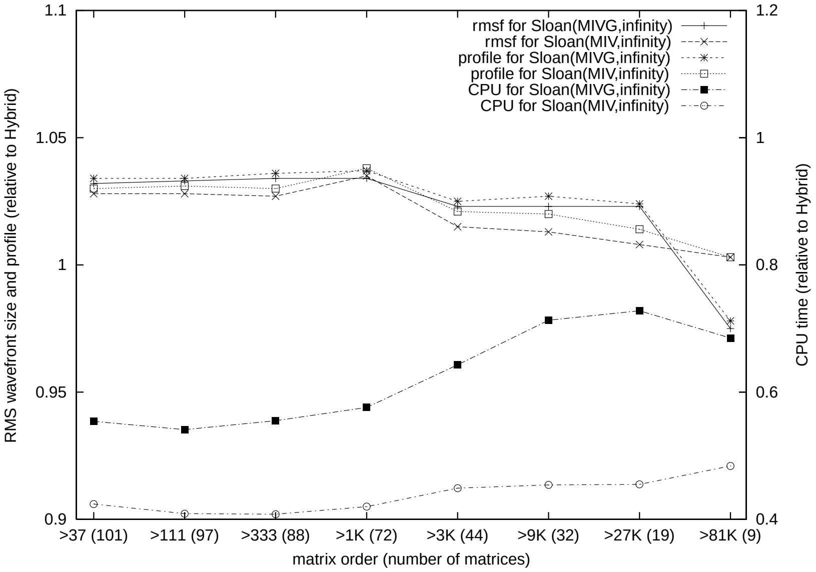


FIG. 4.3. *A comparison of the multilevel algorithms based on two approaches of selecting a maximal independent vertex set. $Sloan(MIV, \infty)$ is based on the Ruge and Stüben algorithm* [28]. *$Sloan(MIVG, \infty)$ is based on the simple greedy algorithm. All results are relative to the Hybrid algorithm. The left y-axis is used for RMS wavefront* (rmsf) *and profile; the right y-axis for CPU time.*

Figure 4.3 also includes the profile for the $Sloan(MIV, \infty)$ and $Sloan(MIVG, \infty)$ orderings relative to the Hybrid orderings. As can be seen, the trend for the profile is very similar to that of the RMS wavefront. In the remainder of this paper, we will present only RMS wavefront results.

4.2. Sloan versus Hybrid on the coarsest graph. The coarsest graph has only a small number of vertices, and so it can be rapidly ordered using any appropriate algorithm. In the results presented in the previous section, the Sloan algorithm was used, but the Hybrid algorithm can be used instead. This gives the multilevel Hybrid algorithm, $Hybrid(MIV, K)$.

Figure 4.4 compares the RMS wavefront for this algorithm with that for the Sloan and the Hybrid algorithms. We see that, for any preset maximum number of levels K, results for the $Hybrid(MIV, K)$ algorithm are comparable to those for the Hybrid algorithm. Even if there are only two levels, the quality of the ordering on the coarsest (level 2) graph is such that the application of a prolongation and refinement step is able to produce a high quality ordering on the fine graph. This is in contrast to $Sloan(MIV, 2)$, where, on the coarsest graph, the Sloan algorithm does not yield such a good ordering. As the number of levels increase, the performance of $Sloan(MIV, K)$ is comparable to $Hybrid(MIV, K)$, indicating that, because the Sloan algorithm performs well on small problems, in terms of quality, the choice between the Sloan and the Hybrid algorithms on the coarsest graph is not important when that graph is small. However, using the Sloan algorithm has the advantage of not requiring the computation of any spectral information.

The fact that the quality of the $Hybrid(MIV, K)$ orderings varies little with the number of levels K indicates that the multilevel process based on the maximal independent vertex set combined with Sloan refinement is of good quality, in the sense

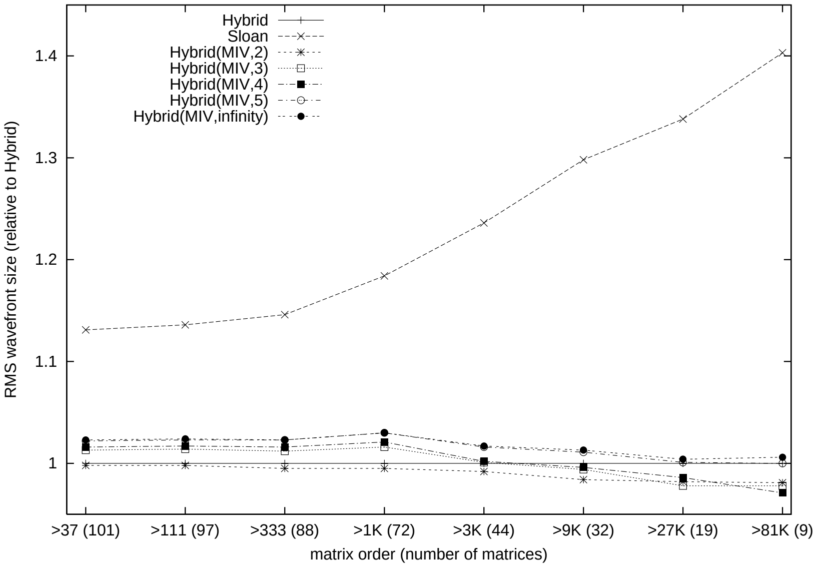


FIG. 4.4. *A comparison of the RMS wavefronts for the Sloan, the Hybrid, and the Hybrid(MIV, K) algorithms.*

that it preserves, if not enhances, the quality of the ordering achieved on the coarsest graph using the Hybrid algorithm.

The CPU time comparisons for the Sloan, the Hybrid, and the *Hybrid*(MIV, K) algorithms are given in Figure 4.5. The *Hybrid*(MIV, K) algorithm needs about half the CPU time of the Hybrid algorithm. For small K, it is slightly more expensive than *Sloan*(MIV, K) because of the extra cost associated with using the Hybrid algo-

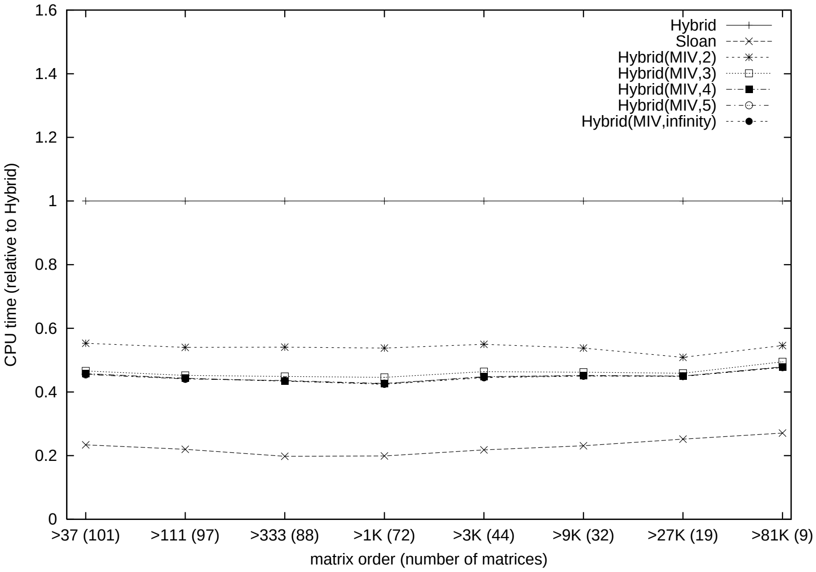


FIG. 4.5. *A comparison of the CPU times for the Sloan, the Hybrid, and the Hybrid(MIV, K) algorithms.*

rithm on the coarsest graph. For large K, the main disadvantage of $\mathit{Hybrid}(MIV, K)$ compared with $\mathit{Sloan}(MIV, K)$ is that $\mathit{Hybrid}(MIV, K)$ requires the computation of a spectral vector on the coarsest graph.

We are primarily interested in how the multilevel approach performs on large problems. Table 4.1 lists the RMS wavefronts for the Hybrid, $\mathit{Sloan}(MIV, \infty)$, and $\mathit{Hybrid}(MIV, \infty)$ algorithms for each of the test problems of order greater than 10000. The smallest wavefront for each problem (and those within 3% of the smallest) are given in bold. It can be seen that, although the three algorithms on average produce orderings with similar RMS wavefronts, their behavior on individual matrices can differ significantly. This is typical of heuristic-based algorithms and, for a given problem, which algorithm will produce the best ordering cannot be predicted a priori. Table 4.2 reports the CPU timings for the three algorithms. The multilevel algorithms $\mathit{Sloan}(MIV, \infty)$ and $\mathit{Hybrid}(MIV, \infty)$ require a similar amount of time, and both are faster than the Hybrid algorithm.

Table 4.1

RMS wavefronts for the Hybrid, the $\mathit{Sloan}(MIV, \infty)$, *and the* $\mathit{Hybrid}(MIV, \infty)$ *algorithms on matrices of order* > 10000.

Identifier	Order	Hybrid	$\mathit{Sloan}(MIV, \infty)$	$\mathit{Hybrid}(MIV, \infty)$
shuttle_eddy	10429	62.55	**60.55**	**59.75**
bcsstk17	10974	286.99	**229.63**	240.93
bcsstk18	11948	204.26	**197.25**	**195.37**
bcsstk29	13992	**193.63**	**192.74**	**192.74**
barth5	15606	**84.20**	97.11	97.75
pds10	16558	**566.94**	680.36	680.36
copter1	17222	401.04	**370.33**	**378.2**
e40r0000	17281	**162.94**	**162.38**	**162.34**
Crplat2	18010	**244.61**	254.16	255.84
tandem_vtx	18454	**288.62**	**287.30**	**282.07**
ford1	18728	**99.42**	108.95	109.56
bcsstk30	28924	**303.03**	**296.99**	321.30
Thread	29736	**1857.44**	1940.28	**1864.85**
bcsstk31	35588	**531.99**	**526.19**	558.09
finance256	37376	179.18	130.42	**116.95**
bcsstk32	44609	**471.84**	578.90	618.11
skirt	45361	**621.77**	738.62	743.59
nasasrb	54870	**336.84**	344.67	**337.29**
Srb1	54924	**327.74**	**333.39**	**332.48**
copter2	55476	597.79	726.00	**572.89**
finance512	74752	137.16	**114.58**	127.21
onera_dual	85567	**563.14**	697.67	632.99
tandem_dual	94069	451.83	**436.37**	**440.46**
MT1	97578	1035.08	1187.96	**971.93**
ford2	100196	**305.05**	327.58	343.31
Shipsec1	140874	1538.23	1666.05	**1434.77**
Fullb	199187	**1867.09**	1955.66	1992.47
Fcondp2	201822	1713.82	**1542.18**	**1559.93**
Troll	213453	3657.43	**2343.68**	2887.74
Halfb	224617	**1347.94**	1431.09	1486.26

4.3. Sensitivity of the multilevel algorithm to the priority function weights. So far, we have established that a multilevel Sloan algorithm based on a maximal independent vertex set gives orderings of comparable quality to the Hybrid algorithm and is significantly faster. Since Kumfert and Pothen [22] showed that the choice of weights can have a very important influence of the quality of Sloan's ordering, in this section we look at the sensitivity of the multilevel algorithm to the

Table 4.2

CPU time (in seconds) for the Hybrid, the Sloan(MIV,∞), and the Hybrid(MIV,∞) algorithms on matrices of order > 10000.

Identifier	Order	Hybrid	*Sloan*(MIV,∞)	*Hybrid*(MIV,∞)
shuttle_eddy	10429	0.68	0.36	0.36
bcsstk17	10974	2.97	0.88	0.89
bcsstk18	11948	1.57	0.65	0.65
bcsstk29	13992	3.52	1.43	1.44
barth5	15606	0.97	0.60	0.59
pds10	16558	4.61	1.78	1.76
copter1	17222	1.85	1.40	1.39
e40r0000	17281	3.46	1.12	1.14
Crplat2	18010	4.63	2.07	2.04
tandem_vtx	18454	2.20	0.96	0.95
ford1	18728	1.09	0.63	0.62
bcsstk30	28924	11.21	4.44	4.53
Thread	29736	25.60	10.29	10.17
bcsstk31	35588	8.07	3.01	3.01
finance256	37376	4.64	1.85	1.81
bcsstk32	44609	10.79	4.74	4.70
skirt	45361	15.18	6.12	6.10
nasasrb	54870	13.32	6.38	6.29
Srb1	54924	14.17	6.86	6.81
copter2	55476	8.13	3.89	3.79
finance512	74752	8.51	3.96	3.81
onera_dual	85567	6.80	4.35	4.29
tandem_dual	94069	7.38	4.65	4.53
MT1	97578	52.88	21.44	21.22
ford2	100196	6.78	4.16	4.11
Shipsec1	140874	56.85	18.47	18.40
Fullb	199187	64.32	29.34	28.24
Fcondp2	201822	57.53	26.95	26.74
Troll	213453	65.50	28.93	29.30
Halfb	224617	65.30	30.83	29.91

choice of weights used in the priority function (2.5), first when ordering the coarsest graph and, second, during the prolongation and refinement stages.

We have seen that, with an unlimited number of levels, the coarsest graph ordering based on both the Sloan algorithm (*Sloan*(MIV,∞)) and the Hybrid algorithm (*Hybrid*(MIV,∞)) yield orderings of similar quality. Following Reid and Scott [27], when the Sloan algorithm is used on the coarsest graph, two orderings are generated from the weight pairs $(2,1)$ and $(16,1)$, and the better of the two is chosen. Figure 4.6 illustrates the effect of using a single pair of weights. If we generate only one ordering based on a single pair of weights (chosen among $(64,1)$, $(16,1)$, $(4,1)$, $(2,1)$, $(1,1)$, $(1,2)$, $(1,4)$, $(1,16)$, and $(1,64)$), the quality of the final ordering obtained using the multilevel algorithm is not as good, although the difference is usually less than 10%. If instead of the pairs $(2,1)$ and $(1,16)$ we use $(1,2)$ and $(1,16)$, the difference in the quality of the final ordering is extremely small, indicating that the precise choice for the weights is not critical. As the coarsest graph can be ordered very quickly because of its small size, if it is important to obtain the smallest possible wavefront, it may be worthwhile to try a number of different weights and choose the best ordering among them.

We have also looked at the sensitivity of the ordering to the choice of weights for the prolongation and refinement stage of the multilevel algorithm. In all the experiments reported so far, we have used the weights $(1,2)$ and $(16,1)$. This choice was recommended in Reid and Scott [27] for the Hybrid algorithm, where it was

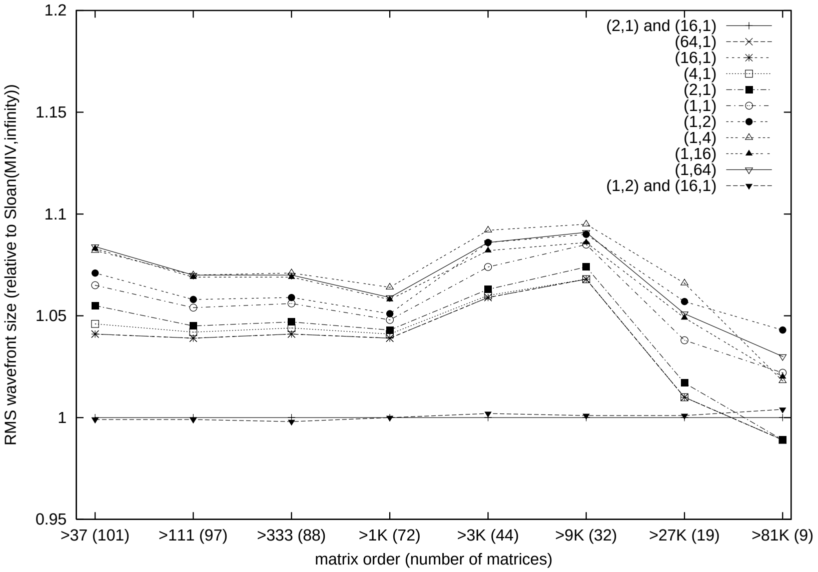


FIG. 4.6. *The effect of the weights used in the coarse graph ordering on the RMS wavefront of the multilevel algorithm. The Sloan(MIV, ∞) algorithm is used with either a single pair of weights (chosen from $(64, 1), \ldots, (1, 64)$) or the two pairs $(1, 2)$ and $(16, 1)$ on the coarsest graph. All results are relative to Sloan(MIV, ∞) with the weights $(2, 1)$ and $(16, 1)$ on the coarsest graph.*

argued that a larger W_2 in (2.5) is preferred when $p(i)$ is of good quality. Figure 4.7 illustrates the effect of varying the first pair of weights on the quality of the ordering given by *Sloan*(MIV, ∞). In this experiment, the second pair is fixed at $(16, 1)$, while the first pair is allowed to vary between $(4, 1)$ and $(1, 16)$. We see using (W_1, W_2)

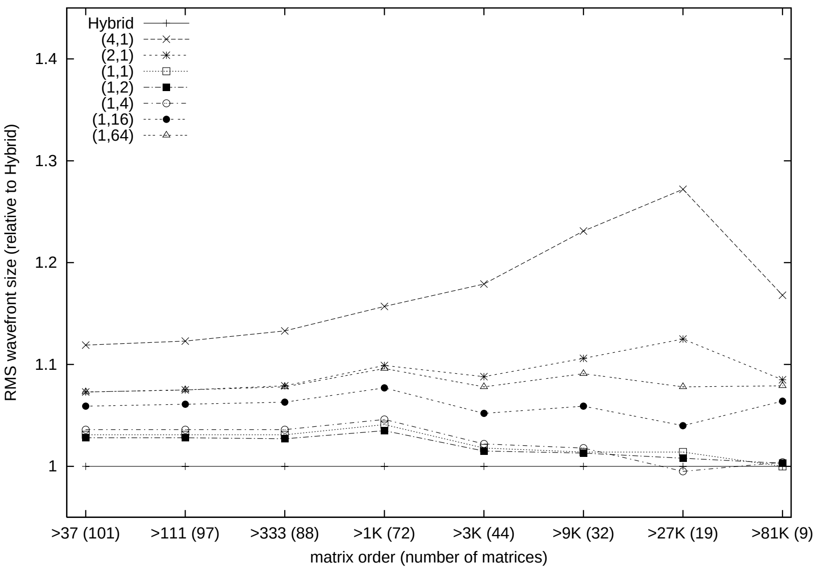


FIG. 4.7. *The effect of the first pair of weights during the refinement process on the RMS wavefront of the multilevel algorithm Sloan(MIV, ∞). The second pair of weights is fixed at $(16, 1)$. All results are relative to the Hybrid algorithm.*

with W_2 slightly larger than or equal to W_1 is beneficial. In general, the pairs $(1,1)$, $(1,2)$, and $(1,4)$ yield similar RMS wavefronts. We performed further experiments that showed the same conclusion can be drawn for the *Hybrid*(MIV,∞) algorithm.

4.4. Effect of the size of the coarsest graph. In the multilevel ordering algorithm, a parameter *MinSize* is used to control the size of the coarsest graph. Further coarsening is not carried out once the size of the current graph is less than *MinSize*. We have chosen *MinSize* $= 100$ in our work.

On small graphs, the Sloan algorithm is competitive with the Hybrid algorithm, but its competitiveness deteriorates as the size of the graph increases. Since we use the Sloan algorithm on the coarsest graph we do not, therefore, want this graph to be too large. Conversely, coarsening down to a very small number of vertices is not recommended either because applying the Sloan algorithm to such a graph does not feed into the refinement process any more information than would be given by a random ordering of the coarsest graph. Figure 4.8 demonstrates the effect of *MinSize* on the quality of the *Sloan*(MIV,∞) ordering. It is seen that *MinSize* $= 50$ or 100 gives the best results, but the precise choice of *MinSize* is not critical. A value that is either too small or too large causes deterioration in the quality of the ordering, although all the values tested yielded orderings that were within 8% of the RMS wavefront given by the Hybrid algorithm.

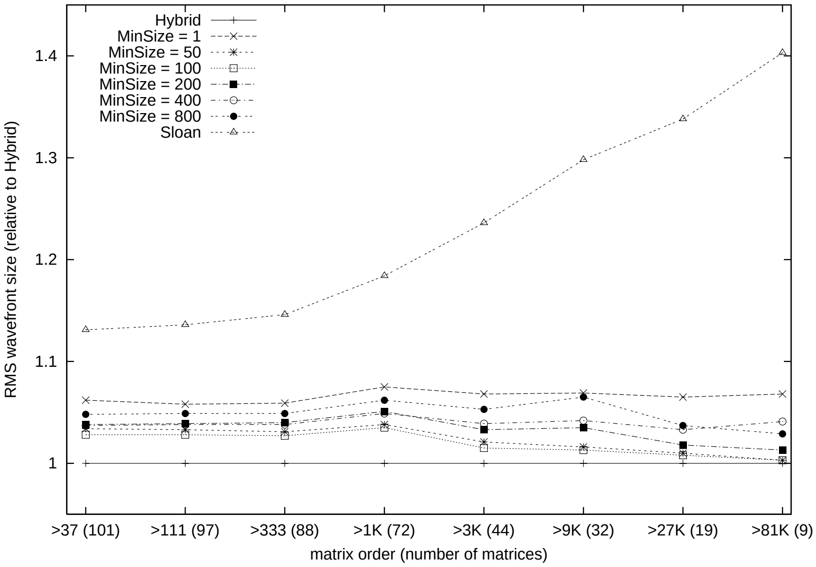


FIG. 4.8. *The effect of the size of the coarsest graph* (*MinSize*) *on the multilevel algorithm Sloan*(MIV,∞). *All results are relative to the Hybrid algorithm.*

5. Conclusions and future work. In this paper, a multilevel reordering algorithm for minimizing the profile and wavefront of sparse symmetric matrices has been developed. This algorithm, which combines a coarsening strategy based on a maximal independent vertex set with the Sloan or Hybrid algorithm on the coarsest graph, has been found to give orderings of similar quality to that of the best existing algorithm (the Hybrid algorithm of Kumfert and Pothen [22]), while being significantly faster. Of particular note is the multilevel Sloan algorithm. In common with the Sloan algorithm, this is a combinatorial algorithm, but it produces much better orderings, particularly for large problems. With no limit imposed on the maximum number of levels, the multilevel Sloan algorithm has been shown to yield orderings of similar quality to that of the Hybrid algorithm with the advantage of not requiring any spectral information.

We are investigating the possibility of further improving the multilevel algorithm so that it consistently outperforms the Hybrid algorithm, not only in terms of CPU time but also in ordering quality. We believe that to achieve this goal it may be necessary to utilize the vertex and edge weights of the coarse graphs. We are looking at whether we can include this information in the reordering and refinement of the coarse graphs. Another possible way of improving the multilevel algorithm is to use a more sophisticated ordering algorithm on the coarsest graph and then to look at translating improvements in the quality of the ordering on the coarsest graph into corresponding improvements on the original fine graph.

It may also be possible to extend our multilevel approach to the ordering of unsymmetric matrices for use with frontal solvers. This will build on the work of Scott [29] on row ordering algorithms and the work of Hu, Maguire, and Blake [16] on applying a multilevel algorithm for reordering unsymmetric matrices into bordered form.

Appendix. The test problems.

Table A.1

The suite of test problems. rmsf *is the initial RMS wavefront and ρ is the ratio between the RMS wavefronts before and after reordering with the Hybrid algorithm.*

Identifier	$\|V\|$	$\|E\|$	rmsf	ρ
1138_bus	1138	1458	87.00	6.88
barth	6691	19748	2673.11	42.86
barth4	6019	17473	404.62	7.56
barth5	15606	45878	284.36	3.38
Baug	9600	232980	1459.93	6.07
bcspwr06	1454	1923	57.68	4.92
bcspwr07	1612	2106	61.00	4.99
bcspwr08	1624	2213	64.35	5.46
bcspwr09	1723	2394	308.51	21.86
bcspwr10	5300	8271	1294.66	47.68
bcsstk08	1074	5943	239.84	3.87
bcsstk11	1473	16384	104.34	2.22
bcsstk12	1473	16384	104.34	2.22
bcsstk13	2003	40940	229.18	0.97
bcsstk14	1806	30824	115.23	1.24
bcsstk15	3948	56934	263.35	1.51
bcsstk17	10974	208838	261.87	0.91
bcsstk18	11948	68571	468.72	2.29
bcsstk21	3600	11500	119.39	2.16
bcsstk23	3134	21022	353.47	1.50
bcsstk24	3562	78174	613.47	4.98
bcsstk28	4410	107307	190.39	1.42
bcsstk29	13992	302748	551.89	2.85
bcsstk30	28924	1007284	641.77	2.12
bcsstk31	35588	572914	672.80	1.26
bcsstk32	44609	985046	2905.61	6.16
bcsstm12	1473	9093	103.80	3.81
bcsstm13	2003	9970	52.36	0.87
blckhole	2132	6370	93.98	1.68
can_1054	1054	5571	274.78	8.83

Table A.2
The suite of test problems (cont.).

Identifier	$\|V\|$	$\|E\|$	rmsf	ρ
can_1072	1072	5686	279.31	8.41
commanche_dual	7920	11880	2397.83	55.98
copter1	17222	96921	1127.23	2.81
copter2	55476	352238	21892.02	36.62
Crplat2	18010	471468	1286.38	5.26
dwg961b	961	4815	179.24	7.10
dwt	607	2262	55.43	2.11
dwt1005	1005	3808	137.66	4.03
dwt1007	1007	3784	26.93	1.31
dwt1242	1242	4592	105.20	3.18
dwt162	162	5100	18.95	2.02
dwt193	193	1650	43.84	1.80
dwt198	198	5970	30.90	4.46
dwt209	209	7670	50.32	3.44
dwt221	221	7040	50.39	5.54
dwt234	234	3000	9.36	1.92
dwt245	245	6080	18.48	1.82
dwt2680	2680	11173	234.42	6.90
dwt307	307	1108	27.36	1.06
dwt310	310	1069	9.85	1.02
dwt346	346	1440	27.15	1.37
dwt361	361	1296	15.38	1.09
dwt419	419	1572	107.07	5.50
dwt492	492	1332	79.51	10.56
dwt503	503	2762	78.60	2.77
dwt512	512	1495	14.55	1.35
dwt59	59	1040	8.22	1.72
dwt592	592	2256	55.18	2.96
dwt607	607	2262	55.43	2.11
dwt66	66	1270	11.01	3.74
dwt72	72	7500	3.46	1.03
dwt758	758	2618	37.95	3.65
dwt869	869	3208	25.02	1.49
dwt87	87	2270	29.38	4.68
dwt878	878	3285	31.92	1.38
dwt918	918	3233	131.14	6.58
dwt992	992	7876	301.99	8.85
e40r0000	17281	270737	438.20	2.69
eris1176	1176	8688	81.59	3.54
Fcondp2	201822	5546247	10322.91	6.02
finance256	37376	130560	7441.93	41.53
finance512	74752	261120	14831.49	108.13
ford1	18728	41424	1954.25	19.66
ford2	100196	222246	4282.70	14.04
Fullb	199187	5754445	45506.16	24.37
Halfb	224617	6081602	35656.53	26.45
jagmesh4	1440	4032	32.62	1.66
jagmesh5	1180	3285	32.57	1.69
jagmesh7	1138	3156	39.52	2.13
jagmesh8	1141	3162	32.17	1.38
jagmesh9	1349	3876	54.67	2.31
lshp3466	3466	10215	109.46	2.38
mhd4800b	4800	11360	17.11	4.85
MT1	97578	4827996	2815.98	2.72
nasasrb	54870	1311227	401.75	1.19
nos7	729	1944	76.26	1.16
onera_dual	85567	166817	9336.32	16.58
pds10	16558	66550	1129.78	1.99
plat1919	1919	15240	739.36	16.38

Table A.3
The suite of test problems (cont.).

Identifier	$\|V\|$	$\|E\|$	rmsf	ρ
qc2534	2534	230413	177.61	1.00
s3rmt3m3	5357	101169	478.58	3.81
Shipsec1	140874	3836265	3290.65	2.14
shuttle_eddy	10429	46585	1161.54	18.57
skirt	45361	1268228	1092.01	1.76
Srb1	54924	1453614	1527.03	4.66
sstmodel	3345	9702	34.27	1.29
tandem_dual	94069	183212	5831.87	12.91
tandem_vtx	18454	117448	4705.44	16.30
Thread	29736	2220156	6676.39	3.59
Troll	213453	5885829	4703.06	1.29
zenios	2873	12159	431.21	54.57

Acknowledgments. We would like to thank our colleagues Iain Duff and John Reid at the Rutherford Appleton Laboratory for helpful comments on a draft of this paper. We are grateful to Christian Damhaug and Gary Kumfert for supplying some of the test problems used in this paper. Thanks are also due to the referees and the editor for very careful reading and detailed comments.

REFERENCES

[1] S. T. Barnard, A. Pothen, and H. D. Simon, *A spectral algorithm for envelope reduction of sparse matrices*, Numer. Linear Algebra Appl., 2 (1995), pp. 317–334.

[2] S. T. Barnard and H. D. Simon, *Fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems*, Concurrency: Practice and Experience, 6 (1994), pp. 101–117.

[3] E. G. Boman and B. Hendrickson, *A Multilevel Algorithm for Reducing the Envelope of Sparse Matrices*, Technical Report SCCM-96-14, Stanford University, Stanford, CA, 1996.

[4] E. Cuthill and J. McKee, *Reducing the bandwidth of sparse symmetric matrices*, in Proceedings of the 24th National Conference of the ACM, 1969, pp. 157–172.

[5] I. S. Duff, J. K. Reid, and J. A. Scott, *The use of profile reduction algorithms with a frontal code*, Internat. J. Numer. Methods Engrg., 28 (1989), pp. 2555–2568.

[6] G. C. Everstine, *A comparison of three resequencing algorithms for the reduction of matrix profile and wavefront*, Internat. J. Numer. Methods Engrg., 14 (1979), pp. 837–853.

[7] M. Fiedler, *Algebraic connectivity of graphs*, Czechoslovak Math. J., 23 (1973), pp. 298–305.

[8] M. Fiedler, *A property of eigenvectors of non-negative symmetric matrices and its application*, Czechoslovak Math. J., 25 (1975), pp. 619–633.

[9] A. George, *Computer Implementation of the Finite-Element Method*, Ph.D. thesis, Department of Computer Science, Stanford University, Stanford, CA, 1971.

[10] A. George and A. Pothen, *An analysis of spectral envelope reduction via quadratic assignment problems*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 706–732.

[11] N. E. Gibbs, W. G. Poole Jr., and P. K. Stockmeyer, *An algorithm for reducing the bandwidth and profile of a sparse matrix*, SIAM J. Numer. Anal., 13 (1976), pp. 236–250.

[12] N. E. Gibbs, *Algorithm* 509: *A hybrid profile reduction algorithm*, ACM Trans. Math. Software, 2 (1976), pp. 378–387.

[13] B. Hendrickson and R. Leland, *A Multilevel Algorithm for Partitioning Graphs*, Technical Report SAND93-1301, Sandia National Laboratories, Albuquerque, NM, 1993; also available online from http://www.supercomp.org/sc95/proceedings/509_BHEN/SC95.HTM.

[14] B. Hendrickson and R. Leland, *The Chaco User's Guide, Version* 2.0, Technical Report SAND94-2692, Sandia National Laboratories, Albuquerque, NM, 1995.

[15] HSL 2000, *A Collection of Fortran Codes for Large Scale Scientific Computation*, http://www.numerical.rl.ac.uk/Activity/HSL, 2000.

[16] Y. F. Hu, K. C. F. Maguire, and R. J. Blake, *A multilevel unsymmetric matrix ordering algorithm for parallel process simulation*, Comput. Chemical Engrg., 23 (2000), pp. 1631–1647.

[17] Y. F. Hu and J. A. Scott, *Multilevel Algorithms for Wavefront Reduction*, Technical Report RAL-TR-2000-031, Rutherford Appleton Laboratory, Didcot, Oxon, UK, 2000.

[18] G. Karypis and V. Kumar, *Parallel Multilevel Graph Partitioning*, Technical Report, Department of Computer Science, University of Minnesota, Minneapolis, MN, 1995.

[19] G. Karypis and V. Kumar, *Multilevel k-way partitioning scheme for irregular graphs*, J. Parallel Distrib. Comput., 48 (1998), pp. 96–129.

[20] G. Karypis and V. Kumar, *A fast and high quality multilevel scheme for partitioning irregular graphs*, SIAM J. Sci. Comput., 20 (1998), pp. 359–392.

[21] B. W. Kernighan and S. Lin, *An efficient heuristic procedure for partitioning graphs*, Bell System Tech. J., 49 (1970), pp. 291–308.

[22] G. Kumfert and A. Pothen, *Two improved algorithms for envelope and wavefront reduction*, BIT, 35 (1997), pp. 1–32.

[23] Y. Lin and J. Yuan, *Minimum Profile of Grid Networks in Structure Analysis*, Preprint, Department of Mathematics, Zhengzhou University, Zhengzhou, Henan, People's Republic of China, 1993.

[24] W. H. Liu and A. H. Sherman, *Comparative analysis of the Cuthill–McKee and the reverse Cuthill–McKee ordering algorithms for sparse matrices*, SIAM J. Numer. Anal., 13 (1976), pp. 198–213.

[25] G. H. Paulino, I. F. M. Menezes, M. Gattass, and S. Mukherjee, *Node and element resequencing using the Laplacian of a finite element graph, Part* I, Internat. J. Numer. Methods Engrg., 37 (1994), pp. 1511–1530.

[26] G. H. Paulino, I. F. M. Menezes, M. Gattass, and S. Mukherjee, *Node and element resequencing using the Laplacian of a finite element graph, Part* II, Internat. J. Numer. Methods Engrg., 37 (1994), pp. 1531–1555.

[27] J. K. Reid and J. A. Scott, *Ordering symmetric sparse matrices for small profile and wavefront*, Internat. J. Numer. Methods Engrg., 45 (1999), pp. 1737–1755.

[28] J. W. Ruge and K. Stüben, *Algebraic multigrid (AMG)*, in Multigrid Methods, S. F. McCormick, ed., Frontiers Appl. Math. 3, SIAM, Philadelphia, 1987, pp. 73–130.

[29] J. A. Scott, *A new row ordering strategy for frontal solvers*, Numer. Linear Algebra Appl., 6 (1999), pp. 1–23.

[30] H. D. Simon, *Partitioning of unstructured problems for parallel processing*, Computer Systems Engrg., 2 (1991), pp. 135–148.

[31] S. W. Sloan, *An algorithm for profile and wavefront reduction of sparse matrices*, Internat. J. Numer. Methods Engrg., 23 (1986), pp. 239–251.

[32] S. W. Sloan, *A FORTRAN program for profile and wavefront reduction*, Internat. J. Numer. Methods Engrg., 28 (1989), pp. 2555–2568.

[33] C. Walshaw, M. Cross, and M. Everett, *Dynamic Mesh Partitioning: A Unified Optimisation and Load-Balancing Algorithm*, Technical Report 95/IM/06, University of Greenwich, London, 1995.

SIAM J. SCI. COMPUT.
Vol. 23, No. 4, pp. 1376–1395

ON THE SOLUTION OF EQUALITY CONSTRAINED QUADRATIC PROGRAMMING PROBLEMS ARISING IN OPTIMIZATION*

NICHOLAS I. M. GOULD†, MARY E. HRIBAR‡, AND JORGE NOCEDAL§

Abstract. We consider the application of the conjugate gradient method to the solution of large equality constrained quadratic programs arising in nonlinear optimization. Our approach is based implicitly on a reduced linear system and generates iterates in the null space of the constraints. Instead of computing a basis for this null space, we choose to work directly with the matrix of constraint gradients, computing projections into the null space by either a normal equations or an augmented system approach. Unfortunately, in practice such projections can result in significant rounding errors. We propose iterative refinement techniques, as well as an adaptive reformulation of the quadratic problem, that can greatly reduce these errors without incurring high computational overheads. Numerical results illustrating the efficacy of the proposed approaches are presented.

Key words. nonlinear optimization, conjugate gradient method, quadratic programming, preconditioning, iterative refinement

AMS subject classifications. 65K10, 49N, 49M, 65F10, 90C06, 90C30

PII. S1064827598345667

1. Introduction.

A variety of algorithms for linearly and nonlinearly constrained optimization (e.g., [9, 14, 15, 36, 37]) use the conjugate gradient (CG) method [28] to solve subproblems of the form

$$\underset{x}{\text{minimize}} \quad q(x) = \tfrac{1}{2}x^T Hx + c^T x \tag{1.1}$$

$$\text{subject to} \qquad Ax = b. \tag{1.2}$$

In nonlinear optimization, the n-vector c usually represents the gradient ∇f of the objective function or the gradient of the Lagrangian, the $n \times n$ symmetric matrix H stands for either the Hessian of the Lagrangian or an approximation to it, and the solution x represents a search direction. The equality constraints $Ax = b$ are obtained by linearizing the constraints of the optimization problem at the current iterate. We will assume here that A is an $m \times n$ matrix, with $m < n$, and that A has full row rank so that the constraints $Ax = b$ constitute m linearly independent equations. We also assume for convenience that H is positive definite in the null space of the constraints, as this guarantees that (1.1)–(1.2) has a unique solution.

As we shall see in section 2.1, the solution of (1.1)–(1.2) can be characterized in terms of a nonunique matrix Z whose columns form a basis for the null space of A. Numerous options are available for computing and representing Z, both explicitly and

[*] Received by the editors October 7, 1998; accepted for publication (in revised form) August 22, 2001; published electronically December 18, 2001.
http://www.siam.org/journals/sisc/23-4/34566.html

[†] Computational Science and Engineering Department, Rutherford Appleton Laboratory, Chilton, Oxfordshire OX11 0QX, England (n.gould@rl.ac.uk).

[‡] Cray Inc., 411 1st Avenue S, Suite 600, Seattle, WA 98104-2860 (marybeth@cray.com). This author's research was supported by Department of Energy grant DE-FG02-87ER25047-A004.

[§] ECE Department, Northwestern University, Evanston, IL 60208 (nocedal@ece.nwu.edu). This author's research was supported by National Science Foundation grant CDA-9726385 and by Department of Energy grant DE-FG02-87ER25047-A004.

implicitly [11, 12, 19, 26, 41, 44]. In the context of large-scale optimization, operations with Z and Z^T can be performed using the LU factorization of a nonsingular submatrix of A; see, for example, [20].

In this paper, we consider techniques for solving (1.1)–(1.2) that use a preconditioned CG method and retain feasibility of the iterates by performing projections into the null space of A without a representation of Z. Unfortunately, a straightforward implementation of these techniques may produce computational errors that cause deteriorating feasibility of the CG iterates. We describe iterative refinement techniques that can improve accuracy, when needed. We also propose a mechanism for redefining the vector c adaptively that does not change the solution of the quadratic problem but that has more favorable numerical properties.

Notation. Throughout the paper $\|\cdot\|$ stands for the ℓ_2 matrix or vector norm. We will denote the floating-point *unit roundoff* (or machine precision) by ϵ_m. For double precision IEEE arithmetic, $\epsilon_m \approx 10^{-16}$. We let $\kappa(A)$ denote the condition number of A, i.e., $\kappa(A) = \sigma_1/\sigma_m$, where $\sigma_1 \geq \cdots \geq \sigma_m > 0$ are the nonzero singular values of A.

2. The CG method with linear constraints.

We now look at applying CG to approximate the solution of the quadratic problem (1.1)–(1.2). First, we present CG method for a reduced problem; then we show how to apply CG to the full system with a scaled projection operator. Finally, we consider the problem (1.1)–(1.2) with a trust region constraint and show how the given CG methods apply.

2.1. The CG method for the reduced system.

A common approach for solving linearly constrained problems is to eliminate the constraints and solve a reduced problem (cf. [21, 39]). More specifically, suppose that Z is an $n \times (n-m)$ matrix spanning the null space of A. Then $AZ = 0$, the columns of A^T together with the columns of Z span $\mathbf{R}^n$, and any solution x^* of the linear equations $Ax = b$ can be written as

$$x^* = A^T x_A{}^* + Z x_Z{}^* \tag{2.1}$$

for some vectors $x_A{}^* \in \mathbf{R}^m$ and $x_Z{}^* \in \mathbf{R}^{n-m}$. The constraints $Ax = b$ yield

$$AA^T x_A{}^* = b, \tag{2.2}$$

which determines the vector $x_A{}^*$. Substituting (2.1) into (1.1), and omitting constant terms ($x_A{}^*$ is a constant now), we see that $x_Z{}^*$ solves the reduced problem

$$\underset{x_Z}{\text{minimize}} \quad \tfrac{1}{2} x_Z{}^T H_{ZZ} x_Z + c_Z{}^T x_Z, \tag{2.3}$$

where

$$H_{ZZ} = Z^T H Z, \quad c_Z = Z^T (H A^T x_A{}^* + c).$$

As we have assumed that the reduced Hessian H_{ZZ} is positive definite, the solution of (2.3) is equivalent to that of the linear system

$$H_{ZZ} x_Z = -c_Z. \tag{2.4}$$

We can now apply the conjugate gradient method to compute an approximate solution of the problem (2.3), or, equivalently, the system (2.4), and substitute this into (2.1) to obtain an approximate solution of the quadratic program (1.1)–(1.2).

This strategy of computing the normal component $A^T x_A$ exactly and the tangential component Zx_Z inexactly is followed in many nonlinear optimization algorithms which ensure that, once linear constraints are satisfied, they remain so throughout the remainder of the optimization calculation (cf. [21]).

Let us now consider the practical application of the CG method to the reduced system (2.4). It is well known that *preconditioning* can improve the rate of convergence of the CG iteration (cf. [3]). We therefore assume that a preconditioner W_{ZZ} is given, where W_{ZZ} is a symmetric, positive definite matrix of dimension $n-m$, which might be chosen to reduce the span of, and to cluster, the eigenvalues of $W_{ZZ}^{-1}H_{ZZ}$. Ideally, one would like to choose W_{ZZ} so that $W_{ZZ}^{-1}H_{ZZ} = I$, and thus $W_{ZZ} = Z^T HZ$ is an ideal preconditioner. Based on this ideal, we consider in this paper preconditioners of the form $W_{ZZ} = Z^T GZ$, where G is a symmetric matrix such that $Z^T GZ$ is positive definite. Some choices of G will be discussed in the next section.

For preconditioners of the form $W_{ZZ} = Z^T GZ$, the preconditioned CG method applied to the $(n-m)$-dimensional reduced system $H_{ZZ}x_Z = -c_Z$ is as follows (see, e.g., [22, p. 532]).

Algorithm 2.1 (preconditioned CG for reduced systems). *Choose an initial point x_Z, compute $r_Z = Z^T HZx_Z + c_Z$, $g_Z = (Z^T GZ)^{-1}r_Z$, and $p_Z = -g_Z$. Repeat the following steps, until a termination test is satisfied:*

$$\alpha = {r_Z}^T g_Z / {p_Z}^T Z^T HZp_Z, \tag{2.5}$$

$$x_Z \leftarrow x_Z + \alpha p_Z, \tag{2.6}$$

$${r_Z}^+ = r_Z + \alpha Z^T HZp_Z, \tag{2.7}$$

$${g_Z}^+ = (Z^T GZ)^{-1}{r_Z}^+, \tag{2.8}$$

$$\beta = ({r_Z}^+)^T {g_Z}^+ / {r_Z}^T g_Z, \tag{2.9}$$

$$p_Z \leftarrow -{g_Z}^+ + \beta p_Z, \tag{2.10}$$

$$g_Z \leftarrow {g_Z}^+ \quad \textit{and} \quad r_Z \quad \leftarrow \quad {r_Z}^+. \tag{2.11}$$

This iteration may be terminated, for example, when ${r_Z}^T(Z^T GZ)^{-1}r_Z$ is sufficiently small.

Several algorithms for large-scale optimization are based on combining a suitable representation of Z with CG methods for solving the reduced system [18, 33, 46]. Coleman and Verma [13] and Nash and Sofer [38] have proposed strategies for defining reduced-system preconditioners which approximate $Z^T HZ$ in different ways.

We present Algorithm 2.1 for illustrative purposes only. In the next section, we describe modifications to this algorithm which make it possible to avoid operating with the null space basis Z.

2.2. The CG method for the full system. If we were to compute an approximate solution using Algorithm 2.1, it must be multiplied by Z and substituted in (2.1) to give the approximate solution of the quadratic program (1.1)–(1.2). Alternatively, we may rewrite Algorithm 2.1 so that the multiplication by Z and the addition of the term $A^T {x_A}^*$ is computed within the CG iteration. To do so, we introduce, in the following algorithm, the n-vectors x, r, g, p which satisfy $x = Zx_Z + A^T {x_A}^*$, $Z^T r = r_Z$, $g = Zg_Z$, and $p = Zp_Z$. We also define the scaled projection matrix

$$P = Z(Z^T GZ)^{-1}Z^T. \tag{2.12}$$

We note, for future reference, that P is independent of the choice of null space basis Z.

ALGORITHM 2.2 (preconditioned CG in expanded form). *Choose an initial point x satisfying $Ax = b$, compute $r = Hx + c$, $g = Pr$, and $p = -g$. Repeat the following steps, until a convergence test is satisfied:*

$$\alpha = r^T g / p^T Hp, \tag{2.13}$$

$$x \leftarrow x + \alpha p, \tag{2.14}$$

$$r^+ = r + \alpha Hp, \tag{2.15}$$

$$g^+ = Pr^+, \tag{2.16}$$

$$\beta = (r^+)^T g^+ / r^T g, \tag{2.17}$$

$$p \leftarrow -g^+ + \beta p, \tag{2.18}$$

$$g \leftarrow g^+ \quad \textit{and} \quad r \leftarrow r^+. \tag{2.19}$$

This will be the main algorithm studied and further refined in this paper. It is important to notice that this algorithm, unlike its predecessor, is independent of the choice of Z. In the next section, different choices for P will be presented.

Note that the vector g^+, which we call the *preconditioned residual*, has been defined to be in the null space of A. As a result, in exact arithmetic, all the search directions p generated by Algorithm 2.2 will also lie in the null space of A, and thus the iterates x will all satisfy $Ax = b$. However, computed representations of the scaled projection P can produce rounding errors that may cause p to have a significant component outside the null space of A, leading to convergence difficulties. This will be the subject of later sections of the paper.

Several types of stopping tests can be used, but since their choice depends on the requirements of the optimization method, we shall not discuss them here. In the numerical tests reported in this paper, we terminate the CG iteration based on the quantity $r^T g \equiv r^T Pr \equiv g^T Gg$. An initial point satisfying $Ax = b$ can be computed, for example, by solving the normal equations (2.2).

Two simple choices of G are $G = \operatorname{diag}(H)$, and $G = I$. The first choice may be appropriate when the diagonal elements of H are of widely different magnitudes. This is the case, for example, in barrier methods for constrained optimization that handle bound constraints $l \le x \le u$ by adding terms of the form $-\mu \sum_{i=1}^{n} (\log(x_i - l_i) + \log(u_i - x_i))$ to the objective function for some positive barrier parameter μ. The second choice, $G = I$, arises in trust region methods, as we discuss next.

2.3. The CG method and the trust region problem. In trust region methods, the problem (1.1)–(1.2) also contains a trust region constraint of the form $\|x\| \le \Delta$. Steihaug [43] noted, however, that the trust region constraint can be easily imposed if the initial estimate of the solution of (1.1)–(1.2) is chosen to be the vector zero. In this case the CG iterates are monotonically increasing in norm, and the CG iteration can be terminated as soon as the norm of one of the iterates exceeds the trust region radius. No other changes to the CG iteration are needed.

For the reduced problem (2.3), the added trust region constraint has the form $\|Zx_z\| \le \Delta_z$. In order to transform it into a spherical constraint, we introduce the change of variables $x_z \leftarrow (Z^T Z)^{-1/2} x_z$ whose effect in the CG iteration is identical to that of replacing $(Z^T GZ)^{-1}$ by $(Z^T Z)^{-1}$ in (2.8). Thus, the choice $G = I$ arises in several trust region methods for constrained optimization [9, 15, 16, 27, 36, 40, 47]. Since the role of this matrix is not to produce a clustering of the eigenvalues, we will regard Algorithm 2.2 with the choice $G = I$ as an *unpreconditioned* CG iteration.

3. The CG algorithm without a null space basis. We are interested here in using Algorithm 2.2 in such a way that a representation of Z is not necessary. This will be possible because, as is well known, there are alternative ways of expressing the scaled projection operator (2.12).

3.1. Computing projections. We now discuss how to apply the projection operator $Z(Z^TGZ)^{-1}Z^T$ to a vector without a representation of the null space basis Z.

Let us begin by considering the simple case when $G = I$, so that P is the orthogonal projection operator onto the null space of A. We denote it by P_Z, i.e.,

$$P_Z = Z(Z^TZ)^{-1}Z^T. \tag{3.1}$$

Thus the preconditioned residual g^+ (2.16) is the result of projecting r^+ into the null space of A and can be written as

$$g^+ = P_Z r^+. \tag{3.2}$$

This projection can be performed in two alternative ways.

The first is to replace P_Z by the equivalent formula

$$P_A = I - A^T(AA^T)^{-1}A \tag{3.3}$$

and thus to replace (3.2) with

$$g^+ = P_A r^+. \tag{3.4}$$

We can express this as

$$g^+ = r^+ - A^Tv^+, \tag{3.5}$$

where v^+ is the solution of

$$AA^Tv^+ = Ar^+. \tag{3.6}$$

Noting that (3.6) are the normal equations, it follows that v^+ is the solution of the least squares problem

$$\underset{v}{\text{minimize}} \quad \|r^+ - A^Tv^+\| \tag{3.7}$$

and that the desired projection g^+ is the corresponding residual. The approach (3.5)–(3.6) for computing the projection $g^+ = P_Z r^+$ will be called the *normal equations approach*. In this paper, we assume that (3.6) will be solved using a Cholesky factorization of AA^T.

The second possibility is to express the projection (3.2) as the solution of the augmented system

$$\begin{pmatrix} I & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} g^+ \\ v^+ \end{pmatrix} = \begin{pmatrix} r^+ \\ 0 \end{pmatrix}. \tag{3.8}$$

In this paper, we assume that this system will be solved by means of a symmetric indefinite factorization that uses 1 × 1 and 2 × 2 pivots [22]. We refer to this as the *augmented system approach*.

Now let us suppose that preconditioning has the more general form

$$g^+ = P_{Z:G} r^+, \quad \text{where} \quad P_{Z:G} = Z(Z^T G Z)^{-1} Z^T. \tag{3.9}$$

This may be expressed as

$$g^+ = P_{A:G} r^+, \quad \text{where} \quad P_{A:G} = G^{-1}\left(I - A^T(AG^{-1}A^T)^{-1}AG^{-1}\right) \tag{3.10}$$

if G is nonsingular, and can be found as the solution of

$$\begin{pmatrix} G & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} g^+ \\ v^+ \end{pmatrix} = \begin{pmatrix} r^+ \\ 0 \end{pmatrix} \tag{3.11}$$

whenever $z^T G z \neq 0$ for all nonzero z for which $Az = 0$ (see, e.g., [21, section 5.4.1]). While (3.10) is far from appealing when G^{-1} does not have a simple form, (3.11) is a useful generalization of (3.8). Clearly, the system (3.8) may be obtained from (3.11) by setting $G = I$, and the perfect preconditioner results if $G = H$, but other choices for G are also possible; all that is required is that $z^T G z > 0$ for all nonzero z for which $Az = 0$. The idea of using the projection (3.3) in the CG method dates back to at least [42]; the alternative (3.11), and its special case (3.8), are proposed in [10], although [10] unnecessarily requires that G be positive definite. A more recent study on preconditioning the projected CG method is [13], while the eigenstructure of the preconditioned system is examined by [35, 37].

Interestingly, preconditioning in Coleman and Verma's null space approach [13] requires the solution of systems like (3.11), but it allows A to be replaced by a sparser matrix. (The price to pay for this relaxation is that products involving a suitable null space matrix are required.) Such an approach has considerable merit, especially in the case where using the exact A leads to significant fill-in during the factorization of the coefficient matrix of (3.11). It remains to be seen how such an approach compares with those we propose here when used in algorithms for large-scale constrained optimization.

Note that (3.4), (3.8), and (3.11) do not make use of a null space basis Z and require only factorization of matrices involving A. Significantly, all three forms allow us to compute an initial point satisfying $Ax = b$, the first because it relies on a factorization of AA^T, from which we can compute $x = A^T(AA^T)^{-1}b$, while factorizations of the system matrices in (3.8) and (3.11) allow us to find a suitable x by solving

$$\begin{pmatrix} I & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ b \end{pmatrix} \quad \text{or} \quad \begin{pmatrix} G & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ b \end{pmatrix}.$$

Unfortunately all three of our proposed alternatives, (3.4), (3.8), and (3.11) for computing g^+ can give rise to significant roundoff errors that prevent the iterates from remaining in the null space of A, particularly as the CG iterates approach the solution. The difficulties are caused by the fact that, as the iterations proceed, the projected vector $g^+ = Pr^+$ becomes increasingly small while r^+ does not. Indeed, the optimality conditions of the quadratic program (1.1)–(1.2) state that the solution x^* satisfies

$$Hx^* + c = A^T \lambda \tag{3.12}$$

for some Lagrange multiplier vector λ. The vector $Hx + c$, which is denoted by r in Algorithm 2.2, will generally stay bounded away from zero, but as indicated by

(3.12), it will become increasingly closer to the range of A^T. In other words, r will tend to become orthogonal to Z, and hence, from (3.9), the preconditioned residual g will converge to zero so long as the smallest eigenvalue of Z^TGZ is bounded away from zero.

That this discrepancy in the magnitudes of $g^+ = Pr^+$ and r^+ will cause numerical difficulties is apparent from (3.5), which shows that significant cancellation of digits will usually take place. The generation of harmful roundoff errors is also apparent from (3.8) and (3.11) because g^+ will be small while the remaining components v^+ remain large. Since the magnitude of the errors generated in the solution of (3.8) and (3.11) is governed by the size of the large component v^+, the vector g^+ is likely to contain large relative errors. These arguments will be made more precise in the next section.

Now consider an example problem. Since the goal of this paper is not to evaluate the efficiency of particular choices of preconditioners, in all the examples given in this paper we will choose $G = I$, which, as we have mentioned, arises in trust region optimization methods without preconditioning. To assess the techniques to be proposed, we need to measure the closeness of g to the null space of A. For this purpose, we have chosen

$$\cos\theta = \max_i \left\{ \frac{A_i^T g}{||A_i||\ ||g||} \right\}, \tag{3.13}$$

where A_i is the ith row of A. The value of $\cos\theta$ provides a relative measure of orthogonality with the property that, for nonzero g, it vanishes if and only if g lies in the null space of A.

Example 3.1. We applied Algorithm 2.2 to solve problem CVXQP3 from the CUTE collection [6], with $n = 1000$ and $m = 750$, where the simple bounds were removed to create a problem of the form (1.1)–(1.2). We used both the normal equations (3.5)–(3.6) and augmented system (3.8) approaches to compute the projection and define $G = I$. The results are given in Figure 3.1, which plots $\sqrt{r^Tg} = ||r_z||$ (resid), the norm of the null space component of the residual, as a function of the iteration number. In both cases the CG iteration was terminated when r^Tg became negative, which indicates that severe errors have occurred since r^Tg must be positive. (Continuing the iteration past this point resulted in oscillations in the norm of the gradient without any significant improvement.) At iteration 50 of both runs, r is of order 10^5 whereas its projection g is of order 10^{-1}. Figure 3.1 also plots (3.13), the cosine of the angle between the preconditioned residual g and the rows of A. Note that this cosine, which should be zero in exact arithmetic, increases and indicates that the CG iterates leave the constraint manifold $Ax = b$.

We believe it is reasonable to attribute the failure of the CG algorithm to the deviation of the iterates from the constraint manifold $Ax = b$, since the derivation of Algorithm 2.2 from its predecessor is predicated on the assumption that the search is restricted to this manifold. An analysis by Arioli, Duff, and de Rijk [2] (which improves on [1]) indicates that, with care, it is possible to ensure that the backward error[1]

$$A_i^T g^+ / (|A||g^+|)_i$$

of the computed g^+ is of the order of the machine precision, ϵ_m for the case $G = I$. (Here $|\cdot|$ denotes the componentwise absolute value.)

[1] This definition needs to be modified if $|A||g^+|$ is (close to) zero. See [1] for details.

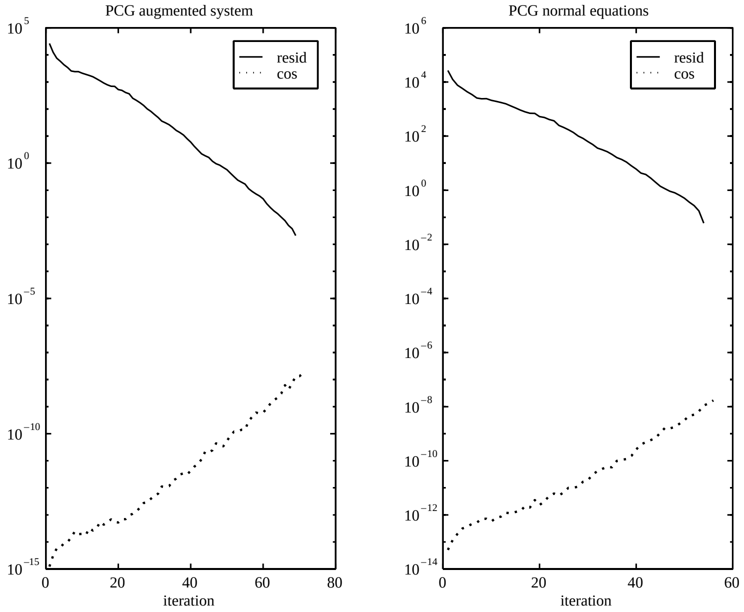


FIG. 3.1. *The CG method with two options for the projection.*

Errors such as those illustrated in Example 3.1 are not uncommon in optimization calculations based on Algorithm 2.2. This is of concern, as it may cause the outer optimization algorithms to fail to achieve feasibility or to require many iterations to do so. A particular example is given by problem ORTHREGA from the CUTE collection, which cannot be solved to a prescribed accuracy using the trust region CG approach of [31]; see [31, pp. 33–34] and section 7.

In sections 5 and 6 we propose several remedies. One of them is based on an adaptive redefinition of r that attempts to minimize the differences in magnitudes between $g^+ = Pr^+$ and r^+. We also describe several forms of iterative refinement for the projection operation. All these techniques are motivated by the roundoff error analysis given next.

4. Sources of errors. We now present error bounds that support the arguments made in the previous section, particularly the claim that the most problematic situation occurs in the latter stages of the CG iteration when g^+ is converging to zero, but r^+ is not. That is, we shall presume that $\|r^+\|$ is much larger than its projection $\|g^+\|$. For simplicity, we shall assume henceforth that A has been scaled so that $\|A\| = \|A^T\| = 1$ and shall only consider the simplest possible choice, $G = I$. Any computed, as opposed to exact, quantity will be denoted by a subscript c.

First consider the *normal equations approach.* Here the projection $g^+ = P_A r^+$ is given by (3.5), where (3.6) is solved by means of the Cholesky factorization of AA^T. In finite precision, it is straightforward to deduce that the relative error in the projection satisfies[2]

$$\frac{\|g^+ - g_c^+\|}{\|g^+\|} \leq \gamma \epsilon_m \kappa^2(A) \frac{\|v^+\|}{\|g^+\|}, \tag{4.1}$$

[2] If $\|g^+\|$ is small, it is preferable to replace the denominators in (4.1) by $\max(\|g^+\|, \epsilon)$, where ϵ is a suitable multiple (e.g., 10) of ϵ_m.

where $\gamma = 2.5n^{3/2}$, using the analysis of [5, p. 49].[3] We can thus conclude that the error in the projection (4.1) can be significant when $\kappa(A)$ or

$$\frac{\|v^+\|}{\|g^+\|} = \frac{\|v^+\|}{\|P_A r^+\|} \approx \frac{\|r^+\|}{\|P_A r^+\|} \tag{4.2}$$

is large, the latter approximation resulting from (3.5) and the assumption that $\|A\| = 1$, since then $\|r^+\| \approx \|A^T v^+\| \leq \|v^+\|$.

When the condition number $\kappa(A)$ is moderate, the contribution of the ratio (4.2) to the relative error (4.1) is normally not large enough to cause failure of the outer optimization calculation. For example, a stopping test in a nonlinear optimization algorithm, which causes termination when projected residual g^+ is (say) 10^{-6} times smaller in norm than the initial residual, has a ratio (4.2) of roughly 10^6. In this case, using double precision arithmetic, one would have sufficient accuracy to make progress toward the solution. However, as the condition number $\kappa(A)$ grows, the loss of significant digits becomes severe, especially since $\kappa(A)$ appears squared in (4.1).

Now consider the *augmented system approach* (3.11). Again we will focus on the choice $G = I$ for which the preconditioned residual $g^+ = Pr^+$ is computed by solving the system (3.8) using a direct method. There are a number of such methods, the strategies of Bunch and Kaufman [7] and Duff and Reid [17] being the best known examples for dense and sparse matrices, respectively. Both form the LDL^T factorization of the augmented matrix (i.e., the matrix appearing on the left-hand side of (3.8)), where L is unit lower triangular and D is block diagonal with 1×1 or 2×2 blocks. This approach is usually (but not always) more stable than the normal equations approach.

In the case which concerns us most, when $\|g^+\|$ converges to zero while $\|v^+\|$ is bounded, an error analysis [4] shows that

$$\frac{\|g^+ - g_c^+\|}{\|g^+\|} \leq \eta\epsilon_m(\sigma_1 + \kappa(A))\frac{\|v^+\|}{\|g^+\|},$$

where η is the product of a low degree polynomial in $n + m$ with the growth factor from the elimination, while σ_1 is the largest singular value of A. It is interesting to compare this bound with (4.1). We see that the ratio (4.2) again plays a crucial role in the analysis and that the augmented system approach is likely to give a more accurate solution g^+ than the method of normal equations in this case. This cannot be stated categorically, however, since the size of the factor η is difficult to predict.

The residual update strategy described in section 6 aims at minimizing the size of the ratio (4.2), and, as we will see, has a highly beneficial effect in Algorithm 2.2. Before presenting it, we discuss various iterative refinement techniques designed to improve the accuracy of the projection operation.

5. Iterative refinement.

Iterative refinement is known as an effective procedure for improving the accuracy of a solution obtained by a method that is not backwards stable. We will now consider how to use it in the context of our normal equations and augmented system approaches.

[3] The bound assumes that there are no errors in the formation of AA^T and Ar^+ or in the backsolves using the Cholesky factors; this is a reasonable assumption in our context [29, section 19.4] provided that $\epsilon_m\kappa^2(A)$ is somewhat smaller than 1. It also ignores less significant errors that arise in the computation of the matrix-vector product A^Tv^+ and in the subtraction $r^+ - A^Tv^+$ given in (3.5).

5.1. Normal equations approach. Let us suppose that we choose $G = I$ and that we compute the projection $P_A r^+$ via the normal equations approach (3.5)–(3.6). An appealing idea for trying to improve the accuracy of this computation is to apply the projection repeatedly. Therefore, rather than computing $g^+ = P_A r^+$ in (2.16), we let $g^+ = P_A \cdots P_A r^+$, where the projection is applied as many times as necessary to keep the errors small. The motivation for this *multiple projections technique* stems from the fact that the computed projection $g_c^+ = (P_A r^+)_c$ is likely to have only a small component, consisting almost entirely of rounding errors, outside of the null space of A. Therefore, applying the projection P_A to the first projection g_c^+ will give an improved estimate because the ratio (4.2) will now be much smaller. By repeating this process we may hope to obtain further improvement of accuracy.

The multiple projection technique may simply be described as setting $g_0^+ = r^+$ and applying the following algorithm.

ALGORITHM 5.1 (multiple projections—normal equations). *Set $i = 0$ and repeat the following steps until a convergence test is satisfied:*

$$\begin{aligned} &\textit{solve} \quad L(L^T v_i^+) = A g_i^+, && (5.1)\\ &\textit{set} \quad g_{i+1}^+ = g_i^+ - A^T v_i^+, && (5.2)\\ &\qquad\ \ i \leftarrow i+1, \end{aligned}$$

where L is the Cholesky factor of AA^T.

We note that this method is only appropriate when $G = I$, although a simple variant is possible when G is diagonal. Also note that the multiple projection technique is equivalent to performing fixed-precision iterative refinement on the normal equations. In the multiple projections approach, the projection g^+ is updated at each iteration. In fixed-precision iterative refinement of the normal equations, the solution of the normal equations v^+ is updated and the projection g^+ is recomputed from this solution.

We resolved the problem given in Example 3.1 using multiple projections and setting $G = I$. At every CG iteration, we measured the cosine (3.13) of the angle between g and the columns of A. If this cosine was greater than 10^{-12}, multiple projections were applied until the cosine was smaller than this value. Using this strategy, we were able to reduce the norm of the null space component of the residual to around 10^{-16} of its initial value.

In the optimization setting we would apply multiple corrections only when needed, e.g., when the angle between the projected residual and the columns of A is not very small; see Algorithm 6.2 in section 6.1.

It is straightforward to analyze the multiple projections strategy (5.1)–(5.2) provided that, as before, we make the simplifying assumptions that A has norm one and that the only rounding errors we make are in forming L and solving (5.1). In this case, we have that

$$\|(g_{i+1}^+)_c - g^+\| \le \|\Delta v_i^+\| \le \left(\gamma \epsilon_m \kappa^2(A)\right)^i \|v^+\|, \tag{5.3}$$

and thus that the error converges R-linearly to zero with constant $\gamma \epsilon_m \kappa^2(A)$, so long as this factor is less than 1. Of course, the reduction in error at this rate cannot be sustained indefinitely, as the other errors we have ignored in (5.1)–(5.2) become important. Nonetheless, one would expect (5.3) to reflect the true behavior until $\|(g_{i+1}^+)_c - g^+\|$ approaches a small multiple of the unit roundoff ϵ_m. It should be stressed, however, that this approach is still limited by the fact that the condition number of A appears squared in (5.3); improvement can be guaranteed only if $\gamma\epsilon_m\kappa^2(A) < 1$.

We should also note that multiple projections are almost identical in their form and numerical properties to *fixed precision iterative refinement to the least squares problem* [5, p. 125]. Since a perturbation analysis of the least squares problem [5, Theorem 1.4.6] gives

$$\|g^+ - g_c^+\| = O\left(\epsilon_m(\|v\| + \kappa(A)\|g^+\|)\right), \tag{5.4}$$

and as the dependence here on the condition number is linear—not quadratic as we have seen for (4.1)—we may deduce that the normal equations approach is not backward stable [5, section 2.2]. Indeed, since $\kappa(A)$ is multiplied by $\|g^+\|$, when g^+ is small the effect of the condition number of A is much smaller in (5.4) than in (4.1). It is precisely under such circumstances that fixed-precision iterative refinement is most appropriate [5, section 2.9.3].

We should mention two other iterative refinement techniques that one might consider which are either not effective or not practical in our context.

The first is to use fixed-precision iterative refinement [5, section 2.9] to attempt to improve the solution v^+ of the normal equations (3.6). This, however, will generally be unsuccessful because fixed-precision iterative refinement improves only a measure of backward stability [22, p. 126], and the Cholesky factorization is already a backward stable method. We have performed numerical tests and found no improvement from this strategy.

However, as is well known, iterative refinement will often succeed if extended precision is used to evaluate the residuals. We could therefore consider using extended precision iterative refinement to improve the solution v^+ of the normal equations (3.6). So long as $\epsilon_m\kappa(A)^2 < 1$, and the residuals of (3.6) are smaller than one in norm, we can expect that the error in the solution of (3.6) will decrease by a factor $\epsilon_m\kappa(A)^2$ until it reaches $O(\epsilon_m)$. However, since optimization algorithms normally use double precision arithmetic for all their computations, extending the precision may not be simple or efficient, and this strategy is not suitable for general purpose software.

For the same reason we will not consider the use of extended precision in (5.1)–(5.2) or in the iterative refinement of the least squares problem.

5.2. Augmented system approach. We can apply fixed precision iterative refinement to the solution obtained from the augmented system (3.11). This gives the following iteration.

ALGORITHM 5.2 (iterative refinement—augmented system). *Repeat the following steps until a convergence test is satisfied.*

$$\begin{aligned} \textit{Compute}\ \ &\rho_g = r^+ - Gg^+ - A^Tv^+ \ \ \textit{and}\ \ \rho_v = -Ag^+,\\ \textit{solve}\ \ &\begin{pmatrix} G & A^T \\ A & 0 \end{pmatrix}\begin{pmatrix} \Delta g^+ \\ \Delta v^+ \end{pmatrix} = \begin{pmatrix} \rho_g \\ \rho_v \end{pmatrix},\\ \textit{and update}\ \ &g^+ \leftarrow g^+ + \Delta g^+ \ \ \textit{and}\ \ v^+ \leftarrow v^+ + \Delta v^+. \end{aligned}$$

Note that this method is applicable for general preconditioners G. The general analysis of Higham [30, Theorem 3.2] indicates that, if the condition number of A is not too large, we can expect high relative accuracy in v^+ and good absolute accuracy in g^+ in most cases.

We solved the problem given in Example 3.1 using this iterative refinement technique. As before, we measured the angle between g and the columns of A at every CG iteration. Iterative refinement was applied so long as the cosine of this angle was greater than 10^{-12}. We observed that $\sqrt{r^T g}$ decreased almost as much as with the multiple projections approach.

In our experience, one iterative refinement step is normally enough to provide good accuracy, but we have encountered cases in which two or three steps are beneficial. As in the case of the multiple projections using the normal equations, we would apply this refinement technique selectively in optimization algorithms.

6. Residual update strategy. We have seen that significant roundoff errors may occur in the computation of the projected residual g^+ if this vector is much smaller than the residual r^+. As discussed in the paragraph preceding Example 3.1, the reason for this error is cancellation. We now describe a procedure for redefining r^+ so that its norm is closer to that of g^+. This will dramatically reduce the roundoff errors in the projection operation and thus nearly eliminate the need to use iterative refinement.

We begin by noting that the iterates x of Algorithm 2.2 are theoretically unaffected if, immediately after computing r^+ in (2.15), we redefine it as

$$r^+ \leftarrow r^+ - A^T y \tag{6.1}$$

for some $y \in \mathbf{R}^m$. This equivalence is due to the fact that r^+ appears only in (2.16) and (2.17) and that we have both $PA^Ty = 0$, and $(g^+)^T A^T y = 0$. It follows that we can redefine r^+ by means of (6.1) in either the normal equations approach (3.4) and (3.9) or in the augmented system approach (3.8) and (3.11), and the results would, in theory, be unaffected.

Having this freedom to redefine r^+, we seek the value of y that minimizes

$$\|r^+ - A^T y\|_{G^{-1}}, \tag{6.2}$$

where $\|\cdot\|_{G^{-1}}$ is the dual (semi-) norm to the norm $s^T G s$ defined on the manifold $As = 0$, and where we require that G is positive definite over this manifold (see [14]). This dual norm is convenient, since the vector y that solves (6.2) is precisely $y = v^+$ from (3.11). This gives rise to the following modification of the CG iteration.

ALGORITHM 6.1 (preconditioned CG with residual update). *Choose an initial point x satisfying $Ax = b$ and compute $r = Hx + c$. Find the vector y that minimizes $\|r - A^T y\|_{G^{-1}}$; this can be done by solving* (6.2) *and setting $y \leftarrow v^+$. Set $r \leftarrow r - A^T y$, compute $g = Pr$, and set $p = -g$. Repeat the following steps until a convergence test is satisfied:*

$$\alpha = r^T g / p^T H p, \tag{6.3}$$

$$x \leftarrow x + \alpha p, \tag{6.4}$$

$$r^+ = r + \alpha H p, \tag{6.5}$$

$$\textit{compute } y \textit{ that minimizes } (6.2), \tag{6.6}$$

$$r^+ \leftarrow r^+ - A^T y, \tag{6.7}$$

$$g^+ = P r^+, \tag{6.8}$$

$$\beta = (r^+)^T g^+ / r^T g, \tag{6.9}$$

$$p \leftarrow -g^+ + \beta p, \tag{6.10}$$

$$g \leftarrow g^+ \quad \textit{and} \quad r \leftarrow r^+. \tag{6.11}$$

This procedure can be improved by adding iterative refinement of the projection operation in (6.8). In this case, at most one or two iterative refinement steps should be used. The added cost of this algorithm is the storage and computation of y each iteration.

Notice that there is a simple interpretation of steps (6.6)–(6.8). We first obtain y by solving (6.2), and as we have indicated the required value is $y = v^+$ from (3.11). However, (3.11) may be rewritten as

$$\begin{pmatrix} G & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} g^+ \\ 0 \end{pmatrix} = \begin{pmatrix} r^+ - A^T v^+ \\ 0 \end{pmatrix}, \tag{6.12}$$

and thus when we obtain g^+ in step (6.8), it is as if we had instead found it by solving

$$\begin{pmatrix} G & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} g^+ \\ u^+ \end{pmatrix} = \begin{pmatrix} r^+ - A^T v^+ \\ 0 \end{pmatrix}. \tag{6.13}$$

Comparing (6.12) and (6.13), it follows that $u^+ = 0$ in exact arithmetic, although all we can expect in floating point arithmetic is that the computed u^+ will be very small, provided of course that (6.13) is solved in a stable fashion. The advantage of using (6.13) compared to (3.11) is that the solution in the latter may be dominated by the large components v^+, while in the former g^+ are the (relatively) large components, and thus we can expect to find them with high relative accuracy if (6.13) is solved in a stable fashion. Viewed in this way, we see that steps (6.6)–(6.8) are actually a limited form of iterative refinement in which the computed v^+, but not the computed g^+ which is discarded, is used to refine the solution. This "iterative semirefinement" has been used in other contexts [8, 23].

There is another interesting interpretation of the reset $r \leftarrow r - A^T y$ performed at the start of Algorithm 6.1. In the parlance of optimization, $r = Hx + c$ is the gradient of the objective function (1.1) and $r - A^T y$ is the gradient of the Lagrangian for the problem (1.1)–(1.2). The vector y computed from (6.2) is called the least squares Lagrange multiplier estimate. (It is common, but not always the case, for optimization algorithms to set $G = I$ in (6.2) to compute these multipliers.) Thus in Algorithm 6.1 we propose that the initial residual be set to the current value of the gradient of the Lagrangian, as opposed to the gradient of the objective function.

One could ask whether it is sufficient to do this resetting of r at the beginning of Algorithm 6.1 and omit steps (6.6)–(6.7) in subsequent iterations. Our computational experience shows that, even though this initial resetting of r causes the first few CG iterations to take place without significant errors, deviations from the null space due to rounding errors arise in subsequent iterations. The strategy proposed in Algorithm 6.1 is safe in that it ensures that r is small at every iteration.

As it stands, Algorithm 6.1 would appear to require two products with P, or, at the very least, one with P to perform (6.8) and some other means, such as (3.6), to determine y. As we shall now see, this need not be the case.

6.1. The case $G = I$.

There is a particularly efficient implementation of the residual update strategy when $G = I$. We can redefine r without the extra cost of storing and computing y as required by Algorithm 6.1. In Algorithm 6.2 below, we present the residual update for $G = I$, combined with iterative refinement. It is this algorithm that is used in the numerical tests in section 7.

Algorithm 6.2 (residual update and iterative refinement for $G = I$). *Choose an initial point x satisfying $Ax = b$, compute $r = Hx + c$, $r \leftarrow Pr$, $g \leftarrow Pr$, where* *the projection is computed by the normal equations* (3.4) *or augmented system* (3.8) *approaches, and set* $p = -g$. *Choose a tolerance* $\theta_{\max}$. *Repeat the following steps until a convergence test is satisfied:*

$$\alpha = r^T g / p^T Hp, \tag{6.14}$$

$$x \leftarrow x + \alpha p, \tag{6.15}$$

$$r^+ = r + \alpha Hp, \tag{6.16}$$

$$g^+ = Pr^+, \tag{6.17}$$

apply iterative refinement to Pr^+*, if necessary,*
until (3.13) *is less than* $\theta_{\max}$,

$$\beta = (r^+)^T g^+ / r^T g, \tag{6.18}$$

$$p \leftarrow -g^+ + \beta p, \tag{6.19}$$

$$g \leftarrow g^+ \quad \textit{and} \quad r \;\leftarrow\; g^+. \tag{6.20}$$

This algorithm was derived from noting that (6.2) is precisely the objective of the least squares problem (3.7) that occurs when computing Pr^+ via the normal equations approach, and therefore the desired value of y is nothing other than the vector v^+ in (3.6) or (3.8). Furthermore, the first block of equations in (3.8) shows that $r^+ - A^T v^+ = g^+$. Therefore, when $G = I$ the computation (6.7) can be replaced by $r^+ \leftarrow Pr^+$ and (6.8) is $g^+ = Pr^+$. In other words, we have applied the projection operation twice, and this is a special case of the multiple projections approach described in the previous section.

Further, (6.7) can be written as $r^+ \leftarrow Pr^+$, or $r^+ = Pr + PH\alpha p$, and therefore (6.8) is

$$g^+ = P(Pr + PH\alpha p). \tag{6.21}$$

As the CG iteration progresses we can expect αp, but not r, to become small. Therefore, we will apply the projection twice to r but only once to $H\alpha p$. Thus (6.21) is replaced by

$$g^+ = P(Pr + H\alpha p), \tag{6.22}$$

which is mathematically equivalent to (6.21), since $PP = P$. This expression is convenient because the term Pr was computed at the previous CG iteration, and therefore we can obtain (6.22) by simply setting $r \leftarrow g^+$ in (6.11) instead of $r \leftarrow r^+$.

Also note that the numerator in the definition (6.3) of α now becomes $g^T g$, which equals $r^T Pg = r^T g$. Thus the formula for α is theoretically the same as in Algorithm 6.1, but the symmetric form $\alpha = g^T g / p^T Hp$ has the advantage that its numerator can never be negative, as is the case with (6.3) when rounding errors dominate the projection operation.

We solved the problem given in Example 3.1 using this residual update strategy with $G = I$. Both the normal equations and augmented system approaches were equally effective in this case. The cosine (3.13) of the angle between the preconditioned residual and the columns of A remained very small as the computation proceeded. For the normal equations approach this cosine was of order 10^{-14} throughout the CG iteration; for the augmented system approach it was of order 10^{-15}. We also noted that we were able to obtain higher accuracy than with the iterative refinement strategies described in the previous section.

6.2. General *G*. We can also improve upon the efficiency of Algorithm 6.1 for general G using slightly outdated information. The idea is simply to use the v^+ obtained when computing g^+ in (6.8) as a suitable y rather than waiting until after the following step (6.5) to obtain a slightly more up-to-date version. The resulting iteration is as follows.

Algorithm 6.3 (residual update strategy for general G). *Apply Algorithm* 6.1 *with the following two changes:*

omit (6.6)–(6.7),

replace (6.11) *by* $g \leftarrow g^+$ *and* $r \leftarrow r^+ - A^T v^+$, *where* v^+ *is obtained as a by-product when using* (3.11) *to compute* (6.8).

Thus a single projection in step (6.8) is needed for each iteration. Notice, however, that for general G, the extra matrix-vector product $A^T v^+$ will be required, since we no longer have the relationship $g^+ = r^+ - A^T v^+$ that we exploited when $G = I$. Although we have not experimented on this idea for this paper, it has proved to be beneficial in other similar circumstances [23] and provides the backbone for the developing HSL [32] nonconvex quadratic programming packages HSL_VE12 [14] (interior-point) and HSL_VE19 [25] (active set). See also [34] for a thorough discussion of existing and new preconditioners along these lines and the results of some comparative testing.

7. Numerical results. We now test the efficacy of the techniques proposed in this paper on a collection of quadratic programs of the form (1.1)–(1.2). The problems were generated during the last iteration of the interior point method for nonlinear programming described in [9] when this method was applied to a set of test problems from the CUTE [6] collection. We apply the CG method without preconditioning, i.e., with $G = I$, to solve these quadratic programs.

We use the augmented system and normal equations approaches to compute projections, and for each we compare the standard CG iteration (stand), given by Algorithm 2.2, with the iterative refinement (ir) techniques described in section 5 and the residual update strategy combined with iterative refinement (update) as given in Algorithm 6.2. The results are given in Table 7.1. The first column gives the problem name and the second gives the dimension of the quadratic program. To test the reliability of the techniques proposed in this paper we used a very demanding stopping test: the CG iteration was terminated when $\sqrt{r^T g} \leq 10^{-12}$. This stopping test would not be used in practice; rather, we wanted to observe the level of accuracy that could be achieved with each approach.

In these experiments we included several other stopping tests in the CG iteration that are typically used by trust region methods for optimization. We terminate if the number of iterations exceeds $2(n-m)$, where $n-m$ denotes the dimension of the reduced system (2.4); a superscript ^{1} in Table 7.1 indicates that this limit was reached. The CG iteration was also stopped if the length of the solution vector is greater than a "trust region radius" that is set by the optimization method (see [9]). We use a superscript ^{2} to indicate that this safeguard was activated, and note that in these problems only excessive rounding errors can trigger it. Finally we terminate if $p^T H p < 0$, indicated by ^{3}, or if significant rounding error resulted in $r^T g < 0$, indicated by ^{4}. The presence of any superscript indicates that the residual test $\sqrt{r^T g} \leq 10^{-12}$ was not met. Note that the standard CG iteration was not able to meet the residual stopping test for any of the problems in Table 7.1 but that iterative refinement and update residual were successful in most cases.

Table 7.2 reports the CPU time for the problems in Table 7.1. Note that the times for the standard CG approach (stand) should be interpreted with caution, since

Table 7.1

Number of CG iterations for the different approaches. [1] *indicates that the iteration limit was reached,* [2] *indicates termination from trust region bound,* [3] *indicates negative curvature was detected, and* [4] *indicates that* $r^T g < 0$.

		Augmented system			Normal equations		
Problem	dim	stand	ir	update	stand	ir	update
CORKSCRW	147	16^2	9	10	4^4	9	11
COSHFUN	61	124^1	124^1	58	124^1	124^1	55
DIXCHLNV	50	91	12	12	5^4	12	12
DTOC3	999	18^4	6	6	2000^1	6	6
DTOC6	1000	6^4	16	16	2^4	16	16
HAGER4	1000	193^4	350	348	1057^4	351	349
HIMMELBK	10	22^1	3	3	7^4	3	3
NGONE	97	0^4	67	56	0^4	65	60
OPTCNTRL	9	20^4	12	4	20^1	2	5
OPTCTRL6	39	90^1	80^1	16	80^1	80^1	16
OPTMASS	402	0^4	5	6	9^3	5	5
ORTHREGA	261	13^4	16^3	16^3	14^3	16^3	16^3
ORTHREGF	805	8^4	18	18	7^4	18	18
READING1	101	3^4	5	5	3^4	5	5

Table 7.2

CPU time in seconds. [1] *indicates that the iteration limit was reached,* [2] *indicates termination from trust region bound,* [3] *indicates negative curvature was detected, and* [4] *indicated that* $r^T g < 0$.

		Augmented system			Normal equations		
Problem	dim	stand	ir	update	stand	ir	update
CORKSCRW	147	0.85^2	1.18	0.88	0.15^4	0.74	0.70
COSHFUN	61	0.37^1	0.66^1	0.18	0.29^1	0.54^1	0.13
DIXCHLNV	50	1.90	0.49	0.30	0.2^4	0.50	0.30
DTOC3	999	0.48^4	0.9	0.60	148.48^1	0.91	0.47
DTOC6	1000	0.32^4	1.51	0.9	0.08^4	1.16	0.66
HAGER4	1000	14.23^4	54.43	34.30	70.57^4	40.48	24.71
HIMMELBK	10	0.13^1	0.07	0.04	0.03^4	0.05	0.04
NGONE	97	0.16^4	21.19	10.69	0.98^4	125.24	77.35
OPTCNTRL	9	0.06^4	0.20	0.06	0.05^1	0.28	0.07
OPTCTRL6	39	0.36^1	0.65^1	0.08	0.29^1	0.45^1	0.06
OPTMASS	402	0.06^4	0.57	0.43	0.34^3	0.38	0.25
ORTHREGA	261	0.98^4	2.02^3	1.14^3	0.91^3	2.52^3	1.88^3
ORTHREGF	805	0.46^4	1.84	1.06	1.14^4	5.65	2.95
READING1	101	0.24^4	0.92	0.40	0.29^4	1.31	0.85

in some of these problems it terminated prematurely. We include the times for this standard CG iteration only to show that the iterative refinement and residual update strategies do not greatly increase the cost of the CG iteration.

Next we report on three problems for which the stopping test $\sqrt{r^T g} \leq 10^{-12}$ could not be met by any of the variants. For these three problems, Table 7.3 provides the least residual norm attained for each strategy.

As a final but indirect test of the techniques proposed in this paper, we report the results obtained with KNITRO (an interior point nonlinear optimization code described in [9]) on 29 nonlinear programming problems from the CUTE collection. This code applies the projected CG method to solve a quadratic program at each iteration. The CG iteration was terminated when $\sqrt{r^T g} \leq 0.1\sqrt{r_0^T g_0}$, which is much less stringent than the termination tests used above. We used the augmented system and normal equations approaches to compute projections, and for each of these strategies we tried the standard CG iteration (stand) and the residual update strategy (update)

Table 7.3

The least residual norm $\sqrt{r^T g}$ attained by each option.

		Augmented system			Normal equations		
Problem	dim	stand	ir	update	stand	ir	update
OBSTCLAE	900	2.3D-07	1.5D-07	5.5D-08	2.3D-07	9.9D-08	4.2D-08
SVANBERG	500	1.8D-07	9.9D-10	5.7D-12	7.7D-08	8.8D-10	2.9D-10
TORSION1	400	3.5D-09	3.5D-09	2.8D-09	5.5D-08	4.6D-08	3.2D-09

Table 7.4

Number of function evaluations and projections required by the optimization method for different variants of the CG iteration. n denotes the number of variables, m the number of general constraints (equalities or inequalities), excluding simple bounds, "st" is the standard CG method, and "up" includes residual updates.

			Augmented system				Normal equations			
			f evals		projections		f evals		projections	
Problem	n	m	st	up	st	up	st	up	st	up
CORKSCRW	456	350	64	61	458	422	65	61	460	411
COSHFUN	61	20	44	40	2213	1025	49	40	2998	1025
DIXCHLNV	100	50	19	19	83	83	19	19	83	83
GAUSSELM	14	11	25	26	92	93	28	41	85	97
HAGER4	2001	1000	18	18	281	281	50	18	2458	281
HIMMELBK	24	14	33	33	88	89	39	33	135	89
NGONE	100	1273	216	133	1763	864	217	187	1821	1146
OBSTCLAE	1024	0	26	26	6233	6068	26	26	6236	6080
OPTCNTRL	32	20	41	51	152	183	***	50	***	179
OPTMASS	1210	1005	36	39	129	145	218	39	427	145
ORTHREGF	1205	400	30	30	73	73	30	30	73	73
READING1	202	100	40	40	130	130	43	40	151	130
SVANBERG	500	500	35	35	7809	4265	40	35	10394	4764
TORSION1	484	0	19	19	2174	2140	19	19	2449	2120
DTOC2	2998	1996	6	6	215	215	6	6	215	215
DTOC3	2999	1998	7	7	16	16	26	7	73	16
DTOC4	2999	1998	5	5	8	8	5	5	8	8
DTOC5	1999	999	6	6	12	12	6	6	12	12
DTOC6	2001	1000	12	12	48	46	64	12	166	46
EIGENA2	110	55	4	4	4	4	4	4	4	4
EIGENC2	464	231	25	25	264	268	25	25	270	269
GENHS28	300	298	4	4	7	7	4	4	7	7
HAGER2	2001	1000	5	5	12	12	5	5	12	12
HAGER3	1001	500	4	4	9	9	4	4	9	9
OPTCTRL6	122	80	14	10	97	75	75	10	880	75
ORTHREGA	517	256	8	8	38	38	***	48	***	99
ORTHREGC	505	250	10	10	60	60	10	10	60	60
ORTHREGD	203	100	11	11	23	23	11	11	23	23

with iterative refinement described in Algorithm 6.2. Now we were concerned with reducing feasibility errors in the CG iterates, not to be able to satisfy a stringent CG termination test, but to ensure that the outer optimization algorithm would converge. The results are given in Table 7.4, where "fevals" denotes the total number of evaluations of the objective function of the nonlinear problem, and "projections" represents the total number of times that a projection operation was performed during the optimization. A *** indicates that the optimization algorithm was unable to locate the solution.

Note that the total number of function evaluations is roughly the same for all strategies, but there are a few cases where the differences in the CG iteration cause the algorithm to follow a different path to the solution. This is to be expected when solving nonlinear problems. Note that for the augmented system approach, the residual update strategy changes the number of projections significantly only in a few problems, but when it does the improvements are very substantial. On the other hand, we observe that for the normal equations approach (which is more sensitive to the condition number $\kappa(A)$) the residual update strategy gives a substantial reduction in the number of projections in about half of the problems. It is interesting that with the residual update, the performance of the augmented system and normal equations approaches is very similar.

8. Conclusions. We have studied the properties of the projected CG method for solving quadratic programming problems of the form (1.1)–(1.2). Due to the form of the preconditioners used by some nonlinear programming algorithms we opted for not computing a basis Z for the null space of the constraints but instead projecting the CG iterates using a normal equations or augmented system approach. We have given examples showing that in either case significant roundoff errors can occur and have presented an explanation for this.

We proposed several remedies. One is to use iterative refinement of the augmented system or normal equations approaches. An alternative is to update the residual at every iteration of the CG iteration, as described in section 6. The latter can be implemented particularly efficiently in the unpreconditioned ($G = I$) case.

Our numerical experience indicates that updating the residual almost always suffices to keep the errors to a tolerable level. Iterative refinement techniques are not as effective by themselves as the update of the residual but can be used in conjunction with it, and the numerical results reported in this paper indicate that this combined strategy is both economical and accurate. The techniques described here are important ingredients within the evolving large scale nonlinear programming packages KNITRO and GALAHAD, as well as the HSL [32] QP modules HSL_VE12 and HSL_VE19.

Acknowledgments. The authors would like to thank Andy Conn and Philippe Toint for their helpful input during the early stages of this research. They are also grateful to Margaret Wright and two anonymous referees for their helpful suggestions.

REFERENCES

[1] M. Arioli, J.W. Demmel, and I.S. Duff, *Solving sparse linear systems with sparse backward errors*, SIAM J. Matrix Anal. Appl., 10 (1989), pp. 165–190.

[2] M. Arioli, I.S. Duff, and P.P.M. de Rijk, *On the augmented system approach to sparse least-squares problems*, Numer. Math., 55 (1989), pp. 667–684.

[3] O. Axelsson. *Iterative Solution Methods*, Cambridge University Press, Cambridge, UK, 1996.

[4] Å. Björck, *Pivoting and stability in augmented systems*, in Numerical Analysis 1991, D.F. Griffiths and G.A. Watson, eds., Pitman Res. Notes Math. Ser. 260, Longman Scientific and Technical, Harlow, UK, 1992, pp. 1–16.

[5] Å. Björck, *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, 1996.

[6] I. Bongartz, A.R. Conn, N.I.M. Gould, and Ph. L. Toint, *CUTE: Constrained and unconstrained testing environment*, ACM Trans. Math. Software, 21 (1995), pp. 123–160.

[7] J.R. Bunch and L.C. Kaufman, *Some stable methods for calculating inertia and solving symmetric linear equations*, Math. Comput., 31 (1977), pp. 163–179.

[8] P. Businger and G.H. Golub, *Linear least squares solutions by Housholder transformations*, Numer. Math., 7 (1965), pp. 269–276.

[9] R.H. Byrd, M.E. Hribar, and J. Nocedal, *An interior point algorithm for large-scale nonlinear programming*, SIAM J. Optim., 9 (1999), pp. 877–900.

[10] T.F. Coleman, *Linearly constrained optimization and projected preconditioned conjugate gradients*, in Proceedings of the Fifth SIAM Conference on Applied Linear Algebra, J. Lewis, ed., SIAM, Philadelphia, 1994, pp. 118–122.

[11] T.F. Coleman and A. Pothen, *The null space problem* I: *Complexity*, SIAM J. Algebraic Discrete Methods, 7 (1986), pp. 527–537.

[12] T.F. Coleman and A. Pothen, *The null space problem* II: *Algorithms*, SIAM J. Algebraic Discrete Methods, 8 (1987), pp. 544–563.

[13] T.F. Coleman and A. Verma, *A Preconditioned Conjugate Gradient Approach to Linear Equality Constrained Minimization*, Technical report, Department of Computer Sciences, Cornell University, Ithaca, NY, 1998.

[14] A.R. Conn, N.I.M. Gould, D. Orban, and Ph. L. Toint, *A primal-dual trust-region algorithm for non-convex nonlinear programming*, Math. Program., 87 (2000), pp. 215–249.

[15] J.E. Dennis Jr., M. El-Alem, and M.C. Maciel, *A global convergence theory for general trust-region based algorithms for equality constrained optimization*, SIAM J. Optim., 7 (1997), pp. 177–207.

[16] J.E. Dennis, M. Heinkenschloss, and L.N. Vicente, *Trust-region interior-point SQP algorithms for a class of nonlinear programming problems*, SIAM J. Control Optim., 36 (1998), pp. 1750–1794.

[17] I.S. Duff and J.K. Reid, *The multifrontal solution of indefinite sparse symmetric linear equations*, ACM Trans. Math. Software, 9 (1983), pp. 302–325.

[18] J.C. Dunn, *Second-order multiplier update calculations for optimal control problems and related large scale nonlinear programs*, SIAM J. Optim., 3 (1993), pp. 489–502.

[19] J.R. Gilbert and M.T. Heath, *Computing a sparse basis for the null-space*, SIAM J. Algebraic Discrete Methods, 8 (1987), pp. 446–459.

[20] P.E. Gill, W. Murray, M.A. Saunders, and M.H. Wright, *Maintaining LU factors of a general sparse matrix*, Linear Algebra Appl., 88/89 (1987), pp. 239–270.

[21] P.E. Gill, W. Murray, and M.H. Wright, *Practical Optimization*, Academic Press, London, 1981.

[22] G.H. Golub and C.F. Van Loan, *Matrix Computations*, 3rd ed., Johns Hopkins University Press, Baltimore, MD, 1996.

[23] N.I.M. Gould, *Iterative methods for ill-conditioned linear systems from optimization*, in Nonlinear Optimization and Related Topics, G. Di Pillo and F. Giannessi, eds., Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999, pp. 123–142.

[24] N.I.M. Gould, S. Lucidi, M. Roma, and Ph. L. Toint, *Solving the trust-region subproblem using the Lanczos method*, SIAM J. Optim., 9 (1999), pp. 504–525.

[25] N.I.M. Gould and Ph. L. Toint, *An Iterative Active-Set Method for Large-Scale Quadratic Programming*, Technical report RAL-TR-2001-026, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 2001.

[26] M.T. Heath, R.J. Plemmons, and R.C. Ward, *Sparse orthogonal schemes for structural optimization using the force method*, SIAM J. Sci. Statist. Comput., 5 (1984), pp. 514–532.

[27] M. Heinkenschloss and L.N. Vicente, *Analysis of Inexact Trust Region Interior-Point SQP Algorithms*, Technical report CRPC-TR95546, Center for Research on Parallel Computers, Houston, TX, 1995.

[28] M.R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Res. Nat. Bur. Standards, 49 (1952), pp. 409–436.

[29] N.J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia, 1996.

[30] N.J. Higham, *Iterative refinement for linear systems and LAPACK*, IMA J. Numer. Anal., 17 (1997), pp. 495–505.

[31] M.E. Hribar, *Large-Scale Constrained Optimization*, Ph.D. thesis, Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL, 1996.

[32] HSL, *A collection of Fortran codes for large scale scientific computation*, 2000.

[33] D. James, *Implicit nullspace iterative methods for constrained least squares problems*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 962–978.

[34] C. Keller, *Constraint Preconditioning for Indefinite Linear Systems*, D.Phil. thesis, Oxford University, Oxford, UK, 2000.

[35] C. Keller, N.I.M. Gould, and A.J. Wathen, *Constraint preconditioning for indefinite linear systems*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1300–1317.

[36] M. Lalee, J. Nocedal, and T.D. Plantenga, *On the implementation of an algorithm for large-scale equality constrained optimization*, SIAM J. Optim., 8 (1998), pp. 682–706.

[37] L. Lukšan and J. Vlček, *Indefinitely preconditioned inexact Newton method for large sparse equality constrained nonlinear programming problems*, Numer. Linear Algebra Appl., 5 (1998), pp. 219–247.

[38] S.G. Nash and A. Sofer, *Preconditioning reduced matrices*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 47–68.

[39] J. Nocedal and S.J. Wright, *Numerical Optimization*, Springer-Verlag, Heidelberg, Berlin, New York, 1999.

[40] T.D. Plantenga, *A trust region method for nonlinear programming based on primal interior-point techniques*, SIAM J. Sci. Comput., 20 (1999), pp. 282–305.

[41] R.J. Plemmons and R.E. White, *Substructuring methods for computing the null space of equilibrium matrices*, SIAM J. Matrix Anal. Appl., 11 (1990), pp. 1–22.

[42] B.T. Polyak, *The conjugate gradient method in extremal problems*, U.S.S.R. Comput. Math. Math. Phys., 9 (1969), pp. 94–112.

[43] T. Steihaug, *The conjugate gradient method and trust regions in large scale optimization*, SIAM J. Numer. Anal., 20 (1983), pp. 626–637.

[44] J.M. Stern and S.A. Vavasis, *Nested dissection for sparse nullspace bases*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 766–775.

[45] Ph. L. Toint, *Towards an efficient sparsity exploiting Newton method for minimization*, in Sparse Matrices and Their Uses, I.S. Duff, ed., Academic Press, London, 1981, pp. 57–88.

[46] Ph. L. Toint and D. Tuyttens, *On large-scale nonlinear network optimization*, Math. Program. Ser. B, 48 (1990), pp. 125–159.

[47] L.N. Vicente, *Trust-Region Interior-Point Algorithms for a Class of Nonlinear Programming Problems*, Ph.D. thesis, Report TR96-05, Department of Computational and Applied Mathematics, Rice University, Houston, TX, 1995.

SIAM J. SCI. COMPUT.
Vol. 23, No. 4, pp. 1396–1417

ROBUST PARALLEL SMOOTHING FOR MULTIGRID VIA SPARSE APPROXIMATE INVERSES[*]

OLIVER BRÖKER[†], MARCUS J. GROTE[‡], CARSTEN MAYER[§], AND ARNOLD REUSKEN[¶]

Abstract. Sparse approximate inverses are considered as smoothers for multigrid. They are based on the SPAI-Algorithm [M. J. Grote and T. Huckle, *SIAM J. Sci. Comput.*, 18 (1997), pp. 838–853], which constructs a sparse approximate inverse M of a matrix A by minimizing $I - MA$ in the Frobenius norm. This yields a new hierarchy of smoothers: SPAI-0, SPAI-1, SPAI(ε). Advantages of SPAI smoothers over classical smoothers are inherent parallelism, possible local adaptivity, and improved robustness. The simplest smoother, SPAI-0, is based on a diagonal matrix M. It is shown to satisfy the smoothing property for symmetric positive definite problems. Numerical experiments show that SPAI-0 smoothing is usually preferable to damped Jacobi smoothing. For the SPAI-1 smoother the sparsity pattern of M is that of A; its performance is typically comparable to that of Gauss–Seidel smoothing; however, both the computation and the application of the smoother remain inherently parallel. In more difficult situations, where the simpler SPAI-0 and SPAI-1 smoothers are not adequate, the SPAI(ε) smoother provides a natural procedure for improvement where needed. Numerical examples illustrate the usefulness of SPAI smoothing.

Key words. multilevel methods, approximate inverses, smoothing property, parallel preconditioning, iterative methods, sparse linear systems

AMS subject classifications. 65F10, 65F50, 65N55

PII. S1064827500380623

1. Introduction.

Multigrid methods are efficient iterative solvers for large linear systems of equations, which result from the discretization of partial differential equations; see Brandt [8], Hackbusch [15, 16], Wesseling [27], and the references therein. They also yield efficient preconditioners when combined with Krylov subspace methods [7]. Any multigrid algorithm relies on the complementary interplay of smoothing and coarse grid correction. While the smoothing process aims at reducing the high-frequency error component, namely that which cannot be represented on coarser grids, the coarse grid correction solves for the low-frequency error component, precisely that which is well represented on the coarser grid. The careful combination of both smoothing and coarse grid correction yields a multigrid iteration, which has a high convergence rate independent of the mesh size.

Standard smoothing techniques typically result from the application of a few steps of a basic iterative method. Here we shall consider smoothers that are based on sparse approximate inverses. Starting from the linear system

$$Ax = b, \tag{1.1}$$

we denote by M a sparse approximation of A^{-1}. Then the corresponding basic iter-

[*] Received by the editors November 8, 2000; accepted for publication (in revised form) August 3, 2001; published electronically December 18, 2001.
http://www.siam.org/journals/sisc/23-4/38062.html

[†] Departement Informatik, ETH Zürich, CH-8092 Zürich, Switzerland (broeker@inf.ethz.ch).

[‡] Departement Mathematik, ETH Zürich, CH-8092 Zürich, Switzerland (grote@math.unibas.ch).

[§] Fachbereich Mathematik, Universität Kaiserslautern, D-67653 Kaiserslautern, Germany (cmayer@mathematik.uni-kl.de).

[¶] Institut für Geometrie und Praktische Mathematik, RWTH Aachen, D-52056 Aachen, Germany (reusken@igpm.rwth-aachen.de).

ative method is

$$x^{(k+1)} = x^{(k)} - M(Ax^{(k)} - b). \tag{1.2}$$

As the approximate inverse M is known explicitly, each iteration step requires only one additional $M \times v$ matrix-vector multiply; thus it is easy to parallelize and cheap to evaluate because M is sparse.

Recently, various algorithms have been proposed, all of which attempt to compute directly a sparse approximate inverse of A. Examples are the FSAI approach by Kolotilina and Yeremin [18], the MR algorithm by Chow and Saad [10], and the AINV approach by Benzi, Meyer, and Tůma [5]. Once computed, the approximate inverse M is applied as a preconditioner to the linear system (1.1) for use with a Krylov subspace iterative method. For a comparative study of various sparse approximate inverse preconditioners we refer to Benzi and Tuma [6]. By choosing an a priori sparsity pattern for M, the cost of computing M can be greatly reduced. Possible choices include powers of A or $A^\top A$, as suggested by Huckle [17] and Chow [11].

Approximate inverse techniques are also gaining in importance as smoothers for multigrid methods. First introduced by Benson and Frederickson [3] and Benson [4], they were shown to be effective on various difficult elliptic problems on unstructured grids by Tang and Wan [25]. Advantages of sparse approximate inverse smoothers over classical smoothers, such as damped Jacobi, Gauss–Seidel, or ILU, are inherent parallelism, possible local adaptivity, and improved robustness.

Here we shall consider sparse approximate inverse (SPAI) smoothers based on the SPAI-Algorithm by Grote and Huckle [14]. The SPAI-Algorithm computes an approximate inverse M explicitly by minimizing $I - MA$ in the Frobenius norm. Both the computation of M and its application as a smoother are inherently parallel. Since an effective sparsity pattern of M is in general unknown a priori, the SPAI-Algorithm attempts to determine the most promising entries dynamically. This strategy has proved effective in generating preconditioners for many difficult and ill-conditioned problems (see Barnard, Bernardo, and Simon [1], Tang [24], and Grote and Huckle [14]). Moreover, it provides the means for adjusting the smoother locally and automatically, if necessary.

We shall consider the following hierarchy of SPAI smoothers: SPAI-0, SPAI-1, and SPAI(ε). For SPAI-0 and SPAI-1 the sparsity pattern of M is fixed: M is diagonal for SPAI-0, whereas for SPAI-1 the sparsity pattern of M is that of A. For SPAI(ε) the sparsity pattern of M is determined automatically by the SPAI-Algorithm [14]; the parameter ε controls the accuracy and the amount of fill-in of M.

Besides the SPAI smoothing operators, all other multigrid components, such as the prolongation, the restriction, and the coarse grid operators, result from standard choices. It is well known that for certain classes of problems, such as convection-diffusion equations, a significant improvement in the efficiency of the multigrid solver can be obtained by using matrix-dependent prolongation and restriction operators (see [13, 20, 27, 30]). An interesting topic for future research is the combination of this new hierarchy of local and inherently parallel smoothers with algebraic multigrid techniques (see, for instance, [19, 21, 22, 26]).

In section 2 we briefly review the SPAI-Algorithm and show how sparse approximate inverses are used as smoothers in multigrid. In section 3 we prove that for SPAI-0 the smoothing property [16] holds under reasonable assumptions on the matrix A. More precisely, for A symmetric and positive definite, we prove that SPAI-0 satisfies the smoothing property, either if A is weakly diagonally dominant or if A has at most seven nonzero off-diagonal entries per row. To our knowledge this is the first fairly general theoretical result on the smoothing property of iterative methods that are based on sparse approximate inverses. Previously, Tang and Wan [25] analyzed the smoothing property of sparse approximate inverse smoothers for boundary value problems with constant coefficients on a two-dimensional regular grid. From a comparison of the SPAI-0 and damped Jacobi smoothers via numerical experiments, we conclude that the parameter-free SPAI-0 smoother is usually preferable to the damped Jacobi method. Finally, in section 4, we present an extensive set of numerical experiments, which demonstrate the usefulness of SPAI smoothing.

2. SPAI smoothing.

Starting from a standard multigrid setting, such as found in [15], [16, Chap. 10], or [27], we recall some basic notions and briefly introduce relevant notation. We assume the hierarchy of spaces,

$$X_\ell = \mathbb{R}^{n_\ell}, \qquad \ell = 0, 1, 2, \dots, \quad n_0 < n_1 < n_2 < \cdots,$$

together with the prolongation and restriction operators

$$p: \; X_{\ell-1} \to X_\ell, \qquad r: \; X_\ell \to X_{\ell-1}, \quad \ell = 1, 2, \dots .$$

To each space X_ℓ we associate a nonsingular operator,

$$A_\ell: \; X_\ell \to X_\ell.$$

We now wish to solve iteratively the linear system

$$A_{\ell_{\max}} x_{\ell_{\max}} = b_{\ell_{\max}}$$

by using a multigrid method. A multigrid iteration results from the recursive application of a two-grid method. A two-grid method on level ℓ consists of ν_1 presmoothing steps on level ℓ, a coarse grid correction on level $\ell - 1$, and ν_2 postsmoothing steps again on level ℓ. The corresponding error propagation is

$$e_\ell^{(m+1)} = [S_\ell^{\nu_2}(I - pA_{\ell-1}^{-1} r A_\ell) S_\ell^{\nu_1}]\, e_\ell^{(m)},$$

where S_ℓ denotes the iteration matrix of the smoother.

2.1. Classical smoothers.

We shall limit the present discussion to the choice of the smoother. All other multigrid components, such as p, r, and $A_{\ell-1}$, follow from standard choices. If the smoother results from a consistent linear iterative method, the iteration matrix of the smoother, S_ℓ, can be written as

$$S_\ell = I - N_\ell A_\ell. \tag{2.1}$$

For instance, let $A = D+L+U$, with D the diagonal, L the lower triangular part, and U the upper triangular part of A. Then damped Jacobi smoothing corresponds to

$$S_\omega = I - \omega D^{-1} A, \tag{2.2}$$

whereas Gauss–Seidel smoothing corresponds to

$$S_{GS} = I - (D + L)^{-1} A. \tag{2.3}$$

In (2.2) the choice of ω must ensure good smoothing properties of the resulting damped Jacobi method. Yet the "optimal" value of ω is known only for certain model problems (see section 3.2). In contrast, the Gauss–Seidel method is parameter-free and typically leads to improved smoothing over the damped Jacobi method. Unfortunately, the Gauss–Seidel method (2.3) is inherently sequential and therefore difficult to implement on a parallel architecture. Yet with an appropriate coloring of the unknowns (e.g., red-black ordering on a regular grid) it is sometimes possible to attain reasonable parallel efficiency with the Gauss–Seidel approach.

If neither damped Jacobi nor Gauss–Seidel leads to satisfactory smoothing, one can resort to more robust smoothers, such as the popular ILU smoothers based on the incomplete LU (ILU) decomposition of A_ℓ; see, for instance, [28]. Because each ILU smoothing step requires the solution of upper and lower triangular systems, it remains inherently sequential and difficult to implement in parallel. It is also difficult to improve the ILU smoother locally, say near the boundary or a singularity, without seriously affecting the sparsity of the LU factors.

2.2. SPAI smoothers. Most smoothers commonly used in multigrid methods, such as damped Jacobi, Gauss–Seidel, or ILU, have the form

$$x_\ell^{(k+1)} = x_\ell^{(k)} - W_\ell^{-1}(A_\ell x_\ell^{(k)} - b_\ell), \tag{2.4}$$

with W_ℓ a (sparse) approximation of A_ℓ; moreover, the computational cost of solving a linear system with matrix W_ℓ must be reasonable. In contrast, the SPAI smoothers lead to the iteration

$$x_\ell^{(k+1)} = x_\ell^{(k)} - M_\ell \, (A_\ell x_\ell^{(k)} - b_\ell), \tag{2.5}$$

where M_ℓ is sparse and explicitly known. Hence the iteration in (2.5) requires only matrix-vector multiplications and vector-vector additions, and no solution of a linear system; it is therefore easy to implement in a parallel environment.

To construct the sparse approximate inverse M of A, we shall minimize $I - MA$ in the Frobenius norm for a prescribed sparsity pattern of M. Here we have dropped the index ℓ to simplify the notation. The Frobenius norm, denoted by $\|\cdot\|_F$, naturally leads to inherent parallelism because the rows m_k of M can be computed independently of one another. Indeed, since

$$\|I - MA\|_F^2 = \sum_{k=1}^{n} \|e_k^\top - m_k A\|_2^2, \tag{2.6}$$

the solution of (2.6) separates into the n *independent least-squares problems* for the sparse (row) vectors m_k,

$$\min_{m_k} \|e_k^\top - m_k A\|_2, \qquad k = 1, \dots, n. \tag{2.7}$$

Here e_k denotes the kth unit vector. Because A and M are sparse these least-squares problems have small dimensions.

Since an effective sparsity pattern of M is usually unknown a priori, the original SPAI-Algorithm [14] begins with a diagonal pattern. Then the algorithm proceeds by augmenting the sparsity pattern of M to further reduce each residual $r_k = e_k^\top - m_k A$. The progressive reduction of the 2-norm of r_k involves two steps. First, the algorithm identifies a set of potential new candidates, based on the sparsity pattern of A and the current (sparse) residual r_k. Second, the algorithm selects the most profitable entries, usually less than five entries, by computing for each candidate a cheap upper bound for the reduction in $\|r_k\|_2$. Once the new entries have been selected and added to m_k, the (small) least-squares problem (2.7) is solved again with the augmented set of indices. The algorithm proceeds until each row m_k of M satisfies

$$\|e_k^\top - m_k A\|_2 < \varepsilon. \tag{2.8}$$

Here ε is a tolerance set by the user, which controls the fill-in and the quality of the preconditioner M. A lower value of ε usually yields a more effective preconditioner, but the cost of computing $M = \text{SPAI}\,(\varepsilon)$ may become prohibitive; moreover, a denser M results in a higher cost per iteration in (2.5). The optimal value of ε minimizes the total time; it depends on the problem, the discretization, the desired accuracy, and the computer architecture. Further details about the original SPAI-Algorithm can be found in [14].

What is the cost associated with computing M? Let q denote the typical number of nonzero entries per row m_k, and let p denote the typical number of nonzero entries per row in A. Then the cost of solving each least-squares problem in (2.7) (by the method of normal equations) is about $(q + p/3)\,p^2$ flops [12, p. 224]. Since $q \gg p$, the total amount of work involved in computing M is about $n\,q\,p^2$ flops. To ensure that the total amount of work for computing the approximate inverses M_ℓ on all levels remains $O(n)$, it is crucial that both p and q remain bounded from above independently of n. Clearly, in contrast to the Jacobi or Gauss–Seidel methods, the additional effort of computing M seems at first cumbersome and expensive. The approximate inverses, however, need only be computed once prior to the multigrid iteration. Moreover, both the computation of M and its repeated application as a smoother are inherently parallel, so that the usefulness of SPAI smoothers in a parallel environment is all too obvious in comparison to the sequential Gauss–Seidel iteration.

In addition to SPAI(ε), we shall also consider the following two greatly simplified SPAI smoothers with fixed sparsity patterns: SPAI-0, where M is diagonal, and SPAI-1, where the sparsity pattern of M is that of A. Both solve the least-squares problem (2.7) and thus minimize $\|I - MA\|_F$ for the sparsity pattern chosen a priori. This eliminates the search for the sparsity pattern of M and thus greatly reduces the cost of computing the approximate inverse. The SPAI-1 smoother coincides with the SAI(0,1) smoother of Tang and Wan [25].

For SPAI-0, $M = \text{diag}(m_{kk})$ is diagonal and can be calculated directly:

$$m_{kk} = \frac{a_{kk}}{\|a_k\|_2^2}, \qquad 1 \le k \le n, \tag{2.9}$$

with a_k the kth row of A. We note that M is always well defined if A is nonsingular. Unlike damped Jacobi, the SPAI-0 smoother is parameter-free.

To summarize, we shall consider the following hierarchy of SPAI smoothers, which all minimize $\|I - MA\|_F$ for a certain sparsity pattern of M.

SPAI-0: $M = \text{diag}(m_{kk})$ is diagonal, with m_{kk} given by (2.9).

SPAI-1: The sparsity pattern of M is that of A.

SPAI(ε): The sparsity pattern of M is determined automatically by the SPAI-Algorithm [14]. Then each row m_k satisfies (2.8) for a given ε.

We have found that, in many situations, SPAI-0 and SPAI-1 yield ample smoothing. However, the added flexibility in providing an automatic criterion for improving the smoother via the SPAI-Algorithm remains very useful. Indeed, either SPAI-0 or

SPAI-1 can be used as initial guess for SPAI(ε) and thus be locally improved upon where needed by reducing ε (see section 4.2). For matrices with inherent (small) block structure, typical from the discretization of systems of partial differential equations, the Block-SPAI-Algorithm [2] greatly reduces the cost of computing M.

3. SPAI-0 smoothing.

In this section we concentrate on the simplest SPAI smoother, SPAI-0. First, we shall show that SPAI-0 satisfies the smoothing property in two quite general situations. Second, we shall compare the two diagonal smoothers, SPAI-0 and damped Jacobi, both in a theoretical idealized setting (constant coefficients, equispaced grid, periodic boundary conditions, etc.) and via numerical experiments.

3.1. The smoothing property.

From [15] and [16] we recall the following two conditions, which play a fundamental role in multigrid convergence theory:

1. The smoothing property [16, Def. 10.6.3]:

$$\|A_\ell S_\ell^\nu\|_2 \le \eta(\nu)\|A_\ell\|_2 \qquad \forall \nu, 0 \le \nu < \infty, \quad \forall \ell \ge 1, \tag{3.1}$$

$$\eta(\nu) \text{ any function with } \quad \lim_{\nu\to\infty} \eta(\nu) = 0.$$

2. The approximation property [16, sect. 10.6.3]:

$$\|A_\ell - pA_{\ell-1}r\|_2 \le \frac{C_A}{\|A_\ell\|_2} \qquad \forall \ell \ge 1. \tag{3.2}$$

Although we have stated these properties with respect to the Euclidean norm, other choices are possible. In general, the smoothing and approximation properties together imply convergence of the two-grid method and of the multigrid W-cycle, with a contraction number independent of the level number ℓ. Moreover, for symmetric positive definite problems, both conditions also imply multigrid V-cycle convergence independent of ℓ; see Hackbusch [16, sect. 10.6] for details.

The approximation property is independent of the smoother S_ℓ; it depends only on the discretization (A_ℓ, $A_{\ell-1}$), the prolongation operator p, and the restriction operator r. In [16] the approximation property is shown to hold for a large class of discrete elliptic boundary value problems. For symmetric positive definite problems the smoothing property usually holds for classical smoothers like damped Jacobi, (symmetric) Gauss–Seidel, and incomplete Cholesky. We shall now prove that the smoothing property (3.1) holds for SPAI-0 under reasonable assumptions on A_ℓ. To do so, we first recall (in a slightly simpler form) the following result for later reference.

LEMMA 3.1 (see [16, Lem. 10.7.4]). *Let A_ℓ and W_ℓ be symmetric and positive definite, and $S_\ell = I - W_\ell^{-1}A_\ell$. Assume that*

$$0 < A_\ell \le \Gamma W_\ell \qquad \forall \ell \ge 0 \quad \textit{with} \quad 0 < \Gamma < 2 \tag{3.3}$$

and that

$$\|W_\ell\|_2 \le C_W \|A_\ell\|_2 \qquad \forall \ell \ge 0. \tag{3.4}$$

Then S_ℓ satisfies the smoothing property (3.1), *with*

$$\eta(\nu) = C_W \max\{\eta_0(\nu), \Gamma|1-\Gamma|^\nu\}, \quad \eta_0(\nu) = \frac{1}{e\nu} + O(\nu^{-2}) \quad (\nu \to \infty). \tag{3.5}$$

In (3.3) and (3.4) both Γ and C_W must be independent of ℓ.

We shall now apply Lemma 3.1 to prove that SPAI-0 satisfies the smoothing property (3.1). To do so, we must show that W_ℓ satisfies (3.3) and (3.4), with $\Gamma < 2$. Here W_ℓ is the inverse of the diagonal approximate inverse defined in (2.9). Hence

$$W = \operatorname{diag}\left(\frac{\|a_i\|_2^2}{a_{ii}}\right), \tag{3.6}$$

where a_i denotes the ith row of A. We have dropped the level index ℓ to simplify the notation. Since A is symmetric and positive definite, $a_{ii} > 0,\ 1 \le i \le n$, and thus W is positive definite.

LEMMA 3.2. *Let A be a symmetric matrix with positive diagonal entries and let W be given by* (3.6)*. Furthermore, let p_i denote the number of nonzero off-diagonal entries in the ith row of A and assume that*

$$p \equiv \max_i p_i \le 7. \tag{3.7}$$

Then A satisfies $A \le \Gamma W$, with $\Gamma = (1 + \sqrt{1+p})/2 < 2$.

Proof. We seek $\Gamma < 2$ such that $A \le \Gamma W$. First, we let $A = D - R$, with $D = \operatorname{diag}(A)$. Then

$$\begin{aligned} &\Gamma W - A \ge 0 \\ \iff\ &\Gamma\left(D + \operatorname{diag}\left(\sum_{j\ne i}\frac{a_{ij}^2}{a_{ii}}\right)\right) - D + R \ge 0 \\ \iff\ &(\Gamma-1)D + \Gamma\operatorname{diag}\left(\sum_{j\ne i}\frac{a_{ij}^2}{a_{ii}}\right) + R \ge 0. \end{aligned} \tag{3.8}$$

We note that the first two terms in (3.8) are diagonal matrices, while all off-diagonal entries are located in R. We now assume that $\Gamma \ge 1$, so that all entries on the main diagonal in (3.8) are nonnegative. According to Gershgorin's theorem, for (3.8) to hold it is sufficient to have

$$\sum_{j\ne i}|a_{ij}| \le (\Gamma-1)a_{ii} + \Gamma\sum_{j\ne i}\frac{a_{ij}^2}{a_{ii}}, \qquad 1 \le i \le n. \tag{3.9}$$

Next, we divide (3.9) by a_{ii}, which yields the equivalent condition

$$\sum_{j\ne i}\beta_{ij} \le \Gamma - 1 + \Gamma\sum_{j\ne i}\beta_{ij}^2, \qquad 1 \le i \le n. \tag{3.10}$$

Here we have defined

$$\beta_{ij} = \frac{|a_{ij}|}{a_{ii}}.$$

Since p_i is the number of nonzero off-diagonal elements in row i of A, we conclude by Cauchy–Schwarz that

$$\sum_{j\ne i}\beta_{ij} \le \sqrt{p_i}\sqrt{\sum_{j\ne i}\beta_{ij}^2} \le \sqrt{p}\sqrt{\sum_{j\ne i}\beta_{ij}^2}, \qquad 1 \le i \le n.$$

Thus, for (3.10) to hold it is sufficient to have

$$\sqrt{p}\,\sqrt{\sum_{j\neq i}\beta_{ij}^2} \leq \Gamma - 1 + \Gamma\sum_{j\neq i}\beta_{ij}^2, \qquad 1\leq i\leq n.$$

Therefore, since $x = \sum_{j\neq i}\beta_{ij}^2$ is real, nonnegative, but otherwise arbitrary, it is sufficient to require that

$$\begin{aligned} &\sqrt{p}\,\sqrt{x} \leq \Gamma - 1 + \Gamma x \qquad \forall x\in[0,\infty),\\ \iff\ &\Gamma^2x^2 + (2\Gamma(\Gamma-1)-p)x + (\Gamma-1)^2 \geq 0 \qquad \forall x\in[0,\infty),\\ \iff\ &(2\Gamma(\Gamma-1)-p)^2 - 4\Gamma^2(\Gamma-1)^2 \leq 0. \end{aligned} \tag{3.11}$$

The last inequality (3.11) is equivalent to

$$-4\Gamma(\Gamma-1)p + p^2 \leq 0,$$

which holds for $\Gamma = (1+\sqrt{1+p})/2$. The assumption $p\leq 7$ yields $\Gamma < 2$. □

Remark 1. The result in Lemma 3.2 is sharp in the following sense. For every $\Gamma\in(0,2)$ there exists a symmetric matrix A with positive diagonal entries and with $p\geq 8$ such that the matrix $\Gamma W - A$ is not positive semidefinite. Indeed, we take the matrix $A = (1-\alpha)I + \alpha\mathbf{1}\mathbf{1}^T$, with $\alpha = (2\Gamma)^{-1}$ and $\mathbf{1} = (1,1,\dots,1)^T\in\mathbb{R}^n$. Then

$$\Gamma W - A = \Gamma\,\mathrm{diag}(1+(n-1)\alpha^2) - \big((1-\alpha)I + \alpha\mathbf{1}\mathbf{1}^T\big),$$

with smallest eigenvalue

$$\lambda_{\min} = \frac{1}{4\Gamma}\big(4\Gamma(\Gamma-1) - (n-1)\big).$$

Hence $\lambda_{\min} < 0$ if $p = n-1 \geq 8$.

Lemma 3.3. *Let W be given by* (3.6), *and let $\hat{C}$ be such that*

$$\max_i \sum_{j=1}^n \frac{a_{ij}^2}{a_{ii}^2} \leq \hat{C}. \tag{3.12}$$

Then W satisfies $\|W\|_2 \leq \hat{C}\,\|A\|_2$.

Proof. The proof is immediate, since

$$\begin{aligned} \|W\|_2 &= \left\|\mathrm{diag}(a_{ii})\,\mathrm{diag}\left(\frac{\|a_i\|_2^2}{a_{ii}^2}\right)\right\|_2\\ &\leq \|\,\mathrm{diag}(a_{ii})\|_2\left\|\mathrm{diag}\left(\frac{\|a_i\|_2^2}{a_{ii}^2}\right)\right\|_2\\ &\leq \|A\|_2\,\hat{C}. \qquad \square \end{aligned}$$

From Lemmas 3.1, 3.2, and 3.3 we now immediately conclude the following result.

Theorem 3.4. *Let A be symmetric positive definite, $\hat{C}$ as in* (3.12), *and let $S = I - MA$, with M the SPAI-*0 *preconditioner given by* (2.9). *Assume that p, the maximal number of nonzero off-diagonal entries per row, satisfies $p\leq 7$; see* (3.7). *Then S satisfies the smoothing property* (3.1), *with $\eta(\nu)$ as in* (3.5), *$C_W = \hat{C}$, and $\Gamma = (1+\sqrt{1+p})/2 < 2$.*

Remark 2. The following example shows that the condition on p in Theorem 3.4 is sharp. Take A as in remark 1, with $\alpha = (\sqrt{n}+1)^{-1}$. Then A is symmetric positive definite, $\|A\|_2 = \sqrt{n}$, and the inequality in (3.12) holds with $\hat{C} = 2$. Since $\mathbf{1} = (1, \dots, 1)^T$ is an eigenvector of both A and S, we immediately conclude that

$$\frac{\|AS^\nu\|_2}{\|A\|_2} \geq \frac{\|AS^\nu \mathbf{1}\|_2}{n} = \left(\frac{1}{2}(\sqrt{n}-1)\right)^\nu .$$

For $n \geq 9$, $\|AS^\nu\|_2 \|A\|_2^{-1} \nrightarrow 0$ as $\nu \to \infty$, and hence S does not satisfy the smoothing property (3.1).

We remark that neither M-matrix properties nor diagonal dominance of A_ℓ are needed to show that the smoothing property holds for SPAI-0. In the context of a multigrid convergence analysis the constant $\hat{C}$ in (3.12) must be independent of the level number ℓ. For the problems considered here this is always satisfied. Condition (3.7) is satisfied by standard second-order finite difference approximations of scalar elliptic boundary value problems in two or three space dimensions. It is also satisfied by linear finite element discretizations on a triangular mesh if each node on the coarsest mesh has at most seven neighbors. This property is then transferred to all finer levels if regular mesh refinement is used.

Next, we show that if A is weakly diagonally dominant, that is,

$$\sum_{j \neq i} |a_{ij}| \leq |a_{ii}| \qquad \forall\, i, \quad 1 \leq i \leq n,$$

we may drop condition (3.7) and thus obtain another criterion for the smoothing property.

THEOREM 3.5. *Let A be symmetric positive definite and weakly diagonally dominant. Furthermore, let $S = I - MA$, with M the SPAI-0 preconditioner given by* (2.9)*, and $C > 0$ be such that for every i, $1 \leq i \leq n$,*

$$\text{(3.13)} \qquad \textit{either} \;\; \sum_{j \neq i} a_{ij}^2 = 0 \;\; \textit{or} \;\; \sum_{j \neq i} \frac{a_{ij}^2}{a_{ii}^2} \geq C.$$

Then S satisfies the smoothing property (3.1)*, with $\eta(\nu)$ as in* (3.5)*, $C_W = 2$, and $\Gamma = 2/(1+C) < 2$.*

Proof. We first consider the case where

$$\text{(3.14)} \qquad \sum_{j \neq i} \frac{a_{ij}^2}{a_{ii}^2} \geq C > 0 \qquad \forall\, i, \quad 1 \leq i \leq n.$$

Again we seek $\Gamma < 2$ such that $A \leq \Gamma W$. To do so, we first follow the proof of Lemma 3.2 until (3.10). Now, since A is weakly diagonally dominant, we have

$$\text{(3.15)} \qquad \sum_{j \neq i} \beta_{ij} = \sum_{j \neq i} \frac{|a_{ij}|}{a_{ii}} \leq 1.$$

Hence for (3.10) to hold, it is sufficient to require

$$1 \leq \Gamma - 1 + \Gamma \sum_{j \neq i} \beta_{ij}^2, \qquad 1 \leq i \leq n,$$

which is equivalent to

$$\Gamma \geq \frac{2}{1+\sum_{j\neq i}\beta_{ij}^2}, \qquad 1 \leq i \leq n.$$

Because of (3.14) we can choose $\Gamma = 2/(1+C) < 2$.

To show that inequality (3.4) is indeed satisfied with $C_W = 2$, we first note that

$$\begin{aligned} \|W\|_2 = \|M^{-1}\|_2 &= \max_i \left(\frac{\|a_i\|_2^2}{a_{ii}}\right) \\ &= \max_i \left(a_{ii} + \frac{1}{a_{ii}}\sum_{j\neq i} a_{ij}^2\right) \\ &\leq \max_i \left(a_{ii} + \frac{1}{a_{ii}}\left(\sum_{j\neq i}|a_{ij}|\right)^2\right). \end{aligned}$$

The weak diagonal dominance of A then implies

$$\|W\|_2 \leq 2\max_i a_{ii} \leq 2\|A\|_2.$$

Next, we consider the general case, where C satisfies (3.13). Let $k > 0$ be the number of rows for which the only nonzero entry lies on the main diagonal. We apply a permutation P to A, so that these k rows appear first in

$$PAP^T = \begin{pmatrix} A_{11} & 0 \\ 0 & A_{22} \end{pmatrix}.$$

Hence A_{11} denotes the $k \times k$ diagonal block. The matrix $S = I - MA$ is also partitioned accordingly:

$$PSP^T = \begin{pmatrix} S_{11} & 0 \\ 0 & S_{22} \end{pmatrix}.$$

Because A_{11} is diagonal, the upper left block $S_{11} = I - A_{11}^{-1}A_{11}$ is identically zero. We now apply the first part of the proof to A_{22}, which yields

$$\|A_{22}S_{22}^{\nu}\|_2 \leq 2\max\{\eta_0(\nu), \Gamma|1-\Gamma|^{\nu}\}\|A_{22}\|_2,$$

with $\Gamma = 2/(1+C)$. Therefore S satisfies the smoothing property, since

$$\begin{aligned} \|AS^{\nu}\|_2 = \|(PAP^T)(PSP^T)^{\nu}\|_2 &= \left\| \begin{pmatrix} A_{11} & 0 \\ 0 & A_{22} \end{pmatrix} \begin{pmatrix} 0 & 0 \\ 0 & S_{22} \end{pmatrix}^{\nu} \right\|_2 \\ &= \|A_{22}S_{22}^{\nu}\|_2 \leq 2\max\{\,\eta_0(\nu), \Gamma|1-\Gamma|^{\nu}\,\}\|A_{22}\|_2 \\ &\leq 2\max\{\,\eta_0(\nu), \Gamma|1-\Gamma|^{\nu}\,\}\|A\|_2. \qquad \square \end{aligned}$$

In the context of a multigrid convergence analysis the constant C in (3.13) must be independent of the level number ℓ. For the problems considered here this is always satisfied. In summary, we have shown for A symmetric positive definite that SPAI-0 satisfies the smoothing property if either A is weakly diagonally dominant or if the maximal number of nonzero off-diagonal entries per row is less than or equal to seven.

3.2. SPAI-0 versus Jacobi. Before we proceed with a comparison of the performance of these two diagonal smoothers via numerical experiments, we first point out a very special situation where SPAI-0 and damped Jacobi, with optimal relaxation parameter ω^*, lead to identical smoothers.

For the discrete Laplacian on a regular grid with periodic boundary conditions, the damping parameter ω^*, which is "optimal" with respect to smoothing, is known. Following the standard Fourier analysis in [27, sect. 7.3], we consider a regular d-dimensional equispaced mesh with n grid points in each dimension. (For simplicity we assume n to be even.) Then the eigenvalues of the discrete Laplacian with periodic boundary conditions are

$$\sigma(\theta) = \frac{4}{h^2} \sum_{j=1}^{d} \sin^2\left(\frac{\theta_j}{2}\right), \qquad \theta = (\theta_1, \theta_2, \dots, \theta_d),$$

with $h = 1/n$ and $\theta_j \in \{-\pi + 2\pi h, -\pi + 4\pi h, \dots, \pi\}$. Note that $\theta \in \prod_{j=1}^{d}[-\pi, \pi]$. The iteration matrix of the damped Jacobi method has eigenvalues

$$\lambda(\theta) = 1 - \frac{2\omega}{d} \sum_{j=1}^{d} \sin^2\left(\frac{\theta_j}{2}\right).$$

We now concentrate on the high frequencies, which correspond to the subset

$$\bar{\Theta}_r = \prod_{j=1}^{d}[-\pi, \pi] \setminus \prod_{j=1}^{d}\left(-\frac{\pi}{2}, \frac{\pi}{2}\right).$$

The amount of damping on the high-frequency components is measured by the *smoothing factor*,

$$\mu(\omega) = \max_{\theta \in \bar{\Theta}_r} |\lambda(\theta)|.$$

It is independent of the mesh size but depends on ω. The optimal damping parameter, ω^*, is that for which the smoothing factor is minimal,

$$\mu^* = \mu(\omega^*) = \min_{\omega} \mu(\omega).$$

To determine ω^* we first calculate $\mu(\omega)$. For $\theta \in \bar{\Theta}_r$, the extreme values of $\sum_{j=1}^{d} \sin^2(\theta_j/2)$ are $1/2$ and d. Thus, we find that

$$\mu(\omega) = \max\left\{\left|1 - \frac{\omega}{d}\right|, |1 - 2\omega|\right\}.$$

The minimization of $\mu(\omega)$ then yields the optimal damping parameter,

$$\omega^* = \frac{2d}{2d+1}. \tag{3.16}$$

PROPOSITION 3.6. *Consider the discrete Laplacian with periodic boundary conditions in d space dimensions, which results from a standard second-order finite difference discretization on an equispaced grid. Then SPAI-*0 *and Jacobi smoothing, with optimal damping parameter* ω^* *given by* (3.16)*, are identical.*

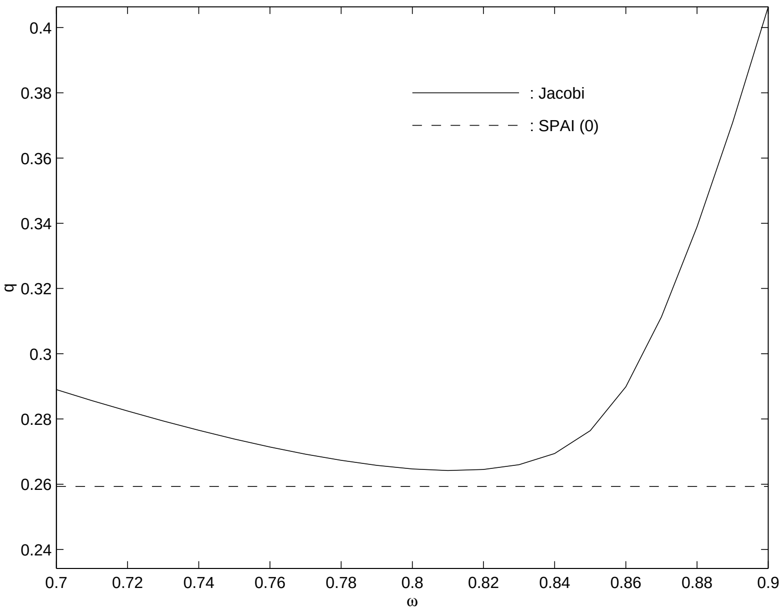


FIG. 3.1. *Comparison of the convergence rates q obtained with SPAI-*0 *and damped Jacobi for varying relaxation parameter ω; see* (4.2) *for the definition of q.*

Proof. Since for the discrete d-dimensional Laplacian we have $a_{ii} = (2d)\,h^{-2}$ and $\|a_i\|_2^2 = 2d\,(2d+1)\,h^{-4}$, for all $1 \le i \le n$, the approximate inverse M defined in (2.9) for SPAI-0 has the constant diagonal entry $m_{ii} = h^2/(2d+1)$. Therefore $M = \omega^* D^{-1}$ and the SPAI-0 and the damped Jacobi smoothers, with optimal damping parameter ω^* given in (3.16), coincide. □

In this special situation, the parameter-free SPAI-0 smoother automatically yields a scaling of diag(A) which minimizes the smoothing factor; in that sense it is optimal.

Although both SPAI-0 and damped Jacobi yield diagonal smoothers, they typically differ, even with constant coefficients on an equispaced mesh. Indeed, the presence of boundary conditions modifies the rows of A_ℓ, which correspond to nodes on the boundary, and thus modifies SPAI-0 locally. We now compare quantitatively the performance of these two diagonal smoothers on the following class of model problems:

$$\begin{aligned} -\mathrm{div}(a(x,y)\nabla u(x,y)) &= 1 \qquad \text{in } \Omega, \\ u(x,y) &= 0 \qquad \text{on } \partial\Omega. \end{aligned} \tag{3.17}$$

First, we choose $\Omega = (0,1)\times(0,1)$, set $a(x,y) \equiv 1$, and discretize the problem with continuous piecewise linear finite elements on a triangular mesh. The various meshes are obtained successively by uniform refinement, starting from the coarsest mesh with a single unknown in the center of Ω. Here we use a multigrid V-cycle iteration with one pre- and one postsmoothing step ($\nu_1 = \nu_2 = 1$). We recall that SPAI-0 is parameter-free, whereas damped Jacobi contains the single relaxation parameter ω. In general the optimal damping parameter ω^* is unknown in advance and must be determined numerically in any given situation.

Is it possible for damped Jacobi to beat SPAI-0 by varying ω and thus determining the optimal value ω^*? In Figure 3.1 we compare the convergence rate of SPAI-0 with that of Jacobi smoothing. The fastest convergence rate with damped Jacobi is attained with $\omega^* \simeq 0.81$, but the convergence rate obtained with SPAI-0 still lies slightly below it.

We have performed similar numerical experiments for more difficult problems, including L-shaped domains and problems with discontinuous coefficients. In all cases the overall picture remains the same: the convergence rate obtained with SPAI-0 smoothing consistently lies below that obtained with damped Jacobi smoothing for all values of ω. Although the improvement is usually small, and thus not really significant, it is remarkable because SPAI-0 is parameter-free.

Finally we compare SPAI-0 with Jacobi smoothing for an elliptic differential operator of the form

$$-c_{11}\partial_{xx} + 2c_{12}\partial_{xy} - c_{22}\partial_{yy},$$

where c_{11}, c_{12}, and c_{22} are constant and $c_{11} \geq c_{22}$. Because of consistency and ellipticity a second-order finite difference discretization on a doubly periodic uniform square grid usually results in the stencil

$$\begin{bmatrix} -\frac{B}{2} & -C & \frac{B}{2} \\ -A & 2(A+C) & -A \\ \frac{B}{2} & -C & -\frac{B}{2} \end{bmatrix},$$

with

$$A, C > 0, \quad A + C = 1, \quad A \geq C, \quad B^2 < AC.$$

In [29] Yavneh and Olvovsky derive the smoothing factor $\mu(\omega)$ of damped Jacobi when applied to such a stencil,

$$\mu(\omega) = \max\left\{ |1 - 2\omega|, \left|1 - \omega\left(1 - \sqrt{A^2 + B^2}\right)\right| \right\}.$$

They also determine the corresponding optimal relaxation parameter, ω^*, which minimizes $\mu(\omega)$ over the high frequencies:

$$\omega^* = \frac{2}{3 - \sqrt{A^2 + B^2}}\ .$$

Since the stencil is constant throughout the entire grid, the approximate inverse M for SPAI-0 reduces to a constant diagonal matrix. Thus, in this very special situation, the SPAI-0 smoother corresponds to the damped Jacobi method with relaxation parameter

$$\omega^{\text{SPAI}} = \frac{4}{4 + B^2 + 2C^2 + 2A^2}\ .$$

Next, we introduce the variables

$$x = \frac{C}{A} \in (0, 1) \text{ and } y = \frac{B^2}{AC} \in [0, 1).$$

Hence the smaller the x, the stronger the anisotropy. We can now rewrite $\mu(\omega)$, ω^*, and ω^{SPAI} only in terms of x and y as

$$\begin{aligned}\mu(\omega) &= \max\left\{ |1-2\omega|\,,\, 1-\omega+\omega\frac{\sqrt{1+xy}}{1+x} \right\}\,,\\ \omega^* &= \frac{2(1+x)}{3(1+x)-\sqrt{1+xy}}\,,\\ \omega^{\text{SPAI}} &= \frac{4(x+1)^2}{4(x+1)^2+2(x^2+1)+xy}\,.\end{aligned}$$

In the left frame of Figure 3.2 the smoothing factor μ^* for the optimal relaxation parameter ω^* is shown. The highest reduction is reached for $x = 1$ and $y = 0$, which corresponds to the Laplacian operator ($A = C = 1/2$, $B = 0$). As x diminishes and eventually drops below 0.2, we observe that damped Jacobi becomes a very poor smoother.

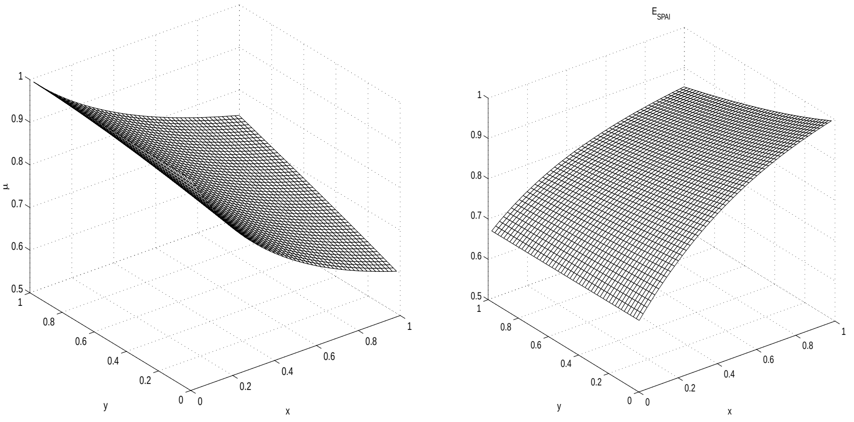


FIG. 3.2. *Comparison of SPAI-*0 *vs. damped Jacobi: the smoothing factor μ^* for optimally damped Jacobi (left) and the relative smoothing efficiency E_{SPAI} for SPAI-*0 *(right).*

To compare the effectiveness of SPAI-0 smoothing with that of optimally damped Jacobi, we introduce the *relative smoothing efficiency*,

$$E_{\text{SPAI}} = \frac{\log\,\mu(\omega^{\text{SPAI}})}{\log\,\mu(\omega^*)} \in (0,1].$$

It is shown in the right frame of Figure 3.2. If $E_{\text{SPAI}} = 100\%$ the SPAI-0 method automatically produces the optimal damping parameter, whereas if $E_{\text{SPAI}} = 50\%$, for instance, SPAI-0 would require twice as many iterations as optimally damped Jacobi to obtain the same smoothing effect. In particular, for the standard five-point Laplacian ($x = 1, y = 0$) SPAI-0 yields $E_{\text{SPAI}} = 100\%$, as it coincides with optimally damped Jacobi; see Proposition 3.6. The relative efficiency for SPAI-0 is somewhat reduced in more general situations with mixed derivatives or strong anisotropy. Nevertheless, E_{SPAI} lies above 75% for $x \geq 0.2$; it drops down to 67% only in extreme cases ($x \to 0$), for which damped Jacobi would be ineffective anyway. Whenever damped Jacobi is a good smoother, SPAI-0 typically provides at least 80–90% efficiency with respect to optimally damped Jacobi smoothing for second-order finite difference discretizations of elliptic differential operators.

4. Numerical results. To illustrate the usefulness and versatility of SPAI smoothing, we shall now consider various standard test problems. In all cases we use a regularly refined sequence of equispaced grids. The differential equation considered is discretized on the finest level with standard finite differences. The coarse grid operators are obtained via the Galerkin product formula, $A_{\ell-1} = r\, A_\ell\, p$, with $r = p^\top$ and p standard linear interpolation. We use a multigrid V-cycle iteration, with two pre- and two postsmoothing steps ($\nu_1 = \nu_2 = 2$) and $x_\ell^{(0)} = 0$ as initial guess. On the coarsest level ($\ell = 0$), we solve exactly for the single unknown remaining at the center of the domain. The iteration proceeds until the relative residual drops below the prescribed tolerance,

$$\frac{\|b - A_\ell\, x_\ell^{(m)}\|}{\|b\|} < 10^{-8}. \tag{4.1}$$

Then we calculate the average rate of convergence,

$$q = \left(\frac{\|b - A_\ell\, x_\ell^{(m)}\|}{\|b\|} \right)^{1/m}. \tag{4.2}$$

We recall that the ordering of the unknowns does not affect the SPAI-smoothing iteration (2.5), but it does affect the Gauss–Seidel iteration. In all numerical examples we shall use lexicographic ordering of the unknowns.

Clearly, when comparing the performance of various smoothers, we cannot limit ourselves to comparing the number of multigrid iterations, but we must also take into account the amount of arithmetic work due to the smoother. To do so, we calculate the total density ratio, ρ, of nonzero entries in M to those in A on all grid levels, $1 \le i \le \ell$, where smoothing is applied:

$$\rho = \frac{\sum_{i=1}^{\ell} \operatorname{nnz}(M_i)}{\sum_{i=1}^{\ell} \operatorname{nnz}(A_i)}. \tag{4.3}$$

Hence the additional amount of work due to the smoother is proportional to ρ. For a standard five-point stencil on a regular two-dimensional grid, $\rho_{\text{SPAI-0}} \simeq 1/5$, like damped Jacobi. Since $\rho_{\text{SPAI-1}} = 1$, the SPAI-1 smoother is about 67% times more expensive here than the SPAI-0 smoother. For SPAI(ε), the total density ratio, $\rho_{\text{SPAI}(\varepsilon)}$, depends on ε: it increases monotonically with decreasing ε. We remark that $\rho_{\text{SPAI}(\varepsilon)} < 1$ whenever SPAI(ε) leads to an approximate inverse sparser than SPAI-1.

4.1. Poisson problem. We first consider the Poisson problem on the unit square with Dirichlet boundary conditions (3.17). In Table 4.1 we compare the convergence rates obtained with various smoothers. All SPAI smoothers lead to h-independent convergence rates. We observe a steady decrease of the convergence rate, q, for smaller values of ε, paralleled, of course, by an increase in ρ given in parentheses. Note that SPAI-1 leads to a more effective but denser smoother than SPAI(ε) with $\varepsilon = 0.35$, yet the situation is reversed as ε decreases below 0.25.

4.2. Locally anisotropic diffusion. We now consider the locally anisotropic diffusion problem

$$-\left(\nu(x,y) \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) = 1 \quad \text{in} \quad (0,1) \times (0,1), \tag{4.4}$$

Table 4.1

Poisson problem: convergence rates q obtained with SPAI-0, SPAI-1, SPAI(ε), and Gauss–Seidel smoothing. The relative total density, ρ, defined in (4.3), is given in parentheses. For SPAI-0, $\rho = 0.17$ and for SPAI-1, $\rho = 1$.

Grid size	G-S	SPAI-0	SPAI-1	SPAI(0.35)	SPAI(0.25)
32×32	0.04	0.09	0.04	0.06 (0.7)	0.03 (1.5)
64×64	0.05	0.09	0.04	0.07 (0.7)	0.03 (1.5)
128×128	0.05	0.09	0.04	0.08 (0.7)	0.04 (1.5)

Table 4.2

Locally anisotropic diffusion: convergence rates q for varying ν on a 128×128 grid. The relative density, ρ, is given in parentheses.

Smoother	ν				
	1	10^{-1}	10^{-2}	10^{-3}	10^{-6}
Gauss–Seidel	0.05	0.42	0.89	0.97	0.98
SPAI-0	0.09	0.72	0.95	0.98	0.98
SPAI-1	0.04	0.37	0.89	0.97	0.98
SPAI(0.4)	0.12 (0.7)	0.16 (0.7)	0.81 (0.7)	0.95 (0.8)	0.97 (0.8)
SPAI(0.25)	0.04 (1.5)	0.07 (1.6)	0.37 (1.7)	0.75 (1.9)	0.87 (1.9)

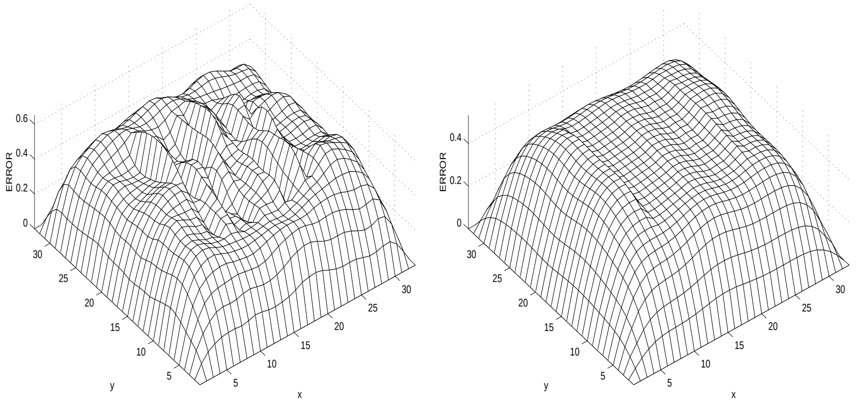


Fig. 4.1. *The error after five smoothing steps for the locally anisotropic diffusion problem with $\nu = 0.01$. Left: Gauss–Seidel; right: SPAI(0.25).*

with $u(x, y) = 0$ on the boundary. We set $\nu(x, y) = 1$ everywhere except inside the square $[1/4, 3/4] \times [1/4, 3/4]$, where $\nu(x, y) \equiv \nu$ is constant.

In Table 4.2 we compare the performance of Gauss–Seidel with SPAI smoothing for varying ν. For $\nu \leq 0.1$, the convergence rates for SPAI(0.4) lie consistently below those for Gauss–Seidel, while SPAI(0.25) leads to an even greater improvement. In particular, for $\nu = 0.01$ Gauss–Seidel smoothing barely converges, whereas SPAI(ε) with $\varepsilon \leq 0.4$ still yields acceptable convergence. Figure 4.1 shows that Gauss–Seidel is unable to smooth the error throughout Ω, mainly in the x-direction. In contrast, SPAI(ε) smoothing with $\varepsilon = 0.25$ results in a smooth error across the entire computational domain for $\nu = 0.01$. In Figure 4.2 we compare rows $e_j^\top A^{-1}$ and $e_j^\top M$, where M is computed with SPAI(0.25), for two unit basis vectors e_j. We consider e_j corresponding to the grid points $(1/2, 1/2)$ in the center of Ω, where $\nu = 0.01$, and $(1/8, 1/8)$ inside the surrounding region where $\nu = 1$. We observe how the approximate inverse, computed by the SPAI-Algorithm, captures the distinct local features of the true inverse. We recall that the sparsity pattern of M is not fixed a priori but adapted automatically by the SPAI-Algorithm. For $\nu \leq 0.1$, the SPAI(0.4) smoother is not only more effective but also sparser than the SPAI-1 smoother. Hence the sophisticated search of the SPAI-Algorithm for an effective sparsity pattern of M clearly benefits the smoother.

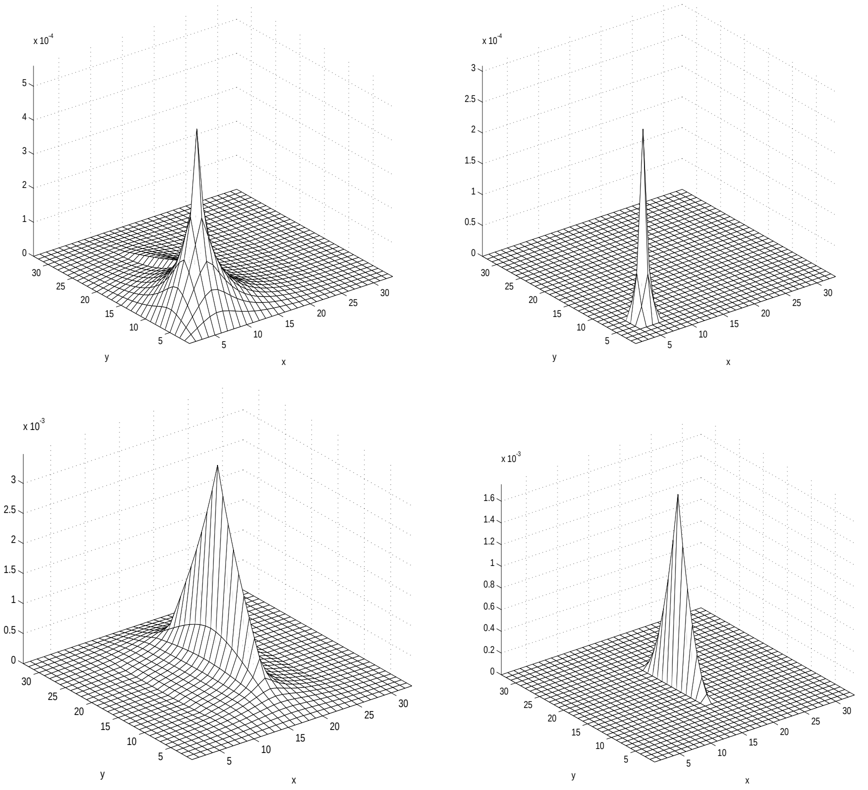


FIG. 4.2. *Comparison of a row of the true inverse,* $e_j^\top A^{-1}$ *(left), and of the approximate inverse,* $e_j^\top M$ *(right), for* $\nu = 0.01$*, where* M *is computed with SPAI*(0.25)*. Top:* e_j *corresponds to the grid point* $(1/8, 1/8)$*; bottom:* e_j *corresponds to the grid point* $(1/2, 1/2)$*.*

These results demonstrate the usefulness of SPAI smoothing; they corroborate previous results obtained in [25], with $\nu = 0.01$ everywhere in Ω. Nevertheless, as the contrast in the anisotropy increases further, the convergence rates obtained with SPAI smoothing deteriorate as well. If the anisotropy occupies a small area of the domain of interest, say only within a boundary layer, further reducing ε enables the algorithm to locally adjust the smoother and still reach acceptable convergence rates. However, if the anisotropy is strong and present throughout the domain, the SPAI smoothers will become too dense and thus too expensive.

Of course, various standard approaches combined with Gauss–Seidel smoothing, such as line relaxation or semicoarsening, provide the means to circumvent some of these problems [23]. However, these techniques are effective mainly on regular Cartesian grids, when the anisotropy is aligned with the mesh throughout Ω. They are difficult to use on unstructured grids and become expensive, for instance, when parts of the domain have strong anisotropy in x, while other parts have strong anisotropy in y. In contrast, the SPAI smoothers are not tied to a regular mesh or any specific discretization; moreover, they automatically adapt to local features of the problem.

4.3. Convection-diffusion problems.

We now consider

$$-\nu\Delta u(x,y) + \vec{v}(x,y)\cdot\nabla u(x,y) = 1, \quad \nu > 0, \tag{4.5}$$

in the unit square, where u vanishes on the boundary. Here u represents any scalar quantity advected by the flow field $\vec{v}$. Because the linear systems cease to be symmetric and positive definite, these problems lie outside of classical multigrid theory. We use centered second-order finite differences for the diffusion, but discretize the convection with first-order upwinding to ensure numerical stability.

TABLE 4.3

Constant flow direction with angle α from the x-axis: the convergence rates, q, obtained with different smoothers on a 128×128 grid for the diffusion and the convection dominated cases $\nu = 0.1$ and $\nu = 0.001$, respectively. The relative density, ρ, is given in parentheses.

Problem	G-S$_G$	SPAI-0	SPAI-1	SPAI(0.35)	SPAI(0.25)
$\nu = 0.1, \alpha = 45°$	0.05	0.11	0.05	0.08 (0.8)	0.04 (1.6)
$\nu = 0.1, \alpha = 225°$	0.05	0.11	0.05	0.08 (0.8)	0.04 (1.6)
$\nu = 0.001, \alpha = 45°$	†	0.98	0.98	0.06 (1.7)	0.02 (2.2)
$\nu = 0.001, \alpha = 225°$	†	0.98	0.98	0.06 (1.7)	0.02 (2.2)

First, we consider a situation of unidirectional flow, with angle α from the x-axis, that is with constant flow direction $\vec{v}(x,y) = [\cos(\alpha), \sin(\alpha)]$. In Table 4.3 we compare the performance of Gauss–Seidel with SPAI smoothing on a regular 128×128 grid. For diffusion dominated flow, $\nu = 0.1$, the convergence rates obtained either with Gauss–Seidel or with SPAI smoothing are essentially independent of the flow direction.

For convection dominated flow, $\nu \ll h$, the multigrid iteration with Gauss–Seidel smoothing does not converge when the coarse grid operators are computed via Galerkin projection. To obtain a convergent scheme, one needs to compute the coarse grid operators via discrete coarse grid approximation, that is, by discretizing (4.5) explicitly on all grid levels [27, p. 79]. In that situation it is well known that the convergence with Gauss–Seidel smoothing strongly depends on the ordering of the unknowns: the mere reversal of the flow direction (or, equivalently, the ordering of the unknowns) results in a large increase in the convergence rate, from $q = 0.17$ to $q = 0.55$; the contrast becomes even more striking for smaller values of ν. Indeed, when the flow direction is aligned with the ordering of the unknowns, the problem degenerates into a quasi-lower triangular system as $\nu \to 0$. In that situation, the Gauss–Seidel inverse, $(L + D)^{-1}$, essentially yields A^{-1} and thus results in rapid convergence. In contrast, the SPAI smoothers are not affected by the ordering of the unknowns and thus yield identical results in Table 4.3 for both $\alpha = 45°$ and $\alpha = 225°$. For $\nu = 0.001$ the performance of SPAI-1 is poor, while SPAI(ε) with $\varepsilon \leq 0.35$ yields excellent convergence rates at little extra cost.

Next, we consider a situation of rotating flow, where u solves (4.5) with $\vec{v}(x,y) = [y - 1/2, 1/2 - x]$. In this special situation, shown in Figure 4.3, it is generally impossible to reorder the unknowns so that the entire system becomes lower triangular for vanishing ν. As a consequence, the multigrid iteration with Gauss–Seidel smoothing usually diverges for small ν. Convergence can be attained with symmetric Gauss–Seidel smoothing, that is, by repeated application of the Gauss–Seidel iterations in

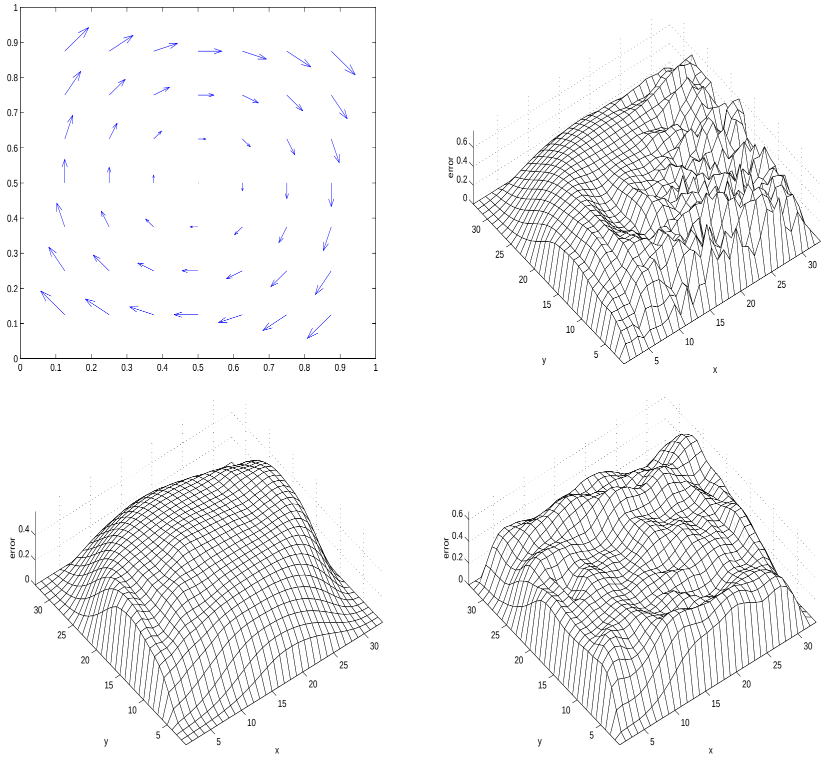


FIG. 4.3. *Rotating flow, $\nu = 10^{-3}$, 32×32 grid: flow field $\vec{v}(x, y)$ (upper left); the effect of three smoothing steps applied to an initially random error: Gauss–Seidel smoothing (upper right); ILU(0) smoothing (lower left); SPAI-1 smoothing (lower right). Note that Gauss–Seidel is unable to smooth the error throughout the domain.*

TABLE 4.4
Diffusion dominated case, $\nu = 0.1$: the convergence rates q obtained with various smoothers. For SPAI(0.35) the relative density is $\rho = 0.7$.

	Gauss–Seidel	SPAI-0	SPAI-1	SPAI(0.35)
64×64	0.04	0.10	0.04	0.07
128×128	0.05	0.10	0.04	0.08
256×256	0.05	0.10	0.04	0.08

natural and reverse ordering of the unknowns [27]; this approach does not generalize easily to unstructured grids. In contrast, SPAI-1 and SPAI(ε) lead to convergent multigrid iterations regardless of the flow pattern or the ordering of the unknowns, and without modifying additional components of the multigrid iteration.

In Table 4.4 we observe that all SPAI smoothers yield h-independent convergence rates in the diffusion dominated case, with $\nu = 0.1$. Although the convergence rate essentially doubles from SPAI-1 to SPAI-0 smoothing, the density ratio ρ drops from 1 to 0.17, which reduces the amount of work in applying the smoother. For SPAI(0.35) the convergence rate lies between those obtained with SPAI-0 and SPAI-1, and so does the relative density $\rho = 0.7$.

Finally, we set $\nu = 10^{-3}$ and thus consider a convection dominated rotating flow, that is, $\nu \ll h$. Both the convergence rates and the relative densities ρ are shown in

Table 4.5 for different smoothers. Neither Gauss–Seidel nor SPAI-0 smoothing lead to a convergent multigrid iteration. Indeed, Gauss–Seidel is unable to smooth the error throughout the domain due to the absence of a dominant "unidirectional" flow direction, as shown in Figure 4.3. Again we note that convergence may be achieved by using symmetric Gauss–Seidel smoothing [27]. Our attempt to use ILU(0) on the 128 × 128 grid eventually failed because of numerical instability encountered in the computation of the ILU decomposition. Alternative variants of ILU would probably work [28].

Table 4.5

Convection dominated case, $\nu = 10^{-3}$: the convergence rates q obtained with various smoothers. The relative density, ρ, is given in parentheses.

	SPAI-1	SPAI (0.5)	SPAI (0.4)	SPAI (0.3)	SPAI (0.2)
128 × 128	0.61	0.67 (0.4)	0.42 (0.6)	0.22 (1.4)	0.09 (3.6)
256 × 256	0.68	0.75 (0.2)	0.45 (0.6)	0.31 (1.3)	0.12 (3.2)

Table 4.5 illustrates the typical behavior of SPAI smoothing versus ε. It shows that reducing ε in the SPAI-Algorithm produces a steady reduction in the convergence rate. Hence a greater reduction of $\|I - MA\|_F$ typically yields an improved smoother. Of course, as ε decreases, both the work in computing and in applying the smoother M rapidly increase, so that the optimal value of ε with the smallest total time probably lies between 0.2 and 0.5 for this problem. For SPAI(0.4) both q and ρ remain essentially constant as the mesh is refined. As we compare the performance of SPAI-1 with that of SPAI(0.4), we remark that both the convergence rate q and the total density ratio ρ are reduced. Therefore the sophisticated search of the original SPAI-Algorithm [14] benefits the smoother by selecting an effective sparsity pattern for M; clearly, the increase in the cost of computing M is worthwhile when memory resources are critical. It may even pay off in reducing total time, since fewer nonzero entries in M results in a cheaper smoother. Again these considerations are problem and architecture dependent; hence it is important to provide the user with a simple but effective way to tune the algorithm, here by adjusting the value of ε.

Table 4.6

Robust convergence with SPAI(0.3) smoothing with respect to ν on a 128×128 grid: both the convergence rate q and the relative density ρ are shown.

	$\nu = 1$	$\nu = 0.1$	$\nu = 0.01$	$\nu = 10^{-3}$	$\nu = 10^{-4}$	$\nu = 10^{-5}$	$\nu = 10^{-6}$
q	0.07	0.07	0.05	0.22	0.73	0.74	0.75
ρ	0.86	0.85	0.92	1.41	2.11	2.29	2.31

What if we decrease ν while keeping the grid spacing fixed? Do we obtain a robust multigrid algorithm (in the sense of [28])? In Table 4.6 we show the convergence rate and density obtained with SPAI (0.3) on the 128 × 128 grid for varying ν. Although both q and ρ vary throughout the range of values for ν, they remain bounded as $\nu \to 0$.

5. Conclusion.

Our numerical results show that sparse approximate inverses based on minimizing the Frobenius norm are an attractive alternative to classical Jacobi or Gauss–Seidel smoothing. For symmetric positive definite problems, SPAI-1 typically behaves like Gauss–Seidel, whereas SPAI-0, which is parameter-free, usually has a slight edge over damped Jacobi with optimal relaxation parameter. Moreover, our proof of the smoothing property for SPAI-0 applies also in situations where that of

Jacobi smoothing cannot be shown. For convection dominated flow problems, such as rotating flow, the ordering independence of SPAI-1 leads to a more robust smoother than Gauss–Seidel. In situations where neither SPAI-0 nor SPAI-1 suffice, SPAI(ε) automatically improves the smoother locally where needed. Both the computation and the application of the SPAI smoothers are inherently parallel.

Nevertheless, our results also show the limitations of SPAI smoothing in difficult situations, such as strong anisotropy, where the lack of isotropic smoothing needs to be compensated by appropriate prolongation and restriction operators. It is very interesting to combine this new hierarchy of local and inherently parallel smoothers with matrix-dependent coarsening strategies, such as found in algebraic multigrid [22]. The first two authors are currently pursuing these issues and will report on them elsewhere [9] in the near future.

The C/MPI version of the SPAI-Algorithm [1, 2] with Matlab and PETSc interfaces can be downloaded from the following address: http://www.sam.math.ethz.ch/˜grote/spai/.

Acknowledgment. We thank the anonymous referees for insightful comments and suggestions.

REFERENCES

[1] S. T. Barnard, L. M. Bernardo, and H. D. Simon, *An MPI implementation of the SPAI preconditioner on the T3E*, Int. J. High Perf. Comput. Appl., 13 (1999), pp. 107–128.

[2] S. T. Barnard and M. J. Grote, *A block version of the SPAI preconditioner*, in Proceedings of the Ninth SIAM Conference on Parallel Processing for Scientific Computing, San Antonio, TX, 1999.

[3] M. W. Benson and P. O. Frederickson, *Iterative solution of large sparse linear systems arising in certain multidimensional approximation problems*, Utilitas Math., 22 (1982), pp. 127–140.

[4] M. W. Benson, *Frequency domain behavior of a set of parallel multigrid smoothing operators*, Int. J. Comput. Math., 36 (1990), pp. 77–88.

[5] M. Benzi, C. D. Meyer, and M. Tůma, *A sparse approximate inverse preconditioner for the conjugate gradient method*, SIAM J. Sci. Comput., 17 (1996), pp. 1135–1149.

[6] M. Benzi and M. Tuma, *A comparative study of sparse approximate inverse preconditioners*, Appl. Numer. Math., 30 (1999), pp. 305–340.

[7] J. Bramble, J. Pasciak, and J. Xu, *Parallel multilevel preconditioners*, Math. Comp., 31 (1990), pp. 333–390.

[8] A. Brandt, *Multi-level adaptive solution to boundary-value problems*, Math. Comp., 31 (1977), pp. 333–390.

[9] O. Bröker and M. J. Grote, *Parallel sparse approximate inverse smoothers for geometric and algebraic multigrid*, Appl. Numer. Math., to appear.

[10] E. Chow and Y. Saad, *Approximate inverse preconditioners via sparse-sparse iterations*, SIAM J. Sci. Comput., 19 (1998), pp. 995–1023.

[11] E. Chow, *A priori sparsity patterns for parallel sparse approximate inverse preconditioners*, SIAM J. Sci. Comput., 21 (2000), pp. 1804–1822.

[12] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 2nd ed., The Johns Hopkins University Press, Baltimore, MD, 1989.

[13] T. Grauschopf, M. Griebel, H. Regler, *Additive multilevel-preconditioners based on bilinear interpolation, matrix-dependent geometric coarsening and algebraic multigrid coarsening for second order elliptic PDEs*, Appl. Numer. Math., 23 (1997), pp. 63–96.

[14] M. J. Grote and T. Huckle, *Parallel preconditioning with sparse approximate inverses*, SIAM J. Sci. Comput., 18 (1997), pp. 838–853.

[15] W. Hackbusch, *Multi-grid Methods and Applications*, Springer-Verlag, Berlin, 1985.

[16] W. Hackbusch, *Iterative Solution of Large Sparse Systems of Equations*, Springer-Verlag, New York, 1994.

[17] T. Huckle, *Approximate sparsity patterns for the inverse of a matrix and preconditioning*, Appl. Numer. Math., 30 (1999), pp. 291–303.

[18] L. Y. Kolotilina and A. Y. Yeremin, *Factorized sparse approximate inverse preconditionings I. Theory*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 45–58.

[19] A. Krechel and K. Stüben, *Operator dependent interpolation in algebraic multigrid*, in Multigrid Methods V, W. Hackbusch and G. Wittum, eds., Lect. Notes Comput. Sci. Eng. 3, Springer-Verlag, Berlin, 1998, pp. 189–211.

[20] A. Reusken, *On a robust multigrid solver*, Computing, 56 (1996), pp. 303–322.

[21] A. Reusken, *On the approximate cyclic reduction preconditioner*, SIAM J. Sci. Comput., 21 (1999), pp. 565–590.

[22] J. W. Ruge and K. Stüben, *Algebraic multigrid*, in Multigrid Methods, S. F. McCormick, ed., SIAM, Philadelphia, 1987, pp. 73–130.

[23] S. Schaffer, *A semicoarsening multigrid method for elliptic partial differential equations with highly discontinuous and anisotropic coefficients*, SIAM J. Sci. Comput., 20 (1998), pp. 228–242.

[24] W.-P. Tang, *Toward an effective approximate inverse preconditioner*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 970–986.

[25] W.-P. Tang and W. L. Wan, *Sparse approximate inverse smoother for multigrid*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1236–1252.

[26] P. Vanek, J. Mandel, and M. Brezina, *Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems*, Computing, 56 (1996), pp. 179–196.

[27] P. Wesseling, *An Introduction to Multigrid Methods*, John Wiley, Chichester, UK, 1992.

[28] G. Wittum, *On the robustness of ILU smoothing*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 699–717.

[29] I. Yavneh and E. Olvovsky, *Multigrid smoothing for symmetric nine-point stencils*, Appl. Math. Comput., 92 (1998), pp. 229–246.

[30] P. M. de Zeeuw, *Matrix prolongations and restrictions in a black-box multigrid solver*, J. Comput. Appl. Math., 33 (1990), pp. 1–27.

SIAM J. SCI. COMPUT.
Vol. 23, No. 4, pp. 1418–1438

DEVELOPMENT OF CFL-FREE, EXPLICIT SCHEMES FOR MULTIDIMENSIONAL ADVECTION-REACTION EQUATIONS*

HONG WANG† AND JIANGGUO LIU†

Abstract. We combine an Eulerian–Lagrangian approach and multiresolution analysis to develop unconditionally stable, explicit, multilevel methods for multidimensional linear hyperbolic equations. The derived schemes generate accurate numerical solutions even if large time steps are used. Furthermore, these schemes have the capability of carrying out adaptive compression without introducing mass balance error. Computational results are presented to show the strong potential of the numerical methods developed.

Key words. advection-reaction equations, characteristic methods, explicit methods, multilevel methods, multiresolution analysis, unconditional stability, wavelet decompositions

AMS subject classifications. 65M25, 65M60, 76M10, 76S05

PII. S1064827500376181

1. Introduction.

Advection-reaction partial differential equations (PDEs) model the reactive transport of solutes in subsurface flows, fluid dynamics, and many other important applications [11, 15]. These equations admit solutions with moving steep fronts, which need to be resolved accurately in applications and often cause severe numerical difficulties. Standard finite difference or finite element methods tend to generate numerical solutions with severe nonphysical oscillations. While upstream weighting methods can eliminate or alleviate these oscillations, they introduce excessive numerical dispersion and grid orientation effects [11, 15]. Most improved methods are explicit, and thus are local, relatively easy to implement, and fully parallelizable. It is well known that there are no explicit, unconditionally stable, consistent finite difference schemes (or virtually any schemes with fixed stencils) for linear hyperbolic PDEs [5]. Consequently, explicit methods are subject to the Courant–Friedrichs–Lewy (CFL) condition and have to use small time steps in numerical simulations to maintain the stability of the methods [11]. On the other hand, implicit methods are unconditionally stable, and so allow large time steps to be used in numerical simulations while still maintaining their stability. But they require inverting a coefficient matrix at each time step in order to generate numerical solutions. The time steps in implicit methods cannot be taken too large due not to the stability constraint but for reasons of accuracy. Local truncation error analysis shows that in implicit methods the temporal errors and spatial errors add up. Thus, the resulting solutions are very sensitive to the time step size.

In recent years, there has been an increasing interest in the application of wavelet techniques in developing efficient numerical schemes for various types of PDEs [3, 7, 14]. For hyperbolic PDEs, the existence of moving steep gradients separating smooth structures is a clear indication that these techniques can be helpful in the design of

*Received by the editors August 4, 2000; accepted for publication (in revised form) July 17, 2001; published electronically December 18, 2001. This research was supported in part by the South Carolina State Commission of Higher Education, a South Carolina Research Initiative grant, ONR grant N00014-94-1-1163, NSF grant DMS-0079549, and the generous awards from Mobil Technology Company and ExxonMobil Upstream Research Company.

http://www.siam.org/journals/sisc/23-4/37618.html

†Department of Mathematics, University of South Carolina, Columbia, SC 29208 (hwang@math.sc.edu, jliu0@math.sc.edu).

efficient numerical techniques for the solution of hyperbolic PDEs. Motivated by these observations, we combine an Eulerian–Lagrangian approach and wavelet techniques to develop unconditionally stable, explicit schemes for multidimensional advection-reaction PDEs, including single-level schemes, multilevel schemes, and adaptive multilevel schemes. These schemes generate accurate numerical solutions even if large time steps are used. Computational results are presented to show the strong potential of the schemes developed.

The rest of this paper is organized as follows. In section 2, we derive a reference equation for the initial-value problems of advection-reaction PDEs in multiple space dimensions. In section 3, we briefly review multiresolution analysis and wavelet decompositions. In section 4, we develop CFL-free, explicit numerical schemes. In section 5, we prove the unconditional stability of these schemes. In section 6, we perform numerical experiments to observe the performance of the schemes and to verify their unconditional stability. In section 7, we outline the extensions of the schemes to the initial-boundary-value problems of advection-reaction PDEs. In section 8, we summarize the results in this paper and draw some conclusions.

2. A reference equation. We consider the initial-value problem for linear advection-reaction PDEs,

$$\begin{aligned} u_t + \nabla\cdot(\mathbf{v}u) + Ru &= q(\boldsymbol{x},t), \qquad &(\boldsymbol{x},t)\in\mathbb{R}^d\times(0,T],\\ u(\boldsymbol{x},0) &= u_0(\boldsymbol{x}), &\boldsymbol{x}\in\mathbb{R}^d, \end{aligned} \tag{2.1}$$

where $\mathbf{v}(\boldsymbol{x},t)$ is a fluid velocity field, $R(\boldsymbol{x},t)$ is a first-order reaction coefficient, $u(\boldsymbol{x},t)$ is the unknown function, and $q(\boldsymbol{x},t)$ and $u_0(\boldsymbol{x})$ are the prescribed source term and initial condition, respectively. We assume that $u_0(\boldsymbol{x})$ and $q(\boldsymbol{x},t)$ have compact support, so the exact solution $u(\boldsymbol{x},t)$ has compact support for any finite time $t>0$.

We define a uniform partition of the time interval $[0,T]$ by $t_n := n\Delta t$ for $n = 0,1,\dots,N$, with $\Delta t := T/N$. If we choose the test functions $w(\boldsymbol{x},t)$ to be of compact support in space, to vanish outside the interval $(t_{n-1},t_n]$, and to be discontinuous in time at time t_{n-1}, the weak formulation for (2.1) is written as

$$\begin{aligned} &\int_{\mathbb{R}^d} u(\boldsymbol{x},t_n)w(\boldsymbol{x},t_n)d\boldsymbol{x} - \int_{t_{n-1}}^{t_n}\int_{\mathbb{R}^d} u(w_t+\mathbf{v}\cdot\nabla w - Rw)(\boldsymbol{x},t)d\boldsymbol{x}dt\\ &\quad = \int_{\mathbb{R}^d} u(\boldsymbol{x},t_{n-1})w(\boldsymbol{x},t_{n-1}^+)d\boldsymbol{x} + \int_{t_{n-1}}^{t_n}\int_{\mathbb{R}^d} q(\boldsymbol{x},t)w(\boldsymbol{x},t)d\boldsymbol{x}dt, \end{aligned} \tag{2.2}$$

where $w(\boldsymbol{x},t_{n-1}^+) := \lim_{t\to t_{n-1}^+} w(\boldsymbol{x},t)$ takes into account the fact that $w(\boldsymbol{x},t)$ is discontinuous in time at time t_{n-1}.

To reflect the hyperbolic nature of (2.1), we follow the ELLAM framework of Celia et al. [1] to choose the test functions w in (2.2) from the solution space of the adjoint equation of (2.1),

$$w_t + \mathbf{v}\cdot\nabla w - Rw = 0. \tag{2.3}$$

Along the characteristic $\boldsymbol{r}(\theta;\check{\boldsymbol{x}},\check{t})$ defined by

$$\frac{d\boldsymbol{r}}{d\theta} = \mathbf{v}(\boldsymbol{r},\theta) \quad \text{with} \quad \boldsymbol{r}(\theta;\check{\boldsymbol{x}},\check{t})|_{\theta=\check{t}} = \check{\boldsymbol{x}}, \tag{2.4}$$

equation (2.3) is rewritten as the following differential equation:

$$\begin{aligned} &\frac{d}{d\theta}w(\boldsymbol{r}(\theta;\check{\boldsymbol{x}},\check{t}),\theta) - R(\boldsymbol{r}(\theta;\check{\boldsymbol{x}},\check{t}),\theta)w(\boldsymbol{r}(\theta;\check{\boldsymbol{x}},\check{t}),\theta) = 0,\\ &\qquad\qquad\qquad\qquad w(\boldsymbol{r}(\theta;\check{\boldsymbol{x}},\check{t}),\theta)|_{\theta=\check{t}} = w(\check{\boldsymbol{x}},\check{t}). \end{aligned} \tag{2.5}$$

Solving this equation yields the following expression for the test functions w:

$$w(\boldsymbol{r}(\theta;\check{\boldsymbol{x}},\check{t}),\theta) = w(\check{\boldsymbol{x}},\check{t})e^{-\int_\theta^{\check{t}} R(\boldsymbol{r}(\gamma;\check{\boldsymbol{x}},\check{t}),\gamma)d\gamma}, \quad \theta \in [t_{n-1},\check{t}], \quad \check{\boldsymbol{x}} \in \mathbb{R}^d. \tag{2.6}$$

Choosing $(\check{\boldsymbol{x}},\check{t}) = (\boldsymbol{x},t_n)$, we see that once the test functions $w(\boldsymbol{x},t_n)$ are specified at time t_n, they are determined completely on the space-time strip $\mathbb{R}^d \times (t_{n-1},t_n]$ with an exponential variation along the characteristic $\boldsymbol{r}(\theta;\boldsymbol{x},t_n)$ for $\theta \in [t_{n-1},t_n]$.

In the numerical schemes we use an Euler or Runge–Kutta formula to approximate the characteristic $\boldsymbol{r}(\theta;\check{\boldsymbol{x}},\check{t})$ and use the following approximate test functions w instead:

$$w(\boldsymbol{r}(\theta;\boldsymbol{x},t_n),\theta) = w(\boldsymbol{x},t_n)e^{-R(\boldsymbol{x},t_n)(t_n-\theta)}, \quad \theta \in [t_{n-1},t_n], \quad \boldsymbol{x} \in \mathbb{R}^d. \tag{2.7}$$

To avoid confusion in the derivation, we replace the dummy variables $\boldsymbol{x}$ and t in the second term on the right-hand side of (2.2) by $\boldsymbol{y}$ and θ and reserve $\boldsymbol{x}$ for the variable in $\mathbb{R}^d$ at time t_n. For any $\boldsymbol{y} \in \mathbb{R}^d$, there exists an $\boldsymbol{x} \in \mathbb{R}^d$ such that $\boldsymbol{y} = \boldsymbol{r}(\theta;\boldsymbol{x},t_n)$. We obtain

$$\begin{aligned}
&\int_{t_{n-1}}^{t_n}\int_{\mathbb{R}^d} q(\boldsymbol{y},\theta)w(\boldsymbol{y},\theta)d\boldsymbol{y}d\theta \\
&\quad = \int_{\mathbb{R}^d}\int_{t_{n-1}}^{t_n} q(\boldsymbol{r}(\theta;\boldsymbol{x},t_n),\theta)w(\boldsymbol{r}(\theta;\boldsymbol{x},t_n),\theta) \left|\frac{\partial \boldsymbol{r}(\theta;\boldsymbol{x},t_n)}{\partial \boldsymbol{x}}\right| d\theta d\boldsymbol{x} \\
&\quad = \int_{\mathbb{R}^d} q(\boldsymbol{x},t_n)w(\boldsymbol{x},t_n)\left[\int_{t_{n-1}}^{t_n} e^{-R(\boldsymbol{x},t_n)(t_n-\theta)}d\theta\right] d\boldsymbol{x} + E_1(q,w) \\
&\quad = \int_{\mathbb{R}^d} \Lambda(\boldsymbol{x},t_n)q(\boldsymbol{x},t_n)w(\boldsymbol{x},t_n)d\boldsymbol{x} + E_1(q,w).
\end{aligned} \tag{2.8}$$

Here $|\frac{\partial \boldsymbol{r}(\theta;\boldsymbol{x},t_n)}{\partial \boldsymbol{x}}|$ is the Jacobian determinant of the transformation from $\boldsymbol{x}$ at time t_n to $\boldsymbol{r}(\theta;\boldsymbol{x},t_n)$ at time θ.

$$\Lambda(\boldsymbol{x},t_n) := \begin{cases} \dfrac{1-e^{-R(\boldsymbol{x},t_n)\Delta t}}{R(\boldsymbol{x},t_n)} & \text{if } R(\boldsymbol{x},t_n) \neq 0, \\ \Delta t & \text{otherwise} \end{cases} \tag{2.9}$$

and

$$\begin{aligned}
E_1(w) := \int_{\mathbb{R}^d}\int_{t_{n-1}}^{t_n} \Bigg[& q(\boldsymbol{r}(\theta;\boldsymbol{x},t_n),\theta)\Big|\frac{\partial \boldsymbol{r}(\theta;\boldsymbol{x},t_n)}{\partial \boldsymbol{x}}\Big| \\
& - q(\boldsymbol{x},t_n)\Bigg] w(\boldsymbol{x},t_n)e^{-R(\boldsymbol{x},t_n)(t_n-\theta)}d\theta d\boldsymbol{x}.
\end{aligned} \tag{2.10}$$

Incorporating (2.8) into (2.2), we obtain a reference equation

$$\begin{aligned}
\int_{\mathbb{R}^d} u(\boldsymbol{x},t_n)w(\boldsymbol{x},t_n)d\boldsymbol{x} = & \int_{\mathbb{R}^d} u(\boldsymbol{x},t_{n-1})w(\boldsymbol{x},t_{n-1}^+)d\boldsymbol{x} \\
& + \int_{\mathbb{R}^d} \Lambda(\boldsymbol{x},t_n)q(\boldsymbol{x},t_n)w(\boldsymbol{x},t_n)d\boldsymbol{x} + E(w),
\end{aligned} \tag{2.11}$$

where

$$E(w) := \int_{t_{n-1}}^{t_n}\int_{\mathbb{R}^d} u(w_t + \mathbf{v}\cdot\nabla w - Rw)(\boldsymbol{x},t)d\boldsymbol{x}dt + E_1(q,w).$$

Previously, the so-called ELLAM schemes were developed by using finite element basis functions in (2.11). These schemes symmetrize the governing transport PDEs and generate accurate numerical solutions even if very coarse spatial grids and time steps are used.

3. Multiresolution analysis and wavelet decompositions.

To develop numerical schemes based on (2.11), we need to define the trial and test functions at time t_n. To do so, we briefly recall multiresolution analysis and wavelet decompositions.

3.1. Multiresolution analysis.

In the standard Fourier analysis, L^2-functions are represented by linear combinations of sines and cosines. In 1910, Haar studied the representation of L^2-functions by step functions taking values ±1 [12]. In the 1980s, these ideas were explored and developed further into the theory of wavelets. The first wavelets were introduced in early 1980s by Stromberg [21] and Morlet et al. [18]. Multiresolution analysis, which is one of the best ways of constructing wavelets, began in image processing [20, 22] and was introduced into mathematics by Mallat [16]. Daubechies used multiresolution analysis to construct compactly supported orthogonal wavelets with arbitrary smoothness, which include the Haar wavelets as the simplest case [8]. We refer readers to the survey article [10] for detailed reviews.

A sequence of closed subspaces $\{\mathcal{V}_j\}_{j\in\mathbb{Z}}$ ($\mathbb{Z}$ is the set of all integers) of $L^2(\mathbb{R})$ is a *multiresolution analysis* if

(a) these spaces are nested: $\mathcal{V}_j \subset \mathcal{V}_{j+1}\ \forall j \in \mathbb{Z}$;

(b) these spaces are dense in $L^2(\mathbb{R})$: $\overline{\cup}_{j\in\mathbb{Z}}\mathcal{V}_j = L^2(\mathbb{R})$ and $\cap_{j\in\mathbb{Z}}\mathcal{V}_j = \emptyset$;

(c) $\mathcal{V}_0$ is invariant under integer shifts: $f \in \mathcal{V}_0 \Longrightarrow f(\cdot - k) \in \mathcal{V}_0\ \forall k \in \mathbb{Z}$;

(d) $\mathcal{V}_j$ is obtained from $\mathcal{V}_0$ by dilation: $f(\cdot) \in \mathcal{V}_j \Longleftrightarrow f(2^{-j}\cdot) \in \mathcal{V}_0\ \forall j \in \mathbb{Z}$;

(e) $\mathcal{V}_0$ is generated by a single (scaling) function ϕ and its translates $\{\phi_{0,k} : k \in \mathbb{Z}\}$, where

$$\phi_{j,k}(x) := 2^{j/2}\phi(2^j x - k), \quad j,k \in \mathbb{Z}. \tag{3.1}$$

Because $\mathcal{V}_0 \subset \mathcal{V}_1$, the scaling function $\phi \in \mathcal{V}_0$ is also a member of $\mathcal{V}_1$. Hence, the following refinement relation holds:

$$\phi = \sum_{k\in\mathbb{Z}} h_k \phi_{1,k}. \tag{3.2}$$

In general, the sum has infinitely many terms, and convergence in (3.2) is understood in the $L^2(\mathbb{R})$-norm. Daubechies first discovered a family of compactly supported orthogonal wavelets [8], so their filters h_k have only finite length. The coiflets developed subsequently have improved symmetry and regularity [9]. In this paper we use compactly supported orthogonal wavelets.

Let $\mathcal{P}_j : L^2(\mathbb{R}) \longrightarrow \mathcal{V}_j$ be the orthogonal projection operator; we then have

$$\mathcal{P}_j f = \sum_{k\in\mathbb{Z}} c_{j,k}(f)\ \phi_{j,k} \qquad \text{with} \quad c_{j,k}(f) := \int_{\mathbb{R}} f(x)\phi_{j,k}(x)dx. \tag{3.3}$$

Let $\mathcal{W}_{j-1}$ be the orthogonal complement of $\mathcal{V}_{j-1}$ in $\mathcal{V}_j$. Then we have the following decomposition:

$$\begin{aligned} \mathcal{V}_j &= \mathcal{V}_{j-1} \oplus \mathcal{W}_{j-1} = \cdots \\ &= \mathcal{V}_{j_c} \oplus \mathcal{W}_{j_c} \oplus \mathcal{W}_{j_c+1} \oplus \cdots \oplus \mathcal{W}_{j-1} \quad \text{for } j > j_c. \end{aligned} \tag{3.4}$$

It is proved that the spaces $\mathcal{W}_j$ can be generated by a single (wavelet) function ψ [8, 9]. In other words, ψ and its integer translates $\psi_{0,k}$, with $\psi_{j,k}$ being defined by

$$\psi_{j,k}(x) := 2^{j/2}\psi(2^j x - k), \quad j,k \in \mathbb{Z}, \tag{3.5}$$

constitute an orthonormal basis for $\mathcal{W}_0$. Hence, for each fixed j, the $\psi_{j,k}$ ($k \in \mathbb{Z}$) form an orthogonal basis for $\mathcal{W}_j$. Since $\psi \in \mathcal{W}_0 \subset \mathcal{V}_1$, it can be expressed as

$$\psi = \sum_{k\in\mathbb{Z}} g_k \phi_{1,k}. \tag{3.6}$$

A permissible choice for the filter g_k is given by $g_k = (-1)^{k-1} h_{-k-1}$ [9].

Let $\mathcal{Q}_j : L^2(\mathbb{R}) \longrightarrow \mathcal{W}_j$ be the orthogonal projection operator

$$\mathcal{Q}_j f = \sum_{k\in\mathbb{Z}} d_{j,k}(f)\ \psi_{j,k} \qquad \text{with} \quad d_{j,k}(f) := \int_{\mathbb{R}} f(x)\psi_{j,k}(x)dx. \tag{3.7}$$

For $f \in L^2(\mathbb{R})$, the telescoping sum

$$\mathcal{P}_{j_f} f = \mathcal{P}_{j_c} f + \sum_{j=j_c}^{j_f-1} (\mathcal{P}_{j+1} f - \mathcal{P}_j f) = \mathcal{P}_{j_c} f + \sum_{j=j_c}^{j_f-1} \mathcal{Q}_j f \tag{3.8}$$

represents the projection $\mathcal{P}_{j_f} f \in \mathcal{V}_{j_f}$ of f at a fine level j_f as a direct sum of $\mathcal{P}_{j_c} f \in \mathcal{V}_{j_c}$ of f at a coarse level j_c and the elements in a sequence of refined spaces $\mathcal{W}_{j_c} \oplus \mathcal{W}_{j_c+1} \oplus \cdots \oplus \mathcal{W}_{j_f-1}$ that provide progressively improved resolution at different scales.

3.2. Cascade algorithm. The cascade algorithm provides an efficient approach for decomposition and reconstruction [9]. Using the refinement equation (3.2) and the definition (3.1) of $\phi_{j,k}(x)$, we see

$$\begin{aligned}
\phi_{j-1,k}(x) &= 2^{(j-1)/2}\phi(2^{j-1}x - k) \\
&= 2^{(j-1)/2} \sum_{l\in\mathbb{Z}} h_l 2^{1/2} \phi(2(2^{j-1}x - k) - l) \\
&= 2^{j/2} \sum_{l\in\mathbb{Z}} h_l \phi(2^j x - 2k - l) \\
&= \sum_{l\in\mathbb{Z}} h_l \phi_{j,l+2k}(x) \\
&= \sum_{l\in\mathbb{Z}} h_{l-2k} \phi_{j,l}(x).
\end{aligned} \tag{3.9}$$

Similarly, we have

$$\psi_{j-1,k}(x) = \sum_{l\in\mathbb{Z}} g_{l-2k} \phi_{j,l}(x). \tag{3.10}$$

In the decomposition process, the cascade algorithm shows how to calculate the scaling coefficients $c_{j-1,k}(f)$ and the wavelet coefficients $d_{j-1,k}(f)$ at a coarser level $j-1$ from the coefficients $c_{j,k}(f)$ at a finer level j:

$$\begin{aligned}
c_{j-1,k}(f) &= \int_{\mathbb{R}} f(x)\phi_{j-1,k}(x)dx = \int_{\mathbb{R}} f(x) \sum_{l\in\mathbb{Z}} h_{l-2k}\phi_{j,l}(x)dx \\
&= \sum_{l\in\mathbb{Z}} h_{l-2k} \int_{\mathbb{R}} f(x)\phi_{j,l}(x)dx = \sum_{l\in\mathbb{Z}} c_{j,l}(f) h_{l-2k}, \\
d_{j-1,k}(f) &= \int_{\mathbb{R}} f(x)\psi_{j-1,k}(x)dx = \int_{\mathbb{R}} f(x) \sum_{l\in\mathbb{Z}} g_{l-2k}\phi_{j,l}(x)dx \\
&= \sum_{l\in\mathbb{Z}} g_{l-2k} \int_{\mathbb{R}} f(x)\phi_{j,l}(x)dx = \sum_{l\in\mathbb{Z}} c_{j,l}(f) g_{l-2k}.
\end{aligned} \tag{3.11}$$

In the reconstruction process, the cascade algorithm shows how to calculate the scaling coefficients $c_{j,k}(f)$ at a finer level j from the scaling coefficients $c_{j-1,k}(f)$ and the wavelet coefficients $d_{j-1,k}(f)$ at a coarser level $j-1$. Using (3.3), (3.7)–(3.10), we have

$$
\begin{aligned}
c_{j,k}(f) &= \int_{\mathbb{R}} f(x)\phi_{j,k}(x)dx = \int_{\mathbb{R}} \mathcal{P}_j f(x)\phi_{j,k}(x)dx \\
&= \int_{\mathbb{R}} \Big[\mathcal{P}_{j-1}f(x) + \mathcal{Q}_{j-1}f(x)\Big]\phi_{j,k}(x)dx \\
&= \int_{\mathbb{R}} \left[\sum_{l\in\mathbb{Z}} c_{j-1,l}(f)\phi_{j-1,l}(x) + \sum_{l\in\mathbb{Z}} d_{j-1,l}(f)\psi_{j-1,l}(x)\right]\phi_{j,k}(x)dx \\
&= \sum_{l\in\mathbb{Z}} c_{j-1,l}(f)\int_{\mathbb{R}} \phi_{j-1,l}(x)\phi_{j,k}(x)dx \\
&\quad + \sum_{l\in\mathbb{Z}} d_{j-1,l}(f)\int_{\mathbb{R}} \psi_{j-1,l}(x)\phi_{j,k}(x)dx \\
&= \sum_{l\in\mathbb{Z}} c_{j-1,l}(f)\int_{\mathbb{R}} \left[\sum_{i\in\mathbb{Z}} h_{i-2l}\phi_{j,i}(x)\right]\phi_{j,k}(x)dx \\
&\quad + \sum_{l\in\mathbb{Z}} d_{j-1,l}(f)\int_{\mathbb{R}} \left[\sum_{i\in\mathbb{Z}} g_{i-2l}\phi_{j,i}(x)\right]\phi_{j,k}(x)dx \\
&= \sum_{l\in\mathbb{Z}}\sum_{i\in\mathbb{Z}} c_{j-1,l}(f)h_{i-2l}\delta_{i,k} + \sum_{l\in\mathbb{Z}}\sum_{i\in\mathbb{Z}} d_{j-1,l}(f)g_{i-2l}\delta_{i,k} \\
&= \sum_{l\in\mathbb{Z}} c_{j-1,l}(f)h_{k-2l} + \sum_{l\in\mathbb{Z}} d_{j-1,l}(f)g_{k-2l}.
\end{aligned} \tag{3.12}
$$

Here $\delta_{i,k}$ is the Dirac delta function, $\delta_{i,k} = 1$ if $i = k$, or 0 otherwise.

4. Unconditionally stable, explicit schemes.

For simplicity, we assume that the support of the solution $u(\boldsymbol{x}, t)$ is contained inside the spatial domain $\Omega := (a_1, b_1)\times\cdots\times(a_d, b_d)$ during the time period $[0, T]$. We will outline the extensions of the schemes developed in this section to initial-boundary-value problems of advection-reaction PDEs in section 7.

Let $N_{0,m} \in \mathbb{N}$ $(m = 1, 2, \ldots, d)$ be the numbers of intervals in the mth coordinate directions. We define spatial grids at the coarsest occurring level 0 by

$$x_{0,k}^{(m)} := a_m + kh_{0,m} \text{ with } h_{0,m} := \frac{b_m - a_m}{N_{0,m}}, \quad 0 \le k \le N_{0,m},\ 1 \le m \le d. \tag{4.1}$$

Using the scaling function $\phi(x)$ in (3.1), we define

$$\phi_{0,k}^{(m)}(x) := h_{0,m}^{-1/2}\phi\left(\frac{x - x_{0,k}^{(m)}}{h_{0,m}}\right) \tag{4.2}$$

to be the scaling functions at the level 0 that are associated with the grids $x_{0,k}^{(m)}$ in the mth coordinate direction. We then define the corresponding functions and grids at level $j = 1, 2, \dots, J$:

$$\begin{aligned} N_{j,m} &:= 2N_{j-1,m} = \cdots = 2^j N_{0,m}, \\ h_{j,m} &:= \tfrac{1}{2} h_{j-1,m} = \cdots = 2^{-j} h_{0,m}, \\ x_{j,k}^{(m)} &:= a_m + kh_{j,m}, \\ \phi_{j,k}^{(m)}(x) &:= 2^{j/2} \phi_{0,0}^{(m)}(2^j x - k), \\ 0 \le k \le N_{j,m}&, \quad 1 \le m \le d, \quad 1 \le j \le J, \end{aligned} \tag{4.3}$$

where levels 0 and J denote the coarsest and finest occurring discretization levels.

4.1. A single-level scheme. We define finite-dimensional spaces $\mathcal{S}_j(\Omega)$ at level j by

$$\mathcal{S}_j(\Omega) := \operatorname{span} \{\Phi_{j,\boldsymbol{k}}(\boldsymbol{x})\}_{\boldsymbol{k} \in \boldsymbol{\omega}_j}, \tag{4.4}$$

where the scaling functions $\Phi_{j,\boldsymbol{k}}(\boldsymbol{x})$, with $\boldsymbol{x} = (x_1, \dots, x_d)$, and the index sets $\boldsymbol{\omega}_j$ at level j are defined by

$$\begin{aligned} \Phi_{j,\boldsymbol{k}}(\boldsymbol{x}) &:= \prod_{m=1}^{d} \phi_{j,k_m}^{(m)}(x_m), \\ \boldsymbol{\omega}_j &:= \left\{ \boldsymbol{k} = (k_1, \dots, k_d) \in \mathbb{N}^d \,\middle|\, 0 \le k_m \le N_{j,m},\ 1 \le m \le d \right\}. \end{aligned} \tag{4.5}$$

Replacing the exact solution u in (2.11) by the trial functions $U(\boldsymbol{x}, t_n) \in \mathcal{S}_J(\Omega)$ and dropping the error term $E(w)$ in (2.11), we obtain the following.

Scheme I. Seek $U(\boldsymbol{x}, t_n) \in \mathcal{S}_J(\Omega)$, the space of the scaling functions defined on the finest occurring discretization level J, with

$$U(\boldsymbol{x}, t_n) = \sum_{\boldsymbol{k} \in \boldsymbol{\omega}_J} c_{J,\boldsymbol{k}}^n \Phi_{J,\boldsymbol{k}}(\boldsymbol{x}), \tag{4.6}$$

such that the following equation holds for any $w(\boldsymbol{x}, t_n) = \Phi_{J,\boldsymbol{k}}(\boldsymbol{x})$ with $\boldsymbol{k} \in \boldsymbol{\omega}_J$:

$$\begin{aligned} &\int_\Omega U(\boldsymbol{x}, t_n) w(\boldsymbol{x}, t_n) d\boldsymbol{x} \\ &\quad = \int_\Omega U(\boldsymbol{x}, t_{n-1}) w(\boldsymbol{x}, t_{n-1}^+) d\boldsymbol{x} + \int_\Omega \Lambda(\boldsymbol{x}, t_n) q(\boldsymbol{x}, t_n) w(\boldsymbol{x}, t_n) d\boldsymbol{x}. \end{aligned} \tag{4.7}$$

This scheme is explicit, since choosing $w(\boldsymbol{x}, t_n) = \Phi_{j,\boldsymbol{k}}(\boldsymbol{x})$ in (4.7) reduces its left-hand side to

$$\int_\Omega U(\boldsymbol{x}, t_n) w(\boldsymbol{x}, t_n) d\boldsymbol{x} = c_{J,\boldsymbol{k}}^n. \tag{4.8}$$

The evaluation of the second term on the right-hand side of (4.7) is standard in wavelet methods for elliptic and parabolic PDEs [3, 6, 7, 14]. However, the evaluation of the first term on the right-hand side of (4.7) in nonconventional, due to the definition of $w(\boldsymbol{x}, t_{n-1}^+)$ that in turn results from the characteristic tracking. In the current context, we adopt a modified forward tracking algorithm [19] to evaluate this term. We would enforce a Simpson or fifth-order Newton–Cotes integration quadrature on each cell (or dyadic subcells) at time step t_{n-1}, which needs only the dyadic values of the wavelets that can be obtained a priori. To evaluate $w(\boldsymbol{x}, t_{n-1}^+)$, we track these discrete quadrature points $\boldsymbol{x}_q$ forward along the characteristics defined by (2.4) to time step t_n and obtain $\tilde{\boldsymbol{x}}_q = \boldsymbol{r}(t_n; \boldsymbol{x}_q, t_{n-1})$. We then use (2.7) to evaluate

$$w(\boldsymbol{x}_q, t_{n-1}^+) = w(\tilde{\boldsymbol{x}}_q, t_n) e^{-R(\tilde{\boldsymbol{x}}_q, t_n)\Delta t}. \tag{4.9}$$

Note that $\tilde{\boldsymbol{x}}_q$ is not necessarily a dyadic point in general. We compute $w(\tilde{\boldsymbol{x}}_q, t_n)$ by an interpolation based on its values at neighboring dyadic points.

4.2. A multilevel scheme. Notice that, as in the case of Fourier series, when the exact solution $u(\boldsymbol{x}, t_n)$ is smooth, its wavelet coefficients

$$d_{j,\boldsymbol{k}}^{n,\boldsymbol{e}}(u) = \int_\Omega u(\boldsymbol{x}, t_n) \Psi_{j,\boldsymbol{k}}^{\boldsymbol{e}}(\boldsymbol{x}) d\boldsymbol{x}$$

decay rapidly as the level j increases [9, 10]. Here the wavelets $\Psi_{j,\boldsymbol{k}}^{\boldsymbol{e}}(\boldsymbol{x})$ are defined by

$$\Psi_{j,\boldsymbol{k}}^{\boldsymbol{e}}(\boldsymbol{x}) = \prod_{m=1}^{d} \left(\phi_{j,k_m}^{(m)}(x_m)\right)^{1-e_m} \left(\psi_{j,k_m}^{(m)}(x_m)\right)^{e_m} \qquad \forall \boldsymbol{e} \in \boldsymbol{E}, \tag{4.10}$$

where $\boldsymbol{E}^{'} := \{0,1\}^d = \{\boldsymbol{e} = (e_1, \ldots, e_d) \mid e_i = 0, 1\}$ is the set of vertices of the d-dimensional unit cube, $\boldsymbol{E} = \boldsymbol{E}^{'} \backslash \{\boldsymbol{0}\}$, and

$$\begin{aligned} \psi_{0,k}^{(m)}(x) &:= h_{0,m}^{-1/2} \psi\left(\frac{x - x_{0,k}^{(m)}}{h_{0,m}}\right), \\ \psi_{j,k}^{(m)}(x) &:= 2^{j/2} \psi_{0,0}^{(m)}(2^j x - k). \end{aligned} \tag{4.11}$$

For example, when $u(\boldsymbol{x}, t_n)$ is differentiable we have

$$\begin{aligned} \left|d_{j,\boldsymbol{k}}^{n,\boldsymbol{e}}(u)\right| &= \left|\int_\Omega u(\boldsymbol{x}, t_n) \Psi_{j,\boldsymbol{k}}^{\boldsymbol{e}}(\boldsymbol{x}) d\boldsymbol{x}\right| \\ &= \inf_{c \in \mathbb{R}} \left|\int_\Omega \left[u(\boldsymbol{x}, t_n) - c\right] \Psi_{j,\boldsymbol{k}}^{\boldsymbol{e}}(\boldsymbol{x}) d\boldsymbol{x}\right| \\ &\le \inf_{c \in \mathbb{R}} \|u(\boldsymbol{x}, t_n) - c\|_{L^2(\Omega_{j,\boldsymbol{k}}^{\boldsymbol{e}})} \\ &\le C 2^{-j/2} \|\nabla u(\boldsymbol{x}, t_n)\|_{L^2(\Omega_{j,\boldsymbol{k}}^{\boldsymbol{e}})}, \end{aligned} \tag{4.12}$$

with $\Omega_{j,\boldsymbol{k}}^{\boldsymbol{e}} = \mathrm{supp}(\Psi_{j,\boldsymbol{k}}^{\boldsymbol{e}})$. By using a multidimensional analogue of expansion (3.8), we have the following multiresolution expansion for $U(\boldsymbol{x}, t_n) \in \mathcal{S}_{J_n}(\Omega)$:

$$U(\boldsymbol{x}, t_n) = \sum_{\boldsymbol{k} \in \boldsymbol{\omega}_0} c_{0,\boldsymbol{k}}^n \Phi_{0,\boldsymbol{k}}(\boldsymbol{x}) + \sum_{j=0}^{J_n - 1} \sum_{\boldsymbol{k} \in \boldsymbol{\omega}_j} \sum_{\boldsymbol{e} \in \boldsymbol{E}} d_{j,\boldsymbol{k}}^{n,\boldsymbol{e}} \Psi_{j,\boldsymbol{k}}^{\boldsymbol{e}}(\boldsymbol{x}). \tag{4.13}$$

We define a CFL-free, explicit, multilevel scheme for problem (2.1) as follows.

Scheme II. Find $U(\boldsymbol{x}, t_n) \in \mathcal{S}_{J_n}(\Omega)$, which is in the form of (4.13) with $0 \le J_n \le J$, such that (4.7) holds for any $w(\boldsymbol{x}, t_n) \in \mathcal{S}_{J_n}(\Omega)$.

Scheme II possesses all the numerical advantages of Scheme I. For example, it is explicit and is a multilevel scheme. In fact, if we choose $w(\boldsymbol{x}, t_n) = \Phi_{0,\boldsymbol{k}}(\boldsymbol{x})$ or $\Psi^{\boldsymbol{e}}_{j,\boldsymbol{k}}(\boldsymbol{x})$ for $\boldsymbol{k} \in \boldsymbol{\omega}_j$, $\boldsymbol{e} \in \boldsymbol{E}$, and $j = 0, 1, \ldots, J_n - 1$, the left-hand side of (4.7) is reduced to

$$\begin{aligned}\int_\Omega U(\boldsymbol{x}, t_n)\Phi_{0,\boldsymbol{k}}(\boldsymbol{x})d\boldsymbol{x} &= c^n_{0,\boldsymbol{k}}, \quad \boldsymbol{k} \in \boldsymbol{\omega}_0,\\ \int_\Omega U(\boldsymbol{x}, t_n)\Psi^{\boldsymbol{e}}_{j,\boldsymbol{k}}(\boldsymbol{x})d\boldsymbol{x} &= d^{n,\boldsymbol{e}}_{j,\boldsymbol{k}}, \quad \boldsymbol{e} \in \boldsymbol{E}, \quad \boldsymbol{k} \in \boldsymbol{\omega}_j, \quad 0 \le j \le J_n - 1.\end{aligned} \tag{4.14}$$

Equations (4.13) and (4.14) generate a multilevel expression of the solution $U(\boldsymbol{x}, t_n)$. Algorithmically, we first compute the coefficients $c^n_{0,\boldsymbol{k}}$ in (4.13) at the coarsest level 0. We then compute the coefficients $d^{n,\boldsymbol{e}}_{j,\boldsymbol{k}}$ for $\boldsymbol{e} \in \boldsymbol{E}$ and $\boldsymbol{k} \in \boldsymbol{\omega}_j$ in the second term on the right-hand side of (4.13) starting from the coarsest occurring level 0 to the finest level J. Finally, because all the wavelets have at least zeroth-order vanishing moments, adding the wavelet expressions level by level does not affect mass balance.

4.3. A CFL-free, explicit multilevel scheme with adaptive and conservative compression. In applications hyperbolic PDEs often admit solutions with very localized phenomena, which are typically smooth outside some very small (but dynamic) regions and could develop steep fronts within these regions. In contrast to Fourier series expansions where local singularities of the solutions could contaminate the decaying properties of Fourier coefficients globally, (4.12) shows that the wavelet coefficients actually become small when the underlying solution is smooth locally. Therefore, we can drop the terms with small wavelet coefficients to reduce the number of unknowns to be solved without introducing large errors. On the other hand, the wavelet coefficients also indicate where relevant detailed information is encountered. This observation motivates the development of a CFL-free, explicit multilevel scheme with the capability for adaptive and conservative compression to fully utilize these properties.

Scheme III. This scheme is divided into four steps.

Step 1. *Initialization.* Project the initial condition $u_0(\boldsymbol{x})$ into the space $\mathcal{S}_J(\Omega)$ to obtain its approximation,

$$U(\boldsymbol{x}, t_0) = \sum_{\boldsymbol{k}\in\boldsymbol{\omega}_0} c^0_{0,\boldsymbol{k}}\Phi_{0,\boldsymbol{k}}(\boldsymbol{x}) + \sum_{j=0}^{J-1}\sum_{\boldsymbol{k}\in\boldsymbol{\omega}_j}\sum_{\boldsymbol{e}\in\boldsymbol{E}} d^{0,\boldsymbol{e}}_{j,\boldsymbol{k}}\Psi^{\boldsymbol{e}}_{j,\boldsymbol{k}}(\boldsymbol{x}), \tag{4.15}$$

with $c^0_{0,\boldsymbol{k}}$ and $d^{0,\boldsymbol{e}}_{j,\boldsymbol{k}}$ being computed by

$$\begin{aligned}c^n_{0,\boldsymbol{k}} &= \int_\Omega u_0(\boldsymbol{x})\Phi_{0,\boldsymbol{k}}(\boldsymbol{x})d\boldsymbol{x}, \quad \boldsymbol{k} \in \boldsymbol{\omega}_0,\\ d^{n,\boldsymbol{e}}_{j,\boldsymbol{k}} &= \int_\Omega u_0(\boldsymbol{x})\Psi^{\boldsymbol{e}}_{j,\boldsymbol{k}}(\boldsymbol{x})d\boldsymbol{x}, \quad \boldsymbol{k} \in \boldsymbol{\omega}_j, \quad \boldsymbol{e} \in \boldsymbol{E}, \quad 0 \le j \le J - 1.\end{aligned} \tag{4.16}$$

We use a time marching algorithm to perform the following steps for $n = 1, 2, \ldots, N$.

Step 2. *Compression step.* Because the wavelet coefficients in the expression

$$U(\boldsymbol{x}, t_{n-1}) = \sum_{\boldsymbol{k}\in\boldsymbol{\omega}_0} c^{n-1}_{0,\boldsymbol{k}}\Phi_{0,\boldsymbol{k}}(\boldsymbol{x}) + \sum_{j=0}^{J-1}\sum_{\boldsymbol{k}\in\widetilde{\boldsymbol{\omega}}^{n-1}_j}\sum_{\boldsymbol{e}\in\boldsymbol{E}} d^{n-1,\boldsymbol{e}}_{j,\boldsymbol{k}}\Psi^{\boldsymbol{e}}_{j,\boldsymbol{k}}(\boldsymbol{x}) \tag{4.17}$$

would be nearly zero in the smooth regions of $U(\boldsymbol{x}, t_{n-1})$, and would be noticeable in the rough regions of $U(\boldsymbol{x}, t_{n-1})$, we can drop many small wavelet coefficients in (4.17) without affecting the accuracy of the solution $U(\boldsymbol{x}, t_n)$. Here $\widetilde{\boldsymbol{\omega}}_j^{n-1}$ are the predicted significant coefficient index sets with $\widetilde{\boldsymbol{\omega}}_j^0 = \boldsymbol{\omega}_j$ and $\widetilde{\boldsymbol{\omega}}_j^{n-1}$ being defined below for subsequent time steps.

We define level-dependent thresholding parameters ε_j by

$$\varepsilon_j := 2^{-jd/2}\ \Delta t \left[\|u_0\|_{L^2(\Omega)} + \|q(\cdot, t_n)\|_{L^2(\Omega)}\right] \tag{4.18}$$

and perform the following thresholding procedure:

$$\widehat{d}_{j,\boldsymbol{k}}^{n-1,\boldsymbol{e}} := \begin{cases} d_{j,\boldsymbol{k}}^{n-1,\boldsymbol{e}} & \text{if } \left|d_{j,\boldsymbol{k}}^{n-1,\boldsymbol{e}}\right| \geq \varepsilon_j, \\ 0 & \text{otherwise.} \end{cases} \tag{4.19}$$

Equivalently, we introduce the significant coefficient index sets

$$\widehat{\boldsymbol{\omega}}_j^{n-1} := \left\{\boldsymbol{k} \in \widetilde{\boldsymbol{\omega}}_j^{n-1} \;\middle|\; \exists \boldsymbol{e} \in \boldsymbol{E} \text{ such that } \left|d_{j,\boldsymbol{k}}^{n-1,\boldsymbol{e}}\right| \geq \varepsilon_j \right\}. \tag{4.20}$$

We define a compression $\widehat{U}(\boldsymbol{x}, t_{n-1})$ of $U(\boldsymbol{x}, t_{n-1})$ by

$$\begin{aligned} \widehat{U}(\boldsymbol{x}, t_{n-1}) &= \sum_{\boldsymbol{k}\in\boldsymbol{\omega}_0} c_{0,\boldsymbol{k}}^{n-1}\Phi_{0,\boldsymbol{k}}(\boldsymbol{x}) + \sum_{j=0}^{J-1} \sum_{\boldsymbol{k}\in\widetilde{\boldsymbol{\omega}}_j^{n-1}} \sum_{\boldsymbol{e}\in\boldsymbol{E}} \widehat{d}_{j,\boldsymbol{k}}^{n-1,\boldsymbol{e}} \Psi_{j,\boldsymbol{k}}^{\boldsymbol{e}}(\boldsymbol{x}) \\ &= \sum_{\boldsymbol{k}\in\boldsymbol{\omega}_0} c_{0,\boldsymbol{k}}^{n-1}\Phi_{0,\boldsymbol{k}}(\boldsymbol{x}) + \sum_{j=0}^{J-1} \sum_{\boldsymbol{k}\in\widehat{\boldsymbol{\omega}}_j^{n-1}} \sum_{\boldsymbol{e}\in\boldsymbol{E}} d_{j,\boldsymbol{k}}^{n-1,\boldsymbol{e}} \Psi_{j,\boldsymbol{k}}^{\boldsymbol{e}}(\boldsymbol{x}). \end{aligned} \tag{4.21}$$

Step 3. *Prediction step.* The wavelet expansion (4.21) provides a convenient way to measure the smoothness of functions in terms of various function space norms as well as locally [9, 10]. Hence, we use it to locate the smooth regions and rough regions of $\widehat{U}(\boldsymbol{x}, t_{n-1})$ by determining the significant coefficient index sets $\widehat{\boldsymbol{\omega}}_j^{n-1}$. In other words, the wavelet expansion itself is a convenient and accurate error indicator for $U(\boldsymbol{x}, t_{n-1})$. We predict where the rough regions of $U(\boldsymbol{x}, t_n)$ will be at time step t_n by determining the predicted significant coefficient index sets $\widetilde{\boldsymbol{\omega}}_j^n$ at level j at time step t_n. In order to locate the latter, we track the index sets $\widehat{\boldsymbol{\omega}}_j^{n-1}$ forward along the characteristics from time t_{n-1} to time t_n. We also take into account the effect of the source term $q(\boldsymbol{x}, t)$ and the reaction term $R(\boldsymbol{x}, t)$ along the characteristics.

Step 4. *Solution step.* Once we determine the predicted significant coefficient index sets $\widetilde{\boldsymbol{\omega}}_j^n$, we define an adaptive refinement subspace $\widetilde{\mathcal{S}}_J^n(\Omega) \subset \mathcal{S}_J(\Omega)$ by

$$\widetilde{\mathcal{S}}_J^n(\Omega) := \operatorname{span}\left\{\left\{\Phi_{0,\boldsymbol{k}}\right\}_{\boldsymbol{k}\in\boldsymbol{\omega}_0}, \left\{\Psi_{j,\boldsymbol{k}}^{\boldsymbol{e}}\right\}_{\boldsymbol{k}\in\widetilde{\boldsymbol{\omega}}_j^n,\ 0\leq j\leq J-1}^{\boldsymbol{e}\in\boldsymbol{E}}\right\}. \tag{4.22}$$

Then we look for $U(\boldsymbol{x}, t_n) \in \widetilde{\mathcal{S}}_J^n(\Omega)$, which is in the form

$$U(\boldsymbol{x}, t_n) = \sum_{\boldsymbol{k}\in\boldsymbol{\omega}_0} c_{0,\boldsymbol{k}}^{n}\Phi_{0,\boldsymbol{k}}(\boldsymbol{x}) + \sum_{j=0}^{J-1} \sum_{\boldsymbol{k}\in\widetilde{\boldsymbol{\omega}}_j^{n}} \sum_{\boldsymbol{e}\in\boldsymbol{E}} d_{j,\boldsymbol{k}}^{n,\boldsymbol{e}} \Psi_{j,\boldsymbol{k}}^{\boldsymbol{e}}(\boldsymbol{x}), \tag{4.23}$$

such that the following equation holds for any $w(\boldsymbol{x}, t_n) \in \widetilde{\mathcal{S}}_J^n(\Omega)$:

$$\begin{aligned} &\int_\Omega U(\boldsymbol{x}, t_n) w(\boldsymbol{x}, t_n) d\boldsymbol{x} \\ &\quad = \int_\Omega \widehat{U}(\boldsymbol{x}, t_{n-1}) w(\boldsymbol{x}, t_{n-1}^+) d\boldsymbol{x} + \int_\Omega \Lambda(\boldsymbol{x}, t_n) q(\boldsymbol{x}, t_n) w(\boldsymbol{x}, t_n) d\boldsymbol{x}. \end{aligned} \tag{4.24}$$

Here $w(\boldsymbol{x}, t_{n-1}^{+})$ and $\Lambda(\boldsymbol{x}, t_n)$ are defined in (2.7) and (2.9) and below (2.2).

We now briefly discuss the schemes we have developed:

1. Scheme I is a linear and single-level scheme, in which the number of levels and the set of wavelet basis functions used at each time step are independent of the solution being approximated. In Scheme II the number of levels could vary at each time step, depending on the solution being approximated. The first term on the right-hand side of (4.13) provides a basic approximation at the coarsest occurring level 0. The second term provides a finer and finer resolution at time t_n as the level j increases from 0 to $J_n - 1$. Both Schemes I and II are fairly easy to implement.
2. Scheme III is a nonlinear scheme, in which the number of levels and the set of wavelet basis functions $\Psi_{j,\boldsymbol{k}}^{\boldsymbol{e}}(\boldsymbol{x})$ chosen in the approximation depend on the solution being approximated [10]. Notice that the significant wavelet coefficients $\widehat{d}_{j,\boldsymbol{k}}^{n-1,\boldsymbol{e}}$, which exceed the thresholding parameters in (4.19), are nonzero only near the moving steep front regions. Hence, with the first term on the right-hand side of (4.23) as a base approximation, the second term on the right-hand side of (4.23) consists of terms with significant coefficients and provides a progressively improved resolution. In this way, Scheme III resolves the moving steep fronts present in the solutions accurately, adaptively, and effectively.
3. The distribution of the significant coefficients $\widehat{d}_{j,\boldsymbol{k}}^{n-1,\boldsymbol{e}}$ or equivalently the significant coefficient index sets $\widehat{\boldsymbol{\omega}}_j^{n-1}$ in (4.20) could be somewhat irregular or unstructured after the thresholding process (4.19), even though they should have some correlations. A naive organization and management of these coefficients could compromise the greatly improved efficiency of the scheme. The tree approximation techniques proposed in [2], in which a node is in the tree whenever one of its child nodes is in the tree, allow a more effective organization/encoding of the positions of the significant coefficients in an optimal order (i.e., the number of nodes in the tree is a constant multiple of the number of significant coefficients $\widehat{d}_{j,\boldsymbol{k}}^{n-1,\boldsymbol{e}}$). By tracking the significant coefficient index sets $\widehat{\boldsymbol{\omega}}_j^{n-1}$ from time step t_{n-1} to t_n along the characteristics, we can obtain predicted significant coefficient index sets $\widetilde{\boldsymbol{\omega}}_j^{n}$ at time step t_n fairly accurately and efficiently.
4. Because all the wavelet basis functions $\Psi_{j,\boldsymbol{k}}^{\boldsymbol{e}}(\boldsymbol{x})$ have at least zeroth-order vanishing moments, the compression used in Scheme III does not introduce any mass balance error.

5. Stability analysis.

In this section we prove the unconditional stability of the numerical schemes.

THEOREM 5.1. *Scheme* I *is unconditionally stable.*

Using (4.6), we choose $w(\boldsymbol{x}, t_n) = \Phi_{J,\boldsymbol{k}}(\boldsymbol{x})$ in (4.7). Then we multiply the resulting equation by $c_{J,\boldsymbol{k}}^{n}$ in (4.6) and add all the resulting equations $\forall \boldsymbol{k} \in \boldsymbol{\omega}_J$ to obtain

$$\begin{aligned}\int_{\Omega} U^2(\boldsymbol{x}, t_n) d\boldsymbol{x} &= \int_{\Omega} U(\boldsymbol{x}, t_{n-1}) U(\boldsymbol{x}, t_{n-1}^{+}) d\boldsymbol{x} \\ &\quad + \int_{\Omega} \Lambda(\boldsymbol{x}, t_n) q(\boldsymbol{x}, t_n) U(\boldsymbol{x}, t_n) d\boldsymbol{x}.\end{aligned} \tag{5.1}$$

We use the facts

$$
\begin{aligned}
\frac{\partial \boldsymbol{r}(\theta;\boldsymbol{x},t_n)}{\partial \boldsymbol{x}} &= \mathbf{I} + \mathcal{O}(t_n-\theta),\\
U(\boldsymbol{x}^*,t_{n-1}^+) &= U(\boldsymbol{x},t_n)e^{-R(\boldsymbol{x},t_n)\Delta t},
\end{aligned}
\tag{5.2}
$$

with $\mathbf{I}$ being the $d\times d$ identity matrix and $\boldsymbol{x}^* := \boldsymbol{r}(t_{n-1};\boldsymbol{x},t_n)$, to bound the first term on the right-hand side of (5.1). For convenience, we replace the dummy $\boldsymbol{x}$ in this term by $\boldsymbol{x}^*$ and reserve $\boldsymbol{x}$ for the corresponding variable at time t_n:

$$
\begin{aligned}
&\left|\int_\Omega U(\boldsymbol{x}^*,t_{n-1})U(\boldsymbol{x}^*,t_{n-1}^+)d\boldsymbol{x}^*\right|\\
&\quad = \left|\int_\Omega U(\boldsymbol{x}^*,t_{n-1})U(\boldsymbol{x},t_n)e^{-R(\boldsymbol{x},t_n)\Delta t}\left|\frac{\partial \boldsymbol{r}(t_{n-1};\boldsymbol{x},t_n)}{\partial \boldsymbol{x}}\right| d\boldsymbol{x}\right|\\
&\quad \le (1+L\Delta t)\ \|U(\boldsymbol{x},t_n)\|_{L^2(\Omega)}\left[\int_\Omega U^2(\boldsymbol{x}^*,t_{n-1})\left|\frac{\partial \boldsymbol{r}(t_{n-1};\boldsymbol{x},t_n)}{\partial \boldsymbol{x}}\right|^2 d\boldsymbol{x}\right]^{1/2}\\
&\quad \le \left(\frac{1}{2}+L\Delta t\right)\left[\|U(\boldsymbol{x},t_n)\|^2_{L^2(\Omega)} + \|U(\boldsymbol{x},t_{n-1})\|^2_{L^2(\Omega)}\right].
\end{aligned}
\tag{5.3}
$$

Here L represents a generic positive constant, which might assume different values at different places.

Recall that $|\Lambda(\boldsymbol{x},t_n)| \le L\Delta t$; we bound the second term on the right-hand side of (5.1) by

$$
\begin{aligned}
&\left|\int_\Omega \Lambda(\boldsymbol{x},t_n)q(\boldsymbol{x},t_n)U(\boldsymbol{x},t_n)dx\right|\\
&\qquad \le L\Delta t\Big[\|U(\boldsymbol{x},t_n)\|^2_{L^2(\Omega)} + \ \|q(\boldsymbol{x},t_n)\|^2_{L^2(\Omega)}\Big].
\end{aligned}
\tag{5.4}
$$

Substituting (5.3) and (5.4) into (5.1) we obtain

$$
\begin{aligned}
\|U(\boldsymbol{x},t_n)\|^2_{L^2(\Omega)} \le{}& \left(\frac{1}{2}+L_0\Delta t\right)\left[\|U(\boldsymbol{x},t_n)\|^2_{L^2(\Omega)} + \|U(\boldsymbol{x},t_{n-1})\|^2_{L^2(\Omega)}\right]\\
&+ L_0\Delta t\|q(\boldsymbol{x},t_n)\|^2_{L^2(\Omega)},
\end{aligned}
\tag{5.5}
$$

where L_0 is a fixed positive constant.

Adding (5.5) for $n=1,2,\ldots,N_1\le N$ results in the following telescoping series:

$$
\begin{aligned}
&\sum_{n=1}^{N_1}\|U(\boldsymbol{x},t_n)\|^2_{L^2(\Omega)}\\
&\quad \le \left(\frac{1}{2}+L_0\Delta t\right)\sum_{n=1}^{N_1}\Big[\|U(\boldsymbol{x},t_n)\|^2_{L^2(\Omega)} + \|U(\boldsymbol{x},t_{n-1})\|^2_{L^2(\Omega)}\Big]\\
&\qquad + L_0\|q\|^2_{\widehat{L}(0,T;L^2(\Omega))}\\
&\quad \le \left(\frac{1}{2}+L_0\Delta t\right)\left[\|U(\boldsymbol{x},t_{N_1})\|^2_{L^2(\Omega)} + \|U(\boldsymbol{x},0)\|^2_{L^2(\Omega)}\right]\\
&\qquad + (1+2L_0\Delta t)\sum_{n=1}^{N_1-1}\ \|U(\boldsymbol{x},t_n)\|^2_{L^2(\Omega)} + L_0\|q\|^2_{\widehat{L}(0,T;L^2(\Omega))}.
\end{aligned}
\tag{5.6}
$$

Canceling the corresponding terms on both sides of (5.6) yields the following inequality:

$$\begin{aligned}\|U(\boldsymbol{x},t_{N_1})\|^2_{L^2(\Omega)} &\le (1+2L_0\Delta t)\|U(\boldsymbol{x},0)\|^2_{L^2(\Omega)}\\ &\quad + 4L_0\Delta t\sum_{n=1}^{N_1}\|U(\boldsymbol{x},t_n)\|^2_{L^2(\Omega)} + 2L_0\|q\|^2_{\widehat{L}(0,T;L^2(\Omega))}.\end{aligned} \tag{5.7}$$

Taking Δt small enough such that $4L_0\Delta t \le 1/2$, we rewrite (5.7) as follows:

$$\begin{aligned}\|U(\boldsymbol{x},t_{N_1})\|^2_{L^2(\Omega)} &\le 8L_0\Delta t\sum_{n=1}^{N_1-1}\|U(\boldsymbol{x},t_n)\|^2_{L^2(\Omega)}\\ &\quad + \frac{5}{2}\|U(\boldsymbol{x},0)\|^2_{L^2(\Omega)} + 4L_0\|q\|^2_{\widehat{L}(0,T;L^2(\Omega))}.\end{aligned} \tag{5.8}$$

Applying Gronwall's inequality to (5.8) yields the following stability estimate:

$$\|U\|_{\widehat{L}^\infty(0,T;L^2(\Omega))} \le L\Big[\|u_0\|_{L^2(\Omega)} + \|q\|_{\widehat{L}^2(0,T;L^2(\Omega))}\Big], \tag{5.9}$$

where $\|U(\boldsymbol{x},0)\|_{L^2(\Omega)}$ is bounded by $\|u_0\|_{L^2(\Omega)}$, and

$$\begin{aligned}\|U\|_{\widehat{L}^\infty(0,T;L^2(\Omega))} &= \max_{0\le n\le N}\|U(\boldsymbol{x},t_n)\|_{L^2(\Omega)},\\ \|U\|_{\widehat{L}^2(0,T;L^2(\Omega))} &= \left[\Delta t\sum_{n=0}^{N}\|U(\boldsymbol{x},t_n)\|^2_{L^2(\Omega)}\right]^{1/2}.\end{aligned} \tag{5.10}$$

Thus, we have proved the unconditional stability of Scheme I.

THEOREM 5.2. *Schemes* II *and* III *are explicit and unconditionally stable.*

The explicitness of Scheme III can be shown similarly to that of Scheme II in (4.14), with $\boldsymbol{\omega}_j$ and J_n replaced by $\widetilde{\boldsymbol{\omega}}_j^n$ and J, respectively.

We now prove the unconditional stability of Schemes II and III. Recalling the expression (4.13) for $U(\boldsymbol{x},t_n)$, we multiply (4.7) with $w(\boldsymbol{x},t_n) = \Phi_{0,\boldsymbol{k}}$ by $c_{0,\boldsymbol{k}}(U_n)$ $\forall \boldsymbol{k}\in\boldsymbol{\omega}_0$, and (4.7) with $w(\boldsymbol{x},t_n) = \Psi^{\boldsymbol{e}}_{j,\boldsymbol{k}}$ by $d^{n,\boldsymbol{e}}_{j,\boldsymbol{k}}(U_n)$ $\forall\boldsymbol{k}\in\boldsymbol{\omega}_j$, $\boldsymbol{e}\in\boldsymbol{E}$, and $0\le j\le J_n-1$, and add all the resulting equations, yielding (5.1) again. Even though the number of levels J_{n-1} at time step t_{n-1} and at time step t_n could be different, the techniques used in Theorem 5.1 still work and lead to the stability estimate (5.9).

To prove the unconditional stability for Scheme III, we recall the expression (4.23) for $U(\boldsymbol{x},t_n)$. Multiplying (4.24) with $w(\boldsymbol{x},t_n) = \Phi_{0,\boldsymbol{k}}$ by $c^n_{0,\boldsymbol{k}}(U_n)$ $\forall\boldsymbol{k}\in\boldsymbol{\omega}_0$, and (4.24) with $w(\boldsymbol{x},t_n) = \Psi^{\boldsymbol{e}}_{j,\boldsymbol{k}}$ by $d^{n,\boldsymbol{e}}_{j,\boldsymbol{k}}(U_n)$ $\forall\boldsymbol{k}\in\widetilde{\boldsymbol{\omega}}^n_j$, $\boldsymbol{e}\in\boldsymbol{E}$, and $0\le j\le J-1$, and adding all the resulting equations, we obtain

$$\begin{aligned}\int_\Omega U^2(\boldsymbol{x},t_n)d\boldsymbol{x} &= \int_\Omega \widehat{U}(\boldsymbol{x},t_{n-1})U(\boldsymbol{x},t^+_{n-1})d\boldsymbol{x}\\ &\quad + \int_\Omega \Lambda(\boldsymbol{x},t_n)q(\boldsymbol{x},t_n)U(\boldsymbol{x},t_n)d\boldsymbol{x}.\end{aligned} \tag{5.11}$$

Using the inequality

$$
\begin{aligned}
&\|U(\boldsymbol{x}, t_{n-1})\|^2_{L^2(\Omega)} \\
&= \sum_{\boldsymbol{k}\in\boldsymbol{\omega}_0} \left|c^{n-1}_{0,\boldsymbol{k}}(U_{n-1})\right|^2 + \sum_{j=0}^{J-1} \sum_{\boldsymbol{k}\in\widetilde{\boldsymbol{\omega}}^{n-1}_j} \sum_{\boldsymbol{e}\in\boldsymbol{E}} \left|d^{n-1,\boldsymbol{e}}_{j,\boldsymbol{k}}(U_{n-1})\right|^2 \\
&= \|\widehat{U}(\boldsymbol{x}, t_{n-1})\|^2_{L^2(\Omega)} + \sum_{j=0}^{J-1} \sum_{\boldsymbol{k}\in\widetilde{\boldsymbol{\omega}}^{n-1}_j\backslash\widehat{\boldsymbol{\omega}}^{n-1}_j} \sum_{\boldsymbol{e}\in\boldsymbol{E}} \left|d^{n-1,\boldsymbol{e}}_{j,\boldsymbol{k}}(U_{n-1})\right|^2 \\
&\geq \|\widehat{U}(\boldsymbol{x}, t_{n-1})\|^2_{L^2(\Omega)},
\end{aligned}
\tag{5.12}
$$

we bound the first term on the right-hand side of (5.11) as in (5.3):

$$
\begin{aligned}
&\left|\int_\Omega \widehat{U}(\boldsymbol{x}, t_{n-1}) U(\boldsymbol{x}, t^+_{n-1}) d\boldsymbol{x}\right| \\
&\leq \left(\frac{1}{2} + L\Delta t\right) \left[\|U(\boldsymbol{x}, t_n)\|^2_{L^2(\Omega)} + \|\widehat{U}(\boldsymbol{x}, t_{n-1})\|^2_{L^2(\Omega)}\right] \\
&\leq \left(\frac{1}{2} + L\Delta t\right) \left[\|U(\boldsymbol{x}, t_n)\|^2_{L^2(\Omega)} + \|U(\boldsymbol{x}, t_{n-1})\|^2_{L^2(\Omega)}\right].
\end{aligned}
\tag{5.13}
$$

We then obtain the estimate (5.5) again. The same technique as that used in proving Theorem 5.1 concludes the proof of this theorem.

6. Numerical experiments. We consider the transport of a two-dimensional Gaussian pulse, with or without a reactive process involved. To gain some basic understanding of the numerical methods, we compare these schemes with the standard upwind scheme that has been well understood and widely used in industrial applications.

In the example runs, the spatial domain is $\Omega := (-1, 1) \times (-1, 1)$, a rotating velocity field is imposed as $V_1(x_1, x_2) = -4x_2$, and $V_2(x_1, x_2) = 4x_1$. The time interval is $[0, T] = [0, \pi/2]$, which is the time period required for one complete rotation. The initial condition $u_0(x_1, x_2)$ is given by

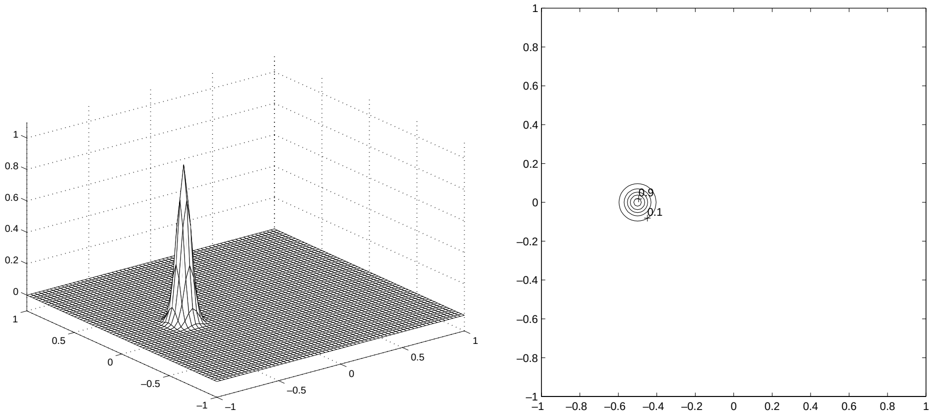
$$
u_0(x_1, x_2) := \exp\left(-\frac{(x_1 - x_{1c})^2 + (x_2 - x_{2c})^2}{2\sigma^2}\right),
\tag{6.1}
$$

where x_{1c}, x_{2c}, and σ are the centered and standard deviations, respectively. The corresponding analytical solution for (2.1) with $q = 0$ is given by

$$
\begin{aligned}
&u(x_1, x_2, t) \\
&= \exp\left(-\frac{(\bar{x}_1 - x_{1c})^2 + (\bar{x}_2 - x_{2c})^2}{2\sigma^2} - \int_0^t R(\boldsymbol{r}(\theta; \bar{x}_1, \bar{x}_2, 0), \theta) d\theta\right),
\end{aligned}
\tag{6.2}
$$

where $\bar{x}_1 := x_1 \cos(4t) + x_2 \sin(4t)$, $\bar{x}_2 := -x_1 \sin(4t) + x_2 \cos(4t)$, and $\boldsymbol{r}(\theta; \bar{x}_1, \bar{x}_2, 0) := (\bar{x}_1 \cos(4\theta) - \bar{x}_2 \sin(4\theta), \bar{x}_1 \sin(4\theta) + \bar{x}_2 \cos(4\theta))$.

This example has been used widely to test different schemes for numerical artifacts such as numerical stability, numerical dispersion, spurious oscillations, deformation, and phase errors as well as other numerical effects arising in porous medium fluid flows. In the numerical experiments, the data are chosen as follows: $q = 0$, $x_{1c} = -0.5$, $x_{2c} = 0$, and $\sigma = 0.0447$. To observe the capability of these schemes in handling reactive



(a) Surface plot

(b) Contour plot with intervals of 0.2

FIG. 1. *Surface and contour plots of the analytical solution at* $T = \frac{\pi}{2}$.

transport processes, we consider both $R = 0$ and $R = \cos(2t)$. When no reaction is present, the analytical solution $u(x_1, x_2, t)$ after one complete rotation is identical to the initial condition $u_0(x_1, x_2)$, which is centered at (x_{1c}, x_{2c}) with a minimum value 0 and a maximum value 1. The surface and contour plots of the analytical solution (which is identical to the initial condition) are presented in Figure 1(a) and (b). For $R = \cos(2t)$, the analytical solution given by (6.2) now becomes

$$u(x_1, x_2, t) = \exp\left(-\frac{1}{2}\sin(2t) - \frac{(\bar{x}_1 - x_{1c})^2 + (\bar{x}_2 - x_{2c})^2}{2\sigma^2}\right), \tag{6.3}$$

which is identical to the analytical solution with no reaction at the final time $t = \frac{\pi}{2}$. Hence, we can have a fair comparison of errors for both $R = 0$ and $R = \cos(2t)$.

We use the fourth-order Daubechies wavelets with a coarsest occurring level of grid size $h_0 = \frac{1}{8}$, a finest occurring level of $J = 3$, and a very coarse time step of $\Delta t = \pi/8$. This leads to a maximal Courant number of 115. We apply Scheme I (which is identical to Scheme II) at the finest level $J = 3$ to compute the uncompressed solution. We then apply Scheme III with the coarsest level of mesh size $h_0 = \frac{1}{8}$ and the finest level $J = 3$ to compute the compressed solutions. In all the schemes, a fourth-order Runge–Kutta method with a micro time step of $\Delta t_m = \Delta t/4$ is used to track the characteristics. The tolerance (4.18) now becomes

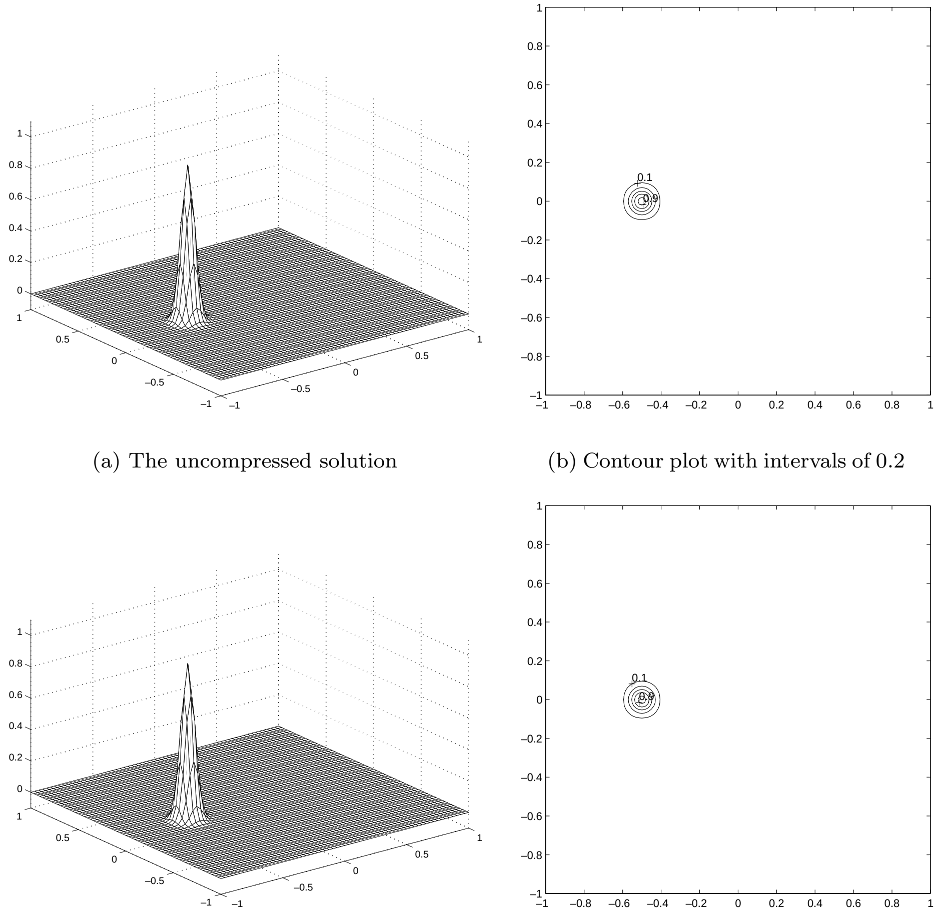
$$\varepsilon_j := 2^{-j}\ \Delta t\ \varepsilon. \tag{6.4}$$

In Table 1, we present the L^1-, L^2-, and L^∞-errors, the maximum and minimum values, and compression ratios (the number of unknowns in the uncompressed solution versus that of the unknowns in the compressed solution) of the uncompressed ($\varepsilon = 0$) and compressed solutions at the final step $T = \pi/2$ for different choices of tolerance ε and for both $R = 0$ and $R = \cos(2t)$. The contour and surface plots for the uncompressed solution and the compressed solution with $\varepsilon = 0.0001$ at the final step $T = \pi/2$ and for $R = \cos(2t)$ are plotted in Figure 2(a)–(d). These results show that the schemes developed in this paper generate very accurate solutions, even if

TABLE 1

Statistics of uncompressed ($\varepsilon = 0$) and compressed solutions for different choices of tolerance ε. The time step is $\Delta t = \frac{\pi}{8}$ with a micro time step of $\Delta t_m = \frac{\Delta t}{4}$ used in tracking. The coarsest mesh size is $h_0 = \frac{1}{8}$ and the finest mesh size is $h_J = \frac{1}{64}$.

ε	Compression rate	L_1-error	L_2-error	L_∞-error	Max	Min	CPU
$R(\mathsf{x}, t) = 0$							
0	N/A	2.92×10^{-4}	1.19×10^{-3}	1.38×10^{-2}	0.992	0	1 m 6 s
10^{-5}	26	2.92×10^{-4}	1.19×10^{-3}	1.38×10^{-2}	0.992	0	47 s
10^{-4}	42	3.04×10^{-4}	1.19×10^{-3}	1.39×10^{-2}	0.992	0	33 s
10^{-3}	75	5.91×10^{-4}	1.85×10^{-3}	2.41×10^{-2}	0.985	0	22 s
$R(\mathsf{x}, t) = \cos(2t)$							
0	N/A	3.10×10^{-4}	1.35×10^{-3}	1.74×10^{-2}	0.991	0	1 m 23 s
10^{-5}	27	3.11×10^{-4}	1.35×10^{-3}	1.74×10^{-2}	0.991	0	1 m
10^{-4}	43	3.21×10^{-4}	1.35×10^{-3}	1.74×10^{-2}	0.991	0	42 s
10^{-3}	75	4.74×10^{-4}	1.66×10^{-3}	2.13×10^{-2}	0.987	0	31 s



(a) The uncompressed solution

(b) Contour plot with intervals of 0.2

(c) The compressed solution ($\varepsilon = 0.0001$)

(d) Contour plot with intervals of 0.2

FIG. 2. *The uncompressed and compressed solutions at $T = \frac{\pi}{2}$.*

very large time steps and fairly coarse grids are used. The schemes are explicit and highly parallelizable. We observe that with fairly large compression ratios, Scheme III generates a solution comparable to the uncompressed solution. This implies a further improvement in terms of computational efficiency and storage. In the numerical implementation of these schemes, we focus on the study of the trade-off between the compressibility and accuracy. We understand that a fine-tuning and optimization of the implementations will fully explore the adaptivity of Scheme III and will further improve its CPU performance. Finally, these schemes handle the reactive effect accurately and generate numerical solutions with about the same errors as the solutions with no reaction involved.

It is well known that the standard upwind scheme is explicit and fairly easy to implement, and can generate very stable solutions with correct qualitative physical trend even for very complex problems. However, the upwind scheme introduces excessive numerical diffusion and tends to severely smear the steep fronts of the numerical solutions. We present in Table 2 the numerical solutions of the upwind scheme with various time steps and spatial grids. With the base spatial grid size of $h = h_J = \frac{1}{64}$ that was used in Table 1, the time step $\Delta t = \frac{\pi}{1200}$ is the largest admissible step size that meets the CFL condition (the Courant number is 0.95). The upwind scheme generates an extremely diffusive solution with a maximal value of only 0.080 that is only 8% of the height of the true solution, even though it is extremely efficient per time step (it took only 4 seconds for 1200 time steps). The surface and contour plots of the solution are presented in Figure 3(a) and (b). We further notice that with the same spatial grid size, the errors increase slightly as the size of the time step is further reduced. This observation can be explained by using the local truncation error analysis, which shows that the local spatial and temporal errors actually have opposite signs and cancel with each other. Hence, reducing further the time step size only leads to increased local truncation error, and so to increased truncation errors of the numerical solutions of the upwind scheme. To improve the accuracy of the numerical solutions, we have to refine both the spatial grids and the time steps with the Courant number being close to unity. With a CPU time comparable to that which Schemes I–III consumed, the upwind scheme generates a solution using a spatial grid size of $h = \frac{1}{128}$ and a time step of $\Delta t = \frac{\pi}{4800}$. However, the resulting solution has a maximal value of only 0.131. The finest grids used are $\Delta t = \frac{\pi}{20000}$ and $h = \frac{1}{1024}$. It took a CPU time of almost 6 hours for the upwind scheme to generate a solution with a maximal value of 0.579. The surface and contour plots of the numerical solution are presented in Figure 3(c) and (d). We also observe slight deformation due to the grid orientation effect. These comparisons show that these schemes are very competitive and hold strong potential.

7. Extension.

We outline the extension of the schemes developed in this paper to the initial-boundary-value problem of advection-reaction PDEs:

$$\begin{aligned} u_t + \nabla\cdot(\mathbf{v}u) + Ru &= q(\boldsymbol{x},t), && (\boldsymbol{x},t)\in\Omega\times(0,T],\\ u(\boldsymbol{x},t) &= u_\Gamma(\boldsymbol{x},t), && \boldsymbol{x}\in\Gamma^{(I)},\quad t\in[0,T],\\ u(\boldsymbol{x},0) &= u_0(\boldsymbol{x}), && \boldsymbol{x}\in\Omega, \end{aligned} \tag{7.1}$$

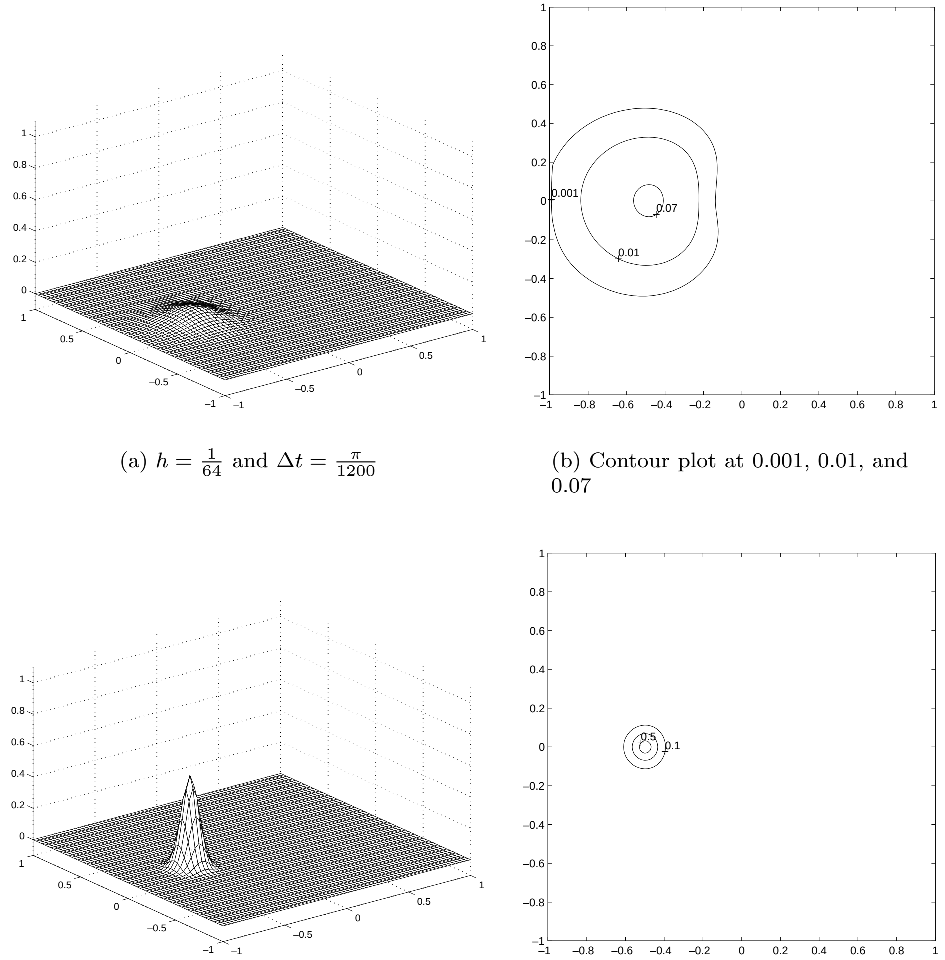
where $\Omega := (a_1,b_1)\times\cdots\times(a_d,b_d)$ is the spatial domain with the boundary $\Gamma = \partial\Omega$. $\Gamma^{(I)}$ and $\Gamma^{(O)}$ are the inflow and outflow boundaries identified by

$$\begin{aligned} \Gamma^{(I)} &:= \{\boldsymbol{x} \mid \boldsymbol{x}\in\partial\Omega,\ \mathbf{v}\cdot\boldsymbol{n}<0\},\\ \Gamma^{(O)} &:= \{\boldsymbol{x} \mid \boldsymbol{x}\in\partial\Omega,\ \mathbf{v}\cdot\boldsymbol{n}>0\}. \end{aligned} \tag{7.2}$$

Table 2

Statistics of upwind schemes at time $T = \frac{\pi}{2}$ with $R = 0$ and different spatial grids and time steps.

h	Δt	Courant #	L_1-error	L_2-error	L_∞-error	Max	Min	CPU
$\frac{1}{64}$	$\frac{\pi}{1200}$	0.95	1.85×10^{-2}	7.02×10^{-2}	9.21×10^{-1}	0.080	0	4 s
$\frac{1}{64}$	$\frac{\pi}{2400}$	0.47	1.91×10^{-2}	7.13×10^{-2}	9.30×10^{-1}	0.070	0	8 s
$\frac{1}{64}$	$\frac{\pi}{4800}$	0.24	1.93×10^{-2}	7.17×10^{-2}	9.34×10^{-1}	0.067	0	17 s
$\frac{1}{128}$	$\frac{\pi}{2400}$	0.95	1.54×10^{-2}	6.30×10^{-2}	8.52×10^{-1}	0.148	0	38 s
$\frac{1}{128}$	$\frac{\pi}{4800}$	0.47	1.61×10^{-2}	6.48×10^{-2}	8.69×10^{-1}	0.131	0	1 m 16 s
$\frac{1}{256}$	$\frac{\pi}{4800}$	0.95	1.17×10^{-2}	5.25×10^{-2}	7.42×10^{-1}	0.258	0	4 m 22 s
$\frac{1}{512}$	$\frac{\pi}{9600}$	0.95	7.99×10^{-3}	3.94×10^{-2}	5.90×10^{-1}	0.411	0	36 m 22 s
$\frac{1}{1024}$	$\frac{\pi}{20000}$	0.91	5.01×10^{-3}	2.67×10^{-2}	4.22×10^{-1}	0.579	0	5 h 42 m



(a) $h = \frac{1}{64}$ and $\Delta t = \frac{\pi}{1200}$

(b) Contour plot at 0.001, 0.01, and 0.07

(c) $h = \frac{1}{1024}$ and $\Delta t = \frac{\pi}{20000}$

(d) Contour plot with intervals of 0.2

Fig. 3. *The upwind solutions with different grid sizes and time steps at $T = \frac{\pi}{2}$.*

The weak formulation corresponding to (2.2) now reads as

$$\begin{aligned}
&\int_\Omega u(\boldsymbol{x},t_n)w(\boldsymbol{x},t_n)d\boldsymbol{x} + \int_{t_{n-1}}^{t_n}\int_\Gamma \mathbf{v}(\boldsymbol{x},t)\cdot\boldsymbol{n}(\boldsymbol{x})\ u(\boldsymbol{x},t)w(\boldsymbol{x},t)dsdt \\
&\quad - \int_{t_{n-1}}^{t_n}\int_\Omega u(w_t + \mathbf{v}\cdot\nabla w - Rw)(\boldsymbol{x},t)d\boldsymbol{x}dt \\
&\quad = \int_\Omega u(\boldsymbol{x},t_{n-1})w(\boldsymbol{x},t_{n-1}^+)d\boldsymbol{x} + \int_{t_{n-1}}^{t_n}\int_\Omega q(\boldsymbol{x},t)w(\boldsymbol{x},t)d\boldsymbol{x}dt.
\end{aligned} \tag{7.3}$$

Now the characteristic $\boldsymbol{r}(\theta;\check{\boldsymbol{x}},\check{t})$ is still determined by (2.4)–(2.6), but with the exception that either $(\check{\boldsymbol{x}},\check{t}) = (\boldsymbol{x},t_n)$ for $\boldsymbol{x}\in\overline{\Omega}$ or $(\check{\boldsymbol{x}},\check{t}) = (\boldsymbol{x},t)$ for $\boldsymbol{x}\in\Gamma^{(O)}$ and $t\in[t_{n-1},t_n]$. Then derivation similar to that for (2.11) leads to the reference equation

$$\begin{aligned}
&\int_\Omega u(\boldsymbol{x},t_n)w(\boldsymbol{x},t_n)d\boldsymbol{x} + \int_{t_{n-1}}^{t_n}\int_{\Gamma^{(O)}} \mathbf{v}(\boldsymbol{x},t)\cdot\boldsymbol{n}(\boldsymbol{x})\ u(\boldsymbol{x},t)w(\boldsymbol{x},t)dsdt \\
&\quad = \int_\Omega u(\boldsymbol{x},t_{n-1})w(\boldsymbol{x},t_{n-1}^+)d\boldsymbol{x} + \int_\Omega \Lambda(\boldsymbol{x},t_n)q(\boldsymbol{x},t_n)w(\boldsymbol{x},t_n)d\boldsymbol{x} \\
&\quad + \int_{t_{n-1}}^{t_n}\int_{\Gamma^{(O)}} \Lambda^{(1)}(\boldsymbol{x},t)\mathbf{v}(\boldsymbol{x},t)\cdot\boldsymbol{n}(\boldsymbol{x})q(\boldsymbol{x},t)w(\boldsymbol{x},t)d\boldsymbol{x} \\
&\quad - \int_{t_{n-1}}^{t_n}\int_{\Gamma^{(O)}} \mathbf{v}(\boldsymbol{x},t)\cdot\boldsymbol{n}(\boldsymbol{x})u_\Gamma(\boldsymbol{x},t)w(\boldsymbol{x},t)d\boldsymbol{x} + E(w),
\end{aligned} \tag{7.4}$$

with

$$\Lambda^{(1)}(\boldsymbol{x},t) := \begin{cases} \dfrac{1-e^{-R(\boldsymbol{x},t)(t-t^*(\boldsymbol{x},t))}}{R(\boldsymbol{x},t)} & \text{if } R(\boldsymbol{x},t)\neq 0, \\ t - t^*(\boldsymbol{x},t) & \text{otherwise,} \end{cases} \tag{7.5}$$

where $t^*(\boldsymbol{x},t) = t_{n-1}$ if $\boldsymbol{r}(\theta;\boldsymbol{x},t)$ does not backtrack to the boundary Γ, or $t^*(\boldsymbol{x},t)$ represents the time instant when $\boldsymbol{r}(\theta;\boldsymbol{x},t)$ backtracks to the boundary Γ otherwise.

Based on the reference equation (7.5), we can define Schemes I–III as before. But now the unknown trial functions $U(\boldsymbol{x},t)$ are defined in $\overline{\Omega}$ at time t_n and on the space-time outflow boundary $\Gamma^{(O)}\times[t_{n-1},t_n]$. The scaling and wavelet basis functions used in section 4 might not be orthogonal anymore near the boundary of the domain $\overline{\Omega}$. Consequently, the schemes might not be explicit anymore. Notice that the region where the basis functions are not orthogonal is of order Δt. Hence, the derived schemes are explicit in most of the domain and are implicit near boundary. In other words, we reduce the space dimension of the implicit scheme by one. Alternatively, we could utilize the results in [4, 17] to modify the wavelet basis functions near boundary to make them orthonormal and again lead to fully explicit schemes.

8. Conclusions. The well-known CFL condition states that there are no explicit, unconditionally stable, consistent finite difference schemes (in fact, any schemes with fixed stencils) for linear hyperbolic PDEs [5]. Therefore, although explicit methods are relatively easy to implement, are local, and are fully parallelizable, the time step sizes in these methods are subject to the CFL condition. In fact, explicit methods often have to use very small time steps in numerical simulations to maintain the stability of the methods [11]. In contrast, implicit methods are unconditionally stable.

However, they require inverting a coefficient matrix at each time step and could be expensive. Moreover, the time step sizes of implicit methods are still restricted for the reason of accuracy, especially when steep fronts pass by.

In this paper we combine an Eulerian–Lagrangian approach and multiresolution analysis to develop three unconditionally stable, explicit schemes for multidimensional linear hyperbolic PDEs. Scheme I is a single-level scheme that uses all the scaling functions at a fine level J as basis functions. It is in the flavor of standard finite element methods and is fairly straightforward to implement. Scheme II is a multilevel scheme that uses all the scaling functions at a coarsest occurring discretization level 0 and all the wavelets on all the levels $0, 1, \ldots, J_n - 1$ as basis functions. It is similar to multigrid methods with a slash cycle, and does not need to go back and forth between the coarse grids and the fine grids (see [13] and the references therein). Scheme III uses a thresholding and compression technique to adaptively select the wavelet basis functions at each successive level $0, 1, \ldots, J - 1$. It significantly reduces the number of equations and coefficients that need to be solved, while still showing accuracy comparable to the uncompressed schemes (Schemes I and II). Hence, it has a greatly improved efficiency and storage. The scheme is nonlinear and is related to adaptive finite element methods. Furthermore, the compression used in the scheme does not introduce any mass balance error. As we have seen, by using a multiresolution analysis and orthogonal wavelets with an Eulerian–Lagrangian approach, we are able to obtain single-level and multilevel, explicit schemes. The use of Lagrangian coordinates enables us to obtain accurate solutions even if very large time steps are used. Moreover, the use of Lagrangian coordinates defines the stencils adaptively following the flow of streamlines, and the stencils are not necessarily fixed. This is the fundamental reason why we could develop CFL-free, unconditionally stable, convergent numerical methods for hyperbolic PDEs without contradicting the well-known CFL condition. Our previous computational results have shown the strong potential of the methods developed. The convergence analysis and error estimate for Scheme I can be derived in a more or less standard way, but the error estimates for Schemes II and III require more work. We will present the theoretical error estimates for these schemes elsewhere.

Acknowledgment. The authors would like to thank the referees for their very helpful comments and suggestions, which greatly improved the quality of this paper.

REFERENCES

[1] M. A. Celia, T. F. Russell, I. Herrera, and R. E. Ewing, *An Eulerian-Lagrangian localized adjoint method for the advection-diffusion equation*, Adv. Water Resources, 13 (1990), pp. 187–206.

[2] A. Cohen, W. Dahmen, I. Daubechies, and R. DeVore, *Tree Approximation and Optimal Encoding*, IMI Preprint Series, Department of Mathematics, University of South Carolina, Columbia, SC, 1999, p. 9.

[3] A. Cohen, W. Dahmen, and R. DeVore, *Adaptive wavelet methods for elliptic operator equations: Convergence rates*, Math. Comp., 70 (2001), pp. 27–75.

[4] A. Cohen, I. Daubechies, B. Jawerth, and P. Vial, *Multiresolution analysis, wavelets and fast algorithms on an interval*, C. R. Acad. Sci. Paris Sér. I Math., 316 (1993), pp. 417–421.

[5] R. Courant, K. O. Friedrichs, and H. Lewy, *Über die partiellen differenzen-gleichungen der mathematisches physik*, Math. Ann., 100 (1928), pp. 32–74.

[6] W. Dahmen, A. Kunoth, and K. Urban, *A wavelet Galerkin method for the Stokes equations*, Computing, 56 (1996), pp. 259–301.

[7] W. Dahmen, R. Schneider, and Y. Xu, *Nonlinear functionals of wavelet expansions—adaptive reconstruction and fast evaluation*, Numer. Math., 86 (2000), pp. 49–101.

[8] I. Daubechies, *Orthogonal bases of compactly supported wavelets*, Comm. Pure Appl. Math., 41 (1988), pp. 909–996.

[9] I. Daubechies, *Ten Lectures on Wavelets*, CBMS–NSF Regional Conf. Ser. in Appl. Math. 61, SIAM, Philadelphia, 1992.

[10] R. A. DeVore, *Nonlinear approximation*, Acta Numer., 7 (1998), pp. 51–150.

[11] B. A. Finlayson, *Numerical Methods for Problems with Moving Fronts*, Ravenna Park Publishing, Seattle, WA, 1992.

[12] A. Haar, *Zur theorie der orthogonalen funktionen-systeme*, Math. Ann., 69 (1910), pp. 331–371.

[13] W. Hackbush, *Multi-Grid Methods and Applications*, Springer-Verlag, Berlin, 1985.

[14] A. Harten, *Multiresolution algorithms for the numerical solution of hyperbolic conservation laws*, Comm. Pure Appl. Math., 48 (1995), pp. 1305–1342.

[15] R. J. LeVeque, *Numerical Methods for Conservation Laws*, Birkhäuser, Basel, 1992.

[16] S. G. Mallet, *Multiresolution and wavelet orthonormal bases in $L^2(\mathbb{R})$*, Trans. Amer. Math. Soc., 315 (1989), pp. 69–87.

[17] Y. Meyer, *Ondelettes sur l'intervalle*, Rev. Mat. Iberoamericana, 7 (1991), pp. 115–133.

[18] J. Morlet, G. Arens, I. Fourgeau, and D. Giard, *Wave propagation and sampling theory*, Geophysics, 47 (1982), pp. 203–236.

[19] T. F. Russell and R. V. Trujillo, *Eulerian-Lagrangian localized adjoint methods with variable coefficients in multiple dimensions*, in Computational Methods in Surface Hydrology, Springer-Verlag, Berlin, 1990, pp. 357–363.

[20] M. J. Smith and D. P. Barnwell, *Exact reconstruction for tree-structured subband coders*, IEEE Trans. Acoust. Speech Signal Process., 34 (1986), pp. 434–441.

[21] J. O. Stromberg, *A modified Franklin system and higher order spline on $\mathbb{R}^n$ as unconditional bases for Hardy spaces*, in Conferences on Harmonic Analysis, in honor of Antoni Zygmund (Vol. I and II), Wadsworth Math. Series, Belmont, CA, 1983, pp. 475–493.

[22] M. Vetterli, *Filter banks allowing perfect reconstruction*, Signal Process., 10 (1986), pp. 219–244.

SIAM J. Sci. Comput.
Vol. 23, No. 5, pp. 1439–1455

IMAGE PROCESSING FOR NUMERICAL APPROXIMATIONS OF CONSERVATION LAWS: NONLINEAR ANISOTROPIC ARTIFICIAL DISSIPATION[*]

THORSTEN GRAHS[†], ANDREAS MEISTER[‡], AND THOMAS SONAR[†]

Abstract. We employ a nonlinear anisotropic diffusion operator like the ones used as a means of filtering and edge enhancement in image processing and in numerical methods for conservation laws. It turns out that algorithms currently used in image processing are very well suited for the design of nonlinear higher order dissipative terms. In particular, we stabilize the well-known Lax–Wendroff formula by means of a nonlinear diffusion term.

Key words. artificial diffusion, higher order methods, nonlinear filter, finite differences

AMS subject classifications. 65M12, 76M20, 35L65

PII. S106482759935143X

1. Introduction.

The construction of suitable artificial viscosity terms for stabilizing finite difference schemes of higher order is a difficult task. In the last decade we therefore observed a strong tendency to construct numerical approximations of conservation laws without explicit knowledge of their numerical diffusion. The modern total variation diminishing (TVD) or essentially nonoscillatory (ENO) schemes belong to this class, in which a basic first-order scheme is enhanced by the use of sophisticated recovery functions; see [10], [4]. There are, however, certain circumstances in which an approach using explicit construction of artificial dissipation would be advantageous. If we consider pseudospectral methods, the concept of ENO recovery is very hard to apply if the degree of the basis polynomials used is high. Here one would like to compute shocked solutions with central differences and to postprocess the oscillatory numerical solution such that

(i) high frequency oscillations are filtered, and
(ii) shocks are steepened and represented with high resolution.

Another area of application is grid-free methods (see [1]), where modern concepts of recovery fail due to the irregularity of the interpolation points. In that case one would like to compute all derivatives from a central interpolant (or, better, a least squares approximation) and postprocess the derivatives as was described above.

Over the years there were no general attempts to derive a constructive theory which would enable the design of suitable artificial viscosities within the CFD community. However, filtering and edge enhancement is a fundamental task in image processing, and in recent years a theory of nonlinear anisotropic diffusion was created and can now be found in textbooks like [14] and [5]. In a noisy picture one also would like to filter the high frequency components before detecting the edges (i.e., jumps in grey level). Then one would like to enhance the edges in order to represent the edges in high resolution. Now there is nothing which keeps us from interpreting our numerical solution corresponding to a conservation law as a photograph or picture,

[*] Received by the editors January 4, 1999; accepted for publication (in revised form) July 12, 2001; published electronically January 4, 2002.
http://www.siam.org/journals/sisc/23-5/35143.html

[†] Institut für Analysis, TU Braunschweig, Pockelsstrasse 14, D-38106 Braunschweig, Germany (t.grahs@tu-bs.de, t.sonar@tu-bs.de).

[‡] Fachbereich Mathematik, Universität Hamburg, Bundesstrasse 55, D-20146 Hamburg, Germany (meister@math.uni-hamburg.de).

at least not in the case of steady solutions. In the same way the photographer would very much prefer to see the contours on his picture as sharp thin lines, the numerical analyst would prefer to see shocks as crispy lines instead of smeared thick regions. To accomplish this photo as well as numerical solution have to be denoised. After removing the high frequencies we would like to spend a dose of diffusion tangential on shocks, but we would like to avoid diffusion across shocks (that is what anisotropy is all about). In contrast, in the vicinity of shocks we would like to solve a kind of nonlinear anisotropic backward heat equation to enhance the structure of a shock. Devices and algorithms satisfying exactly these requirements are ready to use if one is willing to enter the area of image processing.

The aim of this paper is mainly to show the potential of the methods developed for image processing if they are applied consistently to problems in the numerical treatment of hyperbolic conservation laws. We hope that we open up a new chapter in the design of modern nonlinear discretizations of conservation laws by using concepts which are well known in the image processing society. In the present paper we demonstrate our algorithms with the Lax–Wendroff formula for two-dimensional scalar equations. A future paper will be devoted to more serious applications in the field of gas dynamics.

The outline of the paper is as follows. After a brief review of the concepts used in the numerical treatment of conservation laws we apply the classical Lax–Wendroff formula to a steady nonlinear problem in which a shock is present. As is well known the second-order Lax–Wendroff formula answers these type of problems with violent oscillations. In image processing Gaussian smoothing would be applied to the numerical solution in order to filter high frequency components. We exemplify this strategy but never use it in our final algorithms. On one hand, the control of the linear diffusion is very difficult. (A little overdose deteriorates the numerical solution strongly.) On the other hand, there are no visible differences in the final solution when computed with or without presmoothing. We then utilize the structure tensor which serves as a detector of the local coherence of our solution. In looking at eigenvalues and eigenvectors, regions of anisotropy can be detected, as well as regions of constant states. We then proceed to the construction of a nonlinear anisotropic artificial viscosity term. It is here where one has the freedom to choose between many different possible models. In our paper we decided to use an anisotropic regularization of the classical Perona–Malik model due to Weickert. There are certainly more clever choices, but all the important ideas are contained in this specific example. This anisotropic artificial dissipation is discretized so that the discrete equation is stable. The underlying algorithm controlling the discretization is also taken from image processing. The final result reveals the postprocessed Lax–Wendroff solution showing high resolution of the shock and smooth behavior in the continuous regions. Finally, we construct a new splitting scheme containing the algorithmic ingredients described above.

The final algorithm may be displayed in form of the following flowchart.

- In every time step:
  1. Compute a numerical solution $U(t+\Delta t)$ with a finite difference method.
  2. If necessary, filter high frequency components by means of $U_\delta := K_\delta * U$, where K_δ is the Gaussian kernel corresponding to a standard deviation of δ.
  3. Compute the structure tensor $J_0(\nabla U_\delta)$ which contains information about the local coherence of the numerical solution.
  4. Average the structure information in the vicinity of each grid point in order to define a region size in which the orientation of the solution is

examined. This corresponds to computing $J_\rho(\nabla U_\delta) := K_\rho * J_0(\nabla U_\delta)$.

5. Construct an artificial dissipation D from knowledge contained in J_ρ.
6. Solve the discrete version of the nonlinear anisotropic equation

$$\begin{aligned}\partial_t w &= \operatorname{div}(D(w)\operatorname{grad} w),\\ w(x,y,0) &= U(x,y).\end{aligned}$$

- End of time step: set $U(t+\Delta t) := w(\Delta t)$.

2. Conservation laws and finite difference schemes. Consider the scalar conservation law

$$\partial_t u + \partial_x f(u) + \partial_y g(u) = 0 \tag{2.1}$$

on $\Omega := [0,1]^2$, where we assume Cauchy data $u_0(x,y) = u(x,y,0)$ as well as boundary data which may respect the characteristic directions. As is well known, discontinuities develop in general within finite time regardless how smooth the initial data is chosen.

We consider conservative three-point finite difference approximations

$$\begin{aligned}U_{i,j}^{n+1} = U_{i,j}^n &- \Delta t \frac{1}{\Delta x}\left[\Theta(U_{i+1,j}^n, U_{i,j}^n) - \Theta(U_{i,j}^n, U_{i-1,j}^n)\right]\\ &- \Delta t \frac{1}{\Delta y}\left[\Xi(U_{i,j+1}^n, U_{i,j}^n) - \Xi(U_{i,j}^n, U_{i,j-1}^n)\right]\end{aligned}$$

of (2.1), where Θ and Ξ denote numerical flux functions consistent in the sense of $\Theta(s,s) = f(s)$ and $\Xi(s,s) = g(s)$ for all $s \in \mathbb{R}$. It is well known that every numerical flux of a three-point scheme can be written in the viscosity form

$$\Theta(v,w) := \frac{1}{2}(f(v) + f(w)) - \mathcal{Q}(v,w)(v-w),$$

where $\mathcal{Q}$ is the numerical viscosity coefficient; see [11].

It was shown by Tadmor in [11] and [12] that in the class of monotone, and hence first-order schemes, there exists a minimax pair in the sense that a scheme with numerical viscosity coefficient $\mathcal{Q}$ which satisfies the inequality

$$\forall v, w \in S \subset \mathbb{R}: \quad \mathcal{Q}^G(v,w) \le \mathcal{Q}(v,w) \le \mathcal{Q}^{mLF}(v,w)$$

is entropy stable, i.e., converges to the entropy weak solution. Here S denotes the state space of possible values of the solution. It turns out that the minimax pair is given by two well-known finite difference schemes, namely the Godunov scheme, corresponding to $\mathcal{Q}^G$, and the (modified) Lax–Friedrichs scheme, corresponding to $\mathcal{Q}^{LF}$. Both schemes are only first order accurate so that there is need for a constructive recipe giving higher order schemes from lower order ones.

In principle there are two different ways for the construction of higher order numerical methods. One could start directly with the construction of a numerical viscosity coefficient which gives higher order as well as stability. This is the way chosen by Jameson, Schmidt, and Turkel in the early 1980s; see [3]. The construction is difficult due to the inherent nonlinearity of the problem, and a well-behaving dissipation is hard to obtain. However, Jameson's codes still belong to the most successful pieces of software ever written and are known for their flexibility as well as for their stability. The other route leading to stable higher order schemes developed into a mainstream in the mid 1980s. Here one starts with a lower order monotone scheme, recovers the solution (with a MUSCL or ENO technique), and inserts the recovered solution into the low order flux functions. The recipe is quite general, and the most spectacular finite difference schemes of our time, like the TVD or ENO schemes, rely on that construction. Although the process of recovery in itself is not an easy task, we know now quite well how this can be done even on unstructured grids. In this class of methods we do not even see the numerical dissipation of our methods explicitly, which may seem to be a big advantage over the Jameson-type schemes.

However, in recent years a theory of nonlinear anisotropic diffusion was developed within the community of image processing. The algorithms developed there are very well suited for use in the numerical solution of conservation laws. In doing so, we proceed again along the lines of Jameson, Schmidt, and Turkel and try to construct a reasonable dissipation coefficient directly, but we shall see that the algorithmic background is so well developed that we arrive safely at new nonlinear anisotropic dissipation terms, which turn the oscillating Lax–Wendroff scheme into a high resolution method.

3. The numerical solution viewed as a picture. It is well known that the Lax–Wendroff formula is a second-order finite difference scheme for (2.1). In two dimensions there are a variety of different implementations, but we have chosen the one described by Shokin [7], i.e.,

$$\begin{aligned}
&\frac{U_{i,j}^{n+1}-U_{i,j}^{n}}{\Delta t}+\frac{F_{i+1,j}^{n}-F_{i-1,j}^{n}}{2\Delta x}+\frac{G_{i,j+1}^{n}-G_{i,j-1}^{n}}{2\Delta y}\\
&\quad=\frac{\kappa_x}{2}\left[A_{i+1/2,j}^{n}\left(\frac{F_{i+1,j}^{n}-F_{i,j}^{n}}{\Delta x}+\frac{G_{i+1/2,j+1/2}^{n}-G_{i+1/2,j-1/2}^{n}}{\Delta y}\right)\right.\\
&\quad\left.-A_{i-1/2,j}^{n}\left(\frac{F_{i,j}^{n}-F_{i-1,j}^{n}}{\Delta x}+\frac{G_{i-1/2,j+1/2}^{n}-G_{i-1/2,j-1/2}^{n}}{\Delta y}\right)\right]\\
&\quad+\frac{\kappa_y}{2}\left[B_{i,j+1/2}^{n}\left(\frac{F_{i+1/2,j+1/2}^{n}-F_{i-1/2,j+1/2}^{n}}{\Delta x}+\frac{G_{i,j+1}^{n}-G_{i,j}^{n}}{\Delta y}\right)\right.\\
&\quad\left.-B_{i,j-1/2}^{n}\left(\frac{F_{i+1/2,j-1/2}^{n}-F_{i-1/2,j-1/2}^{n}}{\Delta x}+\frac{G_{i,j}^{n}-G_{i,j-1}^{n}}{\Delta y}\right)\right].
\end{aligned}\tag{3.1}$$

Here, $F_{i,j}^n := f(U_{i,j}^n)$ and

$$\begin{aligned}
A_{i\pm1/2,j}^{n} &:= (f'(U_{i\pm1,j}^{n})+f'(U_{i,j}^{n}))/2,\\
B_{i,j\pm1/2}^{n} &:= (g'(U_{i,j\pm1}^{n})+g'(U_{i,j}^{n}))/2,\\
G_{i\pm1/2,j\pm1/2}^{n} &:= (G_{i\pm1,j\pm1}^{n}+G_{i,j})/2,
\end{aligned}$$

etc. The grid coefficients are $\kappa_x := \Delta t/\Delta x, \kappa_y := \Delta t/\Delta y$.

If σ denotes the maximum value of A and B, and if we assume $\Delta x = \Delta y =: h$, then the Lax–Wendroff scheme can be shown to be linearly stable under the somehow pessimistic CFL condition

$$\frac{\Delta t}{h}\sigma \le \frac{1}{2\sqrt{2}}.$$

This was the condition which was implemented in all our test cases.

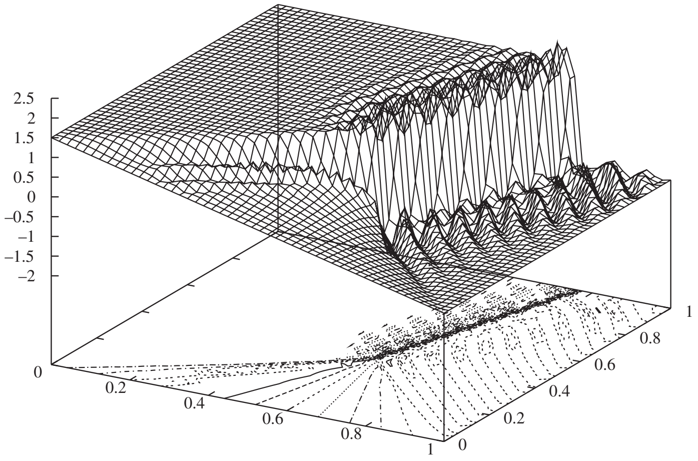


FIG. 1. *Lax–Wendroff solution of a test problem.*

If we now apply the Lax–Wendroff scheme to the boundary value problem (2.1), with $f(u) = 0.5u^2$, $g(u) = u$, and

$$u(x, y, 0) = \begin{cases} 1.5, & x = 0, \\ -2.5x + 1.5, & y = 0, \\ -1.0, & x = 1, \\ 0.0 & \text{else} \end{cases}$$

with 50 × 50 points and determine the boundary condition on the upper side of the unit square through simple extrapolation, then we get a steady solution as shown in Figure 1. The true solution consists of a fan-like continuous wave which develops into a shock. A schematic view of it can be seen in Figure 2.

Since the true solution satisfies the steady equation

$$\partial_y u + \partial_x \frac{u^2}{2} = 0$$

the characteristic equations are given by $dy/ds = 1$, $dx/ds = u$, i.e.,

$$\frac{dy}{dx} = \frac{1}{u}.$$

If we denote by u_L and u_R the given left and right state at $y = 0$, respectively, and we assume a linear distribution

$$u(x, 0) = (u_R - u_L)x + u_L$$

of the boundary data at $y = 0$, then the equation of the leftmost characteristic g_1 is given by $y = x/u_L$. The rightmost characteristic g_2 is given by $y = (x - 1)/u_R$. They meet at the point P where the shock g_3 starts. The coordinates of P are easily computed to be $x_P = u_L/(u_L - u_R)$ and $y_P = 1/(u_L - u_R)$. From the Rankine–Hugoniot condition we get for the shock g_3 the slope

$$\frac{dy}{dx} = \frac{2}{u_L + u_R}$$

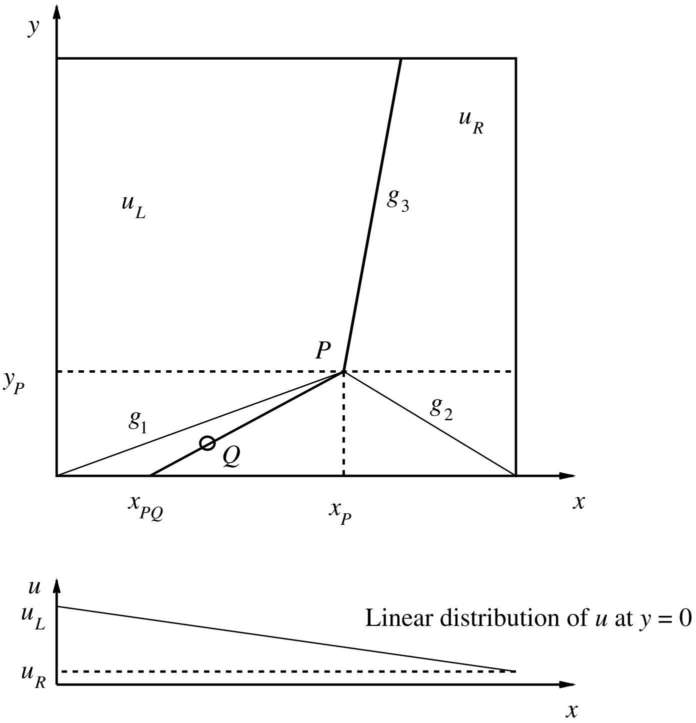


FIG. 2. *True solution of the model problem in the (x, y) plane.*

and finally the equation $y = (2x - 1)/(u_L + u_R)$. From these equations it is easy to compute the true solution pointwise. If the solution is to be known at a point Q lying within the fan region, then the characteristic connecting P and Q meets the x-axis at the point $x_{PQ} = x_P + y_p(x_Q - x_P)/(y_P - y_Q)$ which, according to our assumed linear boundary data distribution at $y = 0$, leads to $u_Q = (u_R - u_L)x_{PQ} + u_L$, which completes the description of the true solution.

In Figure 3 we plotted the pointwise difference between the numerical and the true solution. As can be observed from the numerical solution, there are strong oscillations present in the numerical solution, a behavior which the Lax–Wendroff formula shares with other second-order schemes which do not respect monotonicity conditions. We are now going to correct these behaviors by means borrowed from image analysis.

If we look at the isolines we can think of them as being a photograph. The numerical solution as plotted above the isolines is then interpreted as the grey level function. Obviously, there is noise in the picture (the oscillating part), but there is also an edge (the shock) which is the main feature of the picture. We would now like to do two things, namely

(i) enhance the edge and
(ii) filter the oscillations.

In classical construction of artificial dissipation terms the fullfilment of both requirements leads into trouble. While a diffusion certainly filters the high frequency oscillations it would also deteriorate the quality of the shock. In Jameson's dissipation model there is therefore a shock sensor, modeled by the second derivative of the pressure, which cuts the dissipation off across shocks.

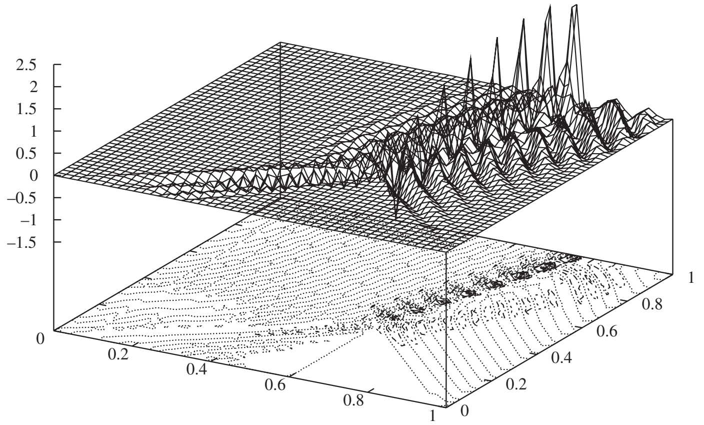


FIG. 3. *Difference between the Lax–Wendroff solution and the true solution.*

However, one would like to introduce dissipation parallel to the edges but avoid dissipation across edges. Dissipation models which satisfy exactly these requirements can be found in nonlinear anisotropic diffusion models from image processing.

4. Structure tensors and dissipation models. Edge enhancement is one of the classical problems in image processing. It was clear quite early in the history of this topic that edge detection without smoothing would lead to unacceptable results due to the noise; see [13]. Therefore, edge detection is only useful if appropriate smoothing is applied beforehand. To accomplish this task Gaussian smoothing may be applied; i.e., the numerical solution U is convolved with the Gaussian kernel

$$U_\delta := K_\delta * U, \quad K_\delta(x,y) := \frac{1}{2\pi\delta^2}\exp\left(-\frac{x^2+y^2}{2\delta^2}\right).$$

The parameter δ is the width of the Gaussian. We do not intend, of course, to dive into a continuous scale-space theory like Weickert [14] or Morel and Solimini [5] in their respective books. In our framework we are given nothing but approximations of Dirac functionals $\langle \delta_{(x_i,y_j)}, u\rangle =: u_{i,j}$ at time $t = n\Delta t$, which we denote by $U^n_{i,j}$. Rearranging all those values (in lexicographical order, say) results in a vector $U := (U^n_{i,j})_{1\le i\le I, 1\le j\le J}$. Of course it would be possible to construct a smooth interpolant of U, but this is already away from the heart of finite difference methods. Hence, the convolution with the Gaussian kernel indicated above is a discrete evolution in our case.

If we consider the continuous model, this type of smoothing is equivalent to solving the heat equation

$$\begin{aligned}\partial_t w &= \Delta w,\\ w(x,y,0) &= U(x,y),\end{aligned}$$

where pseudotime and width are connected via

$$T = \frac{1}{2}\delta^2;$$

i.e., the initial value problem for the heat equation is to be solved until pseudotime $t = T$ is reached. Care has to be taken in order to choose δ. Too large a δ would be

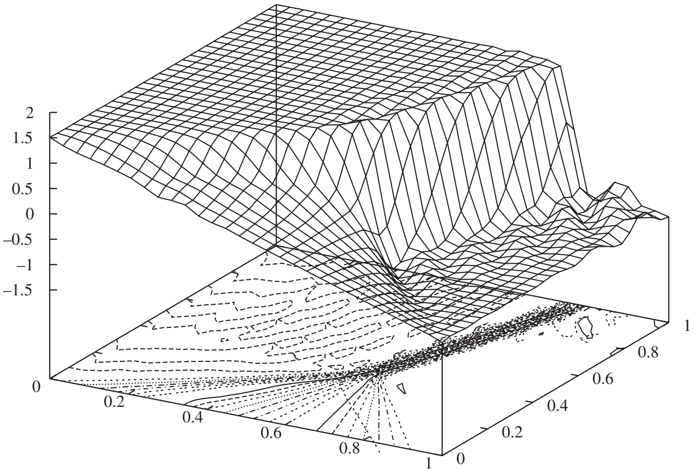


FIG. 4. *Result after presmoothing.*

overdiffusive, while a very small δ would not reduce the high-frequency noise. Since we work on the discrete level the initial value problem for the heat equation is best cast in the discrete form

$$\begin{aligned} U^0_{i,j;\delta} &:= U_{i,j}, \\ U^{m+1/2}_{i,j;\delta} &= U^m_{i,j;\delta} + \Delta t \frac{U^m_{i+1,j;\delta} - 2U^m_{i,j;\delta} + U^m_{i-1,j;\delta}}{\Delta x^2}, \\ U^{m+1}_{i,j;\delta} &= U^{m+1/2}_{i,j;\delta} + \Delta t \frac{U^{m+1/2}_{i,j+1;\delta} - 2U^{m+1/2}_{i,j;\delta} + U^{m+1/2}_{i,j-1;\delta}}{\Delta y^2}, \end{aligned} \tag{4.1}$$

where iteration along pseudotime is performed until the iteration index m is some m_{stop} corresponding to $T = \delta^2/2$. At the stopping time we set $U_{i,j;\delta} := U^{m_{\text{stop}}}_{i,j;\delta}$.

Note that we utilize an explicit splitting scheme which is stable under the CFL condition $\Delta t / \min(\Delta x^2, \Delta y^2) \le 1/2$; see [8]. An implicit discretization would be better behaved, but δ is so small that an explicit splitting is advantageous with respect to computational effort. As an example, we show in Figure 4 the result of a presmoothing with $\delta = 0.02$. In comparison with Figure 1, the filtering influence with respect to high frequency modes can be observed. For the sake of comparison, we again plot the difference between our numerical solution (now after filtering) and the true solution in Figure 5. Note that the error is now already concentrated in the vicinity of the shock. However, the application of the linear heat equation seems to be a dangerous step within the overall algorithm. While high-frequency modes are in fact damped the shock structure deteriorates very fast. Thus, it seems wise to start with a nonlinear diffusion equation directly on the unfiltered data and leave the task of filtering completely to this nonlinear device. From the presmoothed solution U_δ, we compute the structure tensor

$$J_0(\nabla U_{i,j;\delta}) := \nabla U_{i,j;\delta} \cdot \nabla U^T_{i,j;\delta}. \tag{4.2}$$

Since we are still as discrete as possible the meaning of our operator is defined as

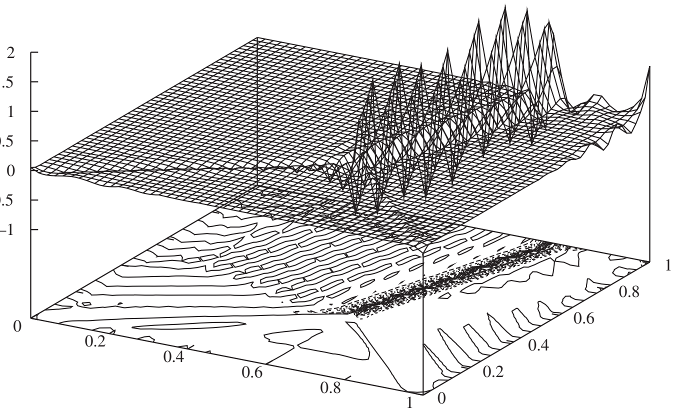


FIG. 5. *Difference between the presmoothed solution and the true solution.*

$$\nabla U_{i,j;\delta} := \begin{pmatrix} \dfrac{U_{i+1,j;\delta} - U_{i-1,j;\delta}}{2\Delta x} \\ \\ \dfrac{U_{i,j+1;\delta} - U_{i,j-1;\delta}}{2\Delta y} \end{pmatrix}. \tag{4.3}$$

Note that the structure tensor is symmetric positive semidefinite.

An easy calculation reveals the eigenvalues

$$\lambda_1 = |\nabla U_{i,j;\delta}|^2, \quad \lambda_2 = 0,$$

corresponding to the eigenvectors

$$v_1 = \nabla U_{i,j;\delta}, \quad v_2 = \nabla^{\perp} U_{i,j;\delta},$$

where

$$\nabla^{\perp} U_{i,j;\delta} := \begin{pmatrix} -\left(\dfrac{U_{i,j+1;\delta} - U_{i,j-1;\delta}}{2\Delta y}\right) \\ \\ \dfrac{U_{i+1,j;\delta} - U_{i-1,j;\delta}}{2\Delta x} \end{pmatrix},$$

so that $\nabla U^T_{i,j;\delta} \cdot \nabla^{\perp} U_{i,j;\delta} = 0$. In fact, the eigenvectors of the structure tensor define the direction parallel to and across an edge, respectively. In the framework of image processing the eigenvalues give the contrast (i.e., the squared gradient) in the eigendirections.

Since the structure tensor is symmetric positive semidefinite, we have the splitting

$$J_0(\nabla U_{i,j;\delta}) = V \Lambda V^{-1}, \tag{4.4}$$

where V is the matrix (v_1, v_2) containing the eigenvectors and Λ is nothing but $\operatorname{diag}(\lambda_1, \lambda_2)$.

In order to average the structure tensor data in the vicinity of each grid point component-wise convolution with the Gaussian kernel of width ρ is applied, i.e.,

$$J_\rho(\nabla U_{i,j;\delta}) := K_\rho * (\nabla U_{i,j;\delta} \cdot \nabla U^T_{i,j;\delta}) \tag{4.5}$$

is computed. The width ρ again is a measure of the averaging region. In practice, we again solve the discrete heat equation component-wise for the structure tensor.

A simple computation concerning the matrix $J_\rho(\nabla U_{i,j;\delta}) =: \left(\begin{smallmatrix} j_{11} & j_{12} \\ j_{12} & j_{22}\end{smallmatrix}\right)$ reveals the eigenvalues

$$\lambda_{1,2;\rho} = \frac{1}{2}\left(j_{11} + j_{22} \pm \sqrt{(j_{11} - j_{22})^2 + 4j_{12}^2}\right), \tag{4.6}$$

the positive square root belonging to $\lambda_{1;\rho}$, which correspond to the eigenvectors

$$\begin{aligned} v_{1;\rho} &= \begin{pmatrix} 2j_{12} \\ j_{22} - j_{11} + \sqrt{(j_{11} - j_{22})^2 + 4j_{12}^2} \end{pmatrix}, \\ v_{2;\rho} &= \begin{pmatrix} j_{11} - j_{22} - \sqrt{(j_{11} - j_{22})^2 + 4j_{12}^2} \\ 2j_{12} \end{pmatrix}, \end{aligned} \tag{4.7}$$

which are again orthogonal. A nice interpretation of these quantities in the framework of image processing can be found in [14]. The parameter δ in the presmoothing processing is called the local scale or noise scale because the process of presmoothing neglects all scales smaller than $\mathcal{O}(\delta)$. In contrast, the parameter ρ is the integration scale indicating the size of the subregions in which the orientation of the numerical solution is analyzed. The eigenvalues $\lambda_{1,2;\rho}$, moreover, serve as descriptors of local structure. Constant solutions are characterized by $\lambda_{1;\rho} = \lambda_{2;\rho} = 0$, while the quantity

$$\left(\lambda_{1;\rho} - \lambda_{2;\rho}\right)^2 = \left(j_{11} - j_{22}\right)^2 + 4j_{12}^2$$

becomes large for anisotropic structures. In the language of image processing one speaks of $(\lambda_{1;\rho} - \lambda_{2;\rho})^2$ as a measure of local coherence.

Now that we have analyzed our numerical solution as if it was a photograph, we are finally looking for a nonlinear anisotropic diffusion equation of the form

$$\begin{aligned} \partial_t w &= \operatorname{div}\left(D(w)\operatorname{grad} w\right), \\ w(x,y,0) &= U(x,y), \end{aligned}$$

where the dissipation coefficient $D(w)$ makes use of the information contained in the structure tensor. It is here where we again use the machinery of image processing since we follow the ansatz

$$D(w) := V_\rho L V_\rho^{-1}, \tag{4.8}$$

where V_ρ contains the eigenvectors of J_ρ and $L = \operatorname{diag}(l_1, l_2)$ is a diagonal matrix the entries of which we have to choose properly. In order to recover shocks (or, equivalently, in order to enhance edges), the diffusivity l_1 perpendicular to edges should be reduced if the contrast $\lambda_{1;\rho}$ is high. This can be achieved by an anisotropic regularization of the Perona–Malik model [6], which can be found in Weickert's book [14]:

$$\begin{aligned} l_1 &= \vartheta(\lambda_{1;\rho}), \\ l_2 &= 1, \\ \vartheta(s) &= \begin{cases} 1, & s \le 0, \\ 1 - \exp\left(\frac{-C_m}{(s/\lambda)^m}\right), & s > 0. \end{cases} \end{aligned} \tag{4.9}$$

The values of m and C_m are chosen in such a way, that the so-called flux $\Phi(s) := s\vartheta(s)$ is increasing in an interval $s \in [0, \lambda]$ and decreasing in $s \in]\lambda, \infty[$. These choices depend on a one-dimensional analysis of the classical Perona–Malik model, and we refer the reader to Weickert's book for details. In agreement with Weickert we chose $m = 4$ and thus $C_4 = 3.31488$. The parameter λ can then be chosen freely.

5. Discretizing the diffusion equation.

After deriving the nonlinear anisotropic diffusion equation which will sharpen the shocks, this equation now needs to be discretized. Since we do not have a theory of a truly discrete diffusion equation to start with, but we have a partial differential equation, the discretization process may result in instabilities if done in a naive way.

As was shown by Weickert [14], there is always a finite difference stencil such that the resulting discretization leads to a stable scheme. Moreover, Weickert was able to prove that three directions suffice to discretize the anisotropic diffusion and the proof is constructive. We do not want to go into the details of Weickert's work but give a suitable discretization of $\operatorname{div}(D(w)\operatorname{grad} w)$, where we utilize the notation

$$D = \begin{pmatrix} a & b \\ b & c \end{pmatrix}.$$

Then, following Weickert's recipe, we get

$$\nabla \cdot (D(U_{i,j;\delta})\nabla U_{i,j;\delta}) = \sum_{k=-1}^{1}\sum_{\ell=-1}^{1} C_{i+k,j+\ell}U_{i+k,i+\ell;\delta}$$

with

$$\begin{aligned}
C_{i-1,j+1} &= \frac{|b_{i-1,j+1}| - b_{i-1,j+1}}{4\Delta x\Delta y} + \frac{|b_{i,j}| - b_{i,j}}{4\Delta x\Delta y},\\
C_{i-1,j-1} &= \frac{|b_{i-1,j-1}| + b_{i-1,j-1}}{4\Delta x\Delta y} + \frac{|b_{i,j}| + b_{i,j}}{4\Delta x\Delta y},\\
C_{i,j+1} &= \frac{c_{i,j+1} + c_{i,j}}{2\Delta y^2} - \frac{|b_{i,j+1}| + |b_{i,j}|}{2\Delta x\Delta y},\\
C_{i,j-1} &= \frac{c_{i,j-1} + c_{i,j}}{2\Delta y^2} - \frac{|b_{i,j-1}| + |b_{i,j}|}{2\Delta x\Delta y},\\
C_{i+1,j+1} &= \frac{|b_{i+1,j+1}| + b_{i+1,j+1}}{4\Delta x\Delta y} + \frac{|b_{i,j}| + b_{i,j}}{4\Delta x\Delta y},\\
C_{i+1,j-1} &= \frac{|b_{i+1,j-1}| - b_{i+1,j-1}}{4\Delta x\Delta y} + \frac{|b_{i,j}| - b_{i,j}}{4\Delta x\Delta y},\\
C_{i-1,j} &= \frac{a_{i-1,j} + a_{i,j}}{2\Delta x^2} - \frac{|b_{i-1,j}| + |b_{i,j}|}{2\Delta x\Delta y},\\
C_{i+1,j} &= \frac{a_{i+1,j} + a_{i,j}}{2\Delta x^2} - \frac{|b_{i+1,j}| + |b_{i,j}|}{2\Delta x\Delta y},\\
C_{i,j} &= -\frac{a_{i-1,j} + 2a_{i,j} + a_{i+1,j}}{2\Delta x^2}\\
&\quad - \frac{|b_{i-1,j+1}| - b_{i-1,j+1} + |b_{i+1,j+1}| + b_{i+1,j+1}}{4\Delta x\Delta y}\\
&\quad - \frac{|b_{i-1,j-1}| + b_{i-1,j-1} + |b_{i+1,j-1}| - b_{i+1,j-1}}{4\Delta x\Delta y}
\end{aligned}$$

$$+\frac{|b_{i-1,j}|+|b_{i+1,j}|+|b_{i,j-1}|+|b_{i,j+1}|+2|b_{i,j}|}{2\Delta x\Delta y}$$
$$-\frac{c_{i,j-1}+2c_{i,j}+c_{i,j+1}}{2\Delta y^2}$$

and employ a simple forward difference in time. For this discretization, Weickert has shown that stability in terms of a discrete maximum-minimum principle can only be proven if the spectral condition number of D is below 5.82. For larger condition numbers, he mentioned indications based on experiments that some weaker stability properties might exist.

We remark in passing that the approach presented above is fully conservative. This is expressed as the property of conservation of mean grey level in image analysis; see [14].

6. A new splitting scheme. In previous sections we have described in detail how algorithms and methodology established in image processing can be used in the framework of numerical methods for conservation laws. In order to get a true scheme for conservation laws and not just a postprocessing tool, we now consider the coupling

$$U_{i,j}^{n+1}=\mathcal{D}(\Delta\tau)\mathcal{C}(\Delta t)U_{i,j}^{n},$$

where $\mathcal{C}$ is the operator associated with the convective part (the Lax–Wendroff method in our setting) while $\mathcal{D}$ represents the operator of nonlinear anisotropic diffusion. Note that Δt is the time scale of our convection problem and for the nonlinear diffusion part. The above splitting is known to be of first order in time only, but our considerations also work out for more sophisticated splittings like that of Strang. Note that time accuracy in our steady test case is by no means mandatory. The splitting means that in each time step we apply the discrete convection (i.e., the Lax–Wendroff scheme) first and the nonlinear diffusion part afterwards.

Since the presmoothing step using the linear heat equation cannot be controlled efficiently (the numerical solution deteriorates massively if the smoothing variance is only marginally too high), this algorithmic step was simply left out. The anisotropic nonlinear diffusion equation was weighted with a factor $\Delta x\Delta y$ in order to guarantee consistency; i.e., we compute

$$\begin{aligned}
w_{i,j}^0 &:= \mathcal{C}(\Delta t)U_{i,j}^n,\\
w_{i,j}^1 &= w_{i,j}^0+\Delta t\Delta x\Delta y x\nabla\cdot(D(w_{i,j}^0)\nabla w_{i,j}^0),\\
U_{i,j}^{n+1} &= w_{i,j}^1.
\end{aligned}$$

Now the dose of dissipation depends on the spatial mesh as it should while the accuracy of the space discretization is retained. Note that in principle we can allow for more than one time step in the nonlinear diffusion equation. However, since we like to interpret this equation as an artificial dissipation, one step is natural. Our numerical experiments, furthermore, indicate that one diffusion step in fact is enough to achieve results with high resolution.

The parameter chosen for the nonlinear diffusion is $\lambda=10$, $\rho=2\sqrt{\Delta x}\sqrt{\Delta y}$.

In Figure 6 we show the numerical solution of our splitting scheme after 50, 100, and 150 time steps on the left side. On the right, the corresponding coherence measure

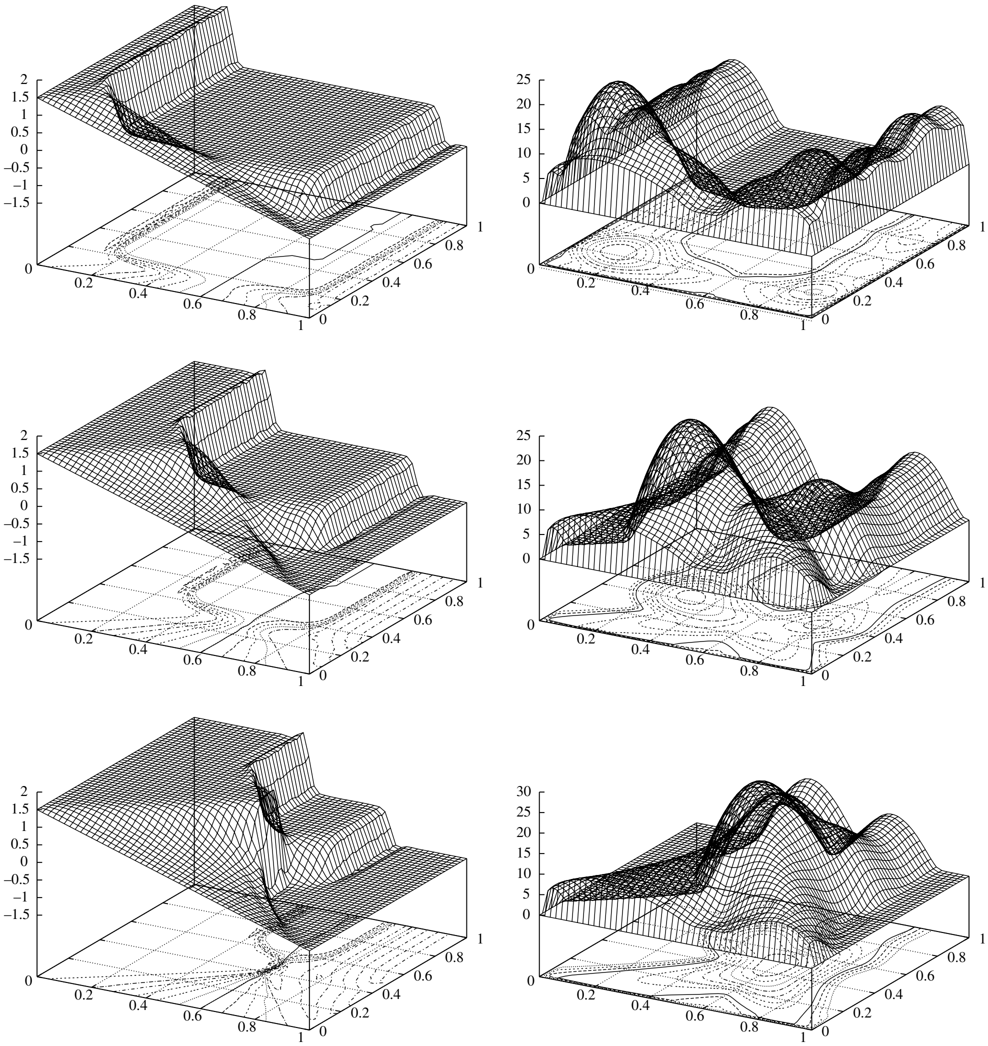


FIG. 6. *Numerical solution and coherence measure after* 50, 100, *and* 150 *time steps.*

$(\lambda_{1;\rho} - \lambda_{2;\rho})^2$ is plotted in logarithmic scale, where all data were shifted by 1 in order to avoid the computation of the logarithm of zero. One can see that this measure in fact indicates regions of anisotropic phenomena. Figure 7 shows the results at later times after 200, 250, and 300 time steps. The shock is now formed and is constantly sharpened by the diffusion step. Figure 8 shows the steady state (1000 time steps) and the corresponding coherence measure. Note that the shock is sharply resolved while there are marginal overshoots at the onset of the shock. In contrast to the result of the pure Lax–Wendroff scheme (cf. Figure 1) we observe that the splitting scheme with the new anisotropic nonlinear artificial dissipation behaves very nicely. In order to further reduce the small wiggles in the onset of the shock, the nonlinear diffusion tensor is weighted with the derivatives of the fluxes. This procedure is quite natural if dissipation models of classical finite difference schemes are analyzed; see [9], for example. Instead of considering the structure tensor $J_\rho(\nabla U) = \left(\begin{smallmatrix} j_{11} & j_{12} \\ j_{12} & j_{22} \end{smallmatrix}\right)$, we

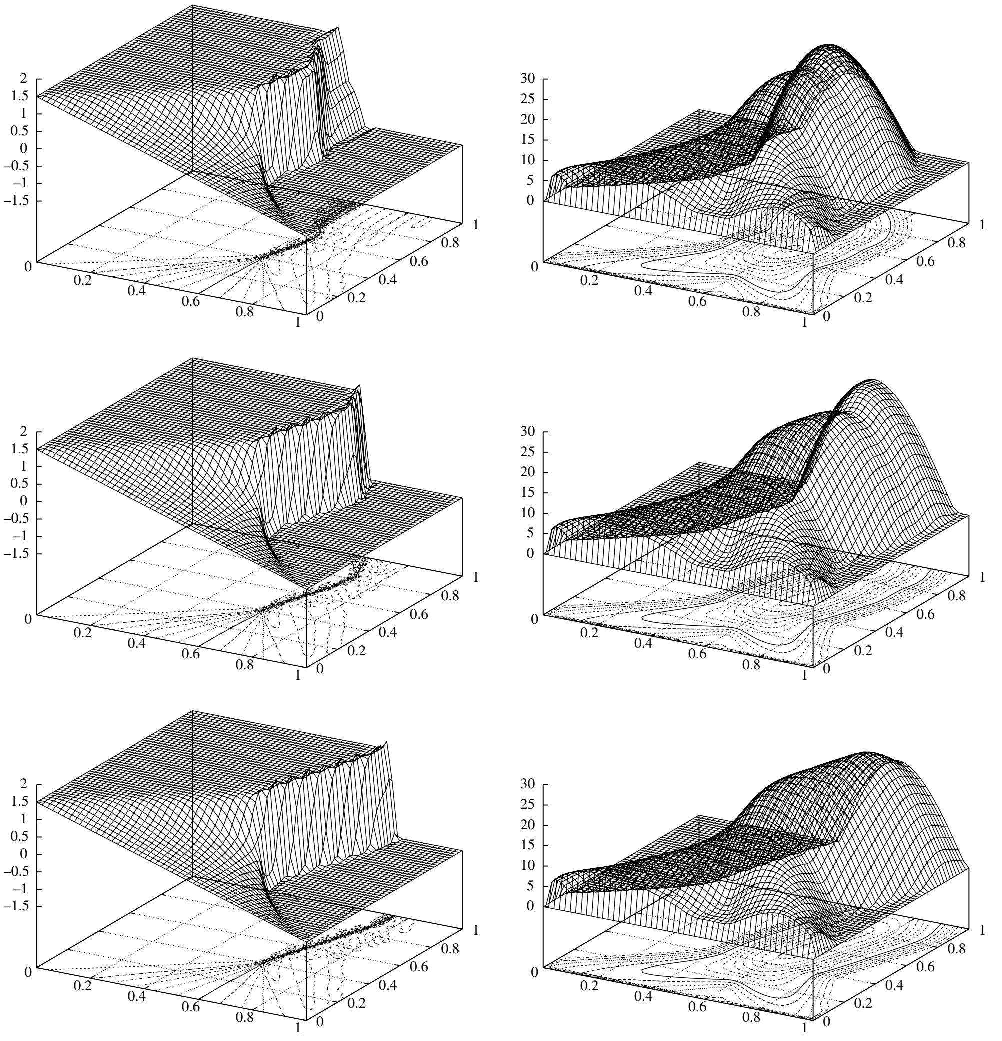


FIG. 7. *Numerical solution and coherence measure after* 200, 250, *and* 300 *time steps.*

therefore employ

$$\begin{pmatrix} j_{11}(f'(U))^2 & j_{12}f'(U)g'(U) \\ j_{12}f'(U)g'(U) & j_{22}(g'(U))^2 \end{pmatrix}.$$

In Figure 9 we again show the numerical solution of our splitting scheme at times $t = 50$, 100, 150. Figure 10 shows the steady state. Note that this solution exhibits not only a sharp shock transition but that it is also nearly free of any over- or undershoots. The solution is very close to solutions obtained with modern second-order TVD methods.

7. Conclusions. We have presented an approach to construct new artificial dissipation terms to be used in the computation of solutions to nonlinear conservation laws. These new dissipation models rely on well-known techniques in image processing and provide nonlinear and anisotropic artificial dissipation terms. While the construc-

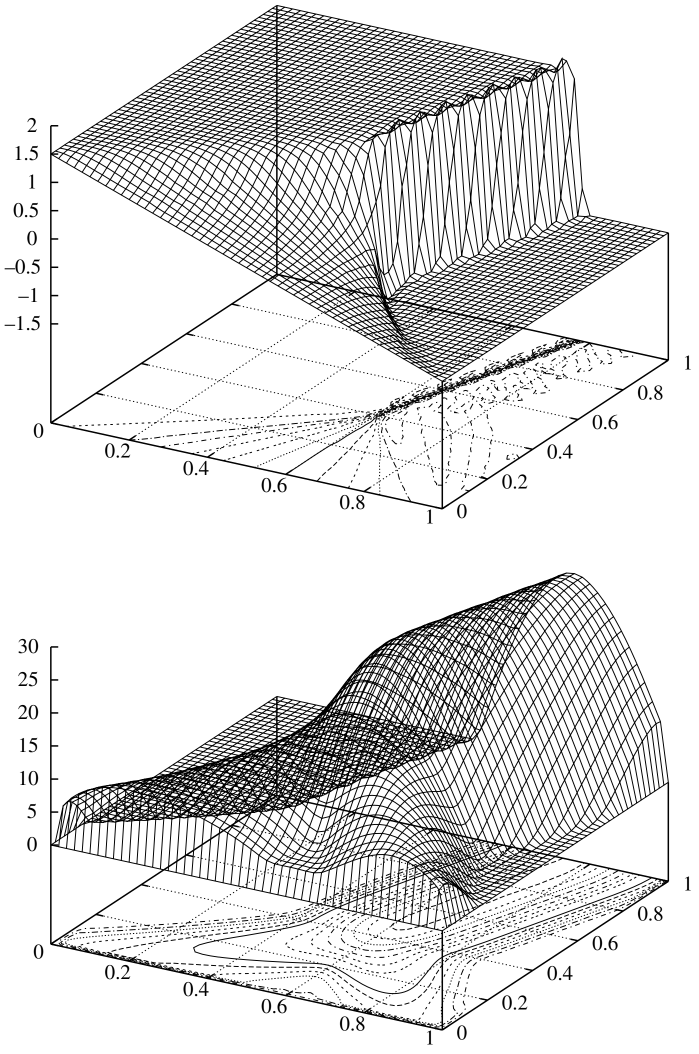


FIG. 8. *Numerical solution and coherence measure after* 1000 *time steps* (*steady state*).

tion of classical artificial dissipation is somehow ad hoc, the theory of anisotropic diffusion allows dissipation terms built on a sound mathematical footage. The approach is fully conservative because it is entirely based on conservation laws. Moreover, there is now hope to find useful dissipation terms which can be employed in meshless methods (see [1]), where up until now higher order discretizations are not achievable.

An extension for this approach to systems was presented in [2]. Nevertheless, since the control of four nonlinear diffusion equations in the case of the Euler equations in two dimensions is much more difficult than one nonlinear equation, a deeper understanding of the behavior and the analysis for the scalar case is needed.

Hence the class of new schemes constructed as described above have to be further examined concerning their accuracy and stability properties. This mathematical analysis will be the topic of future research.

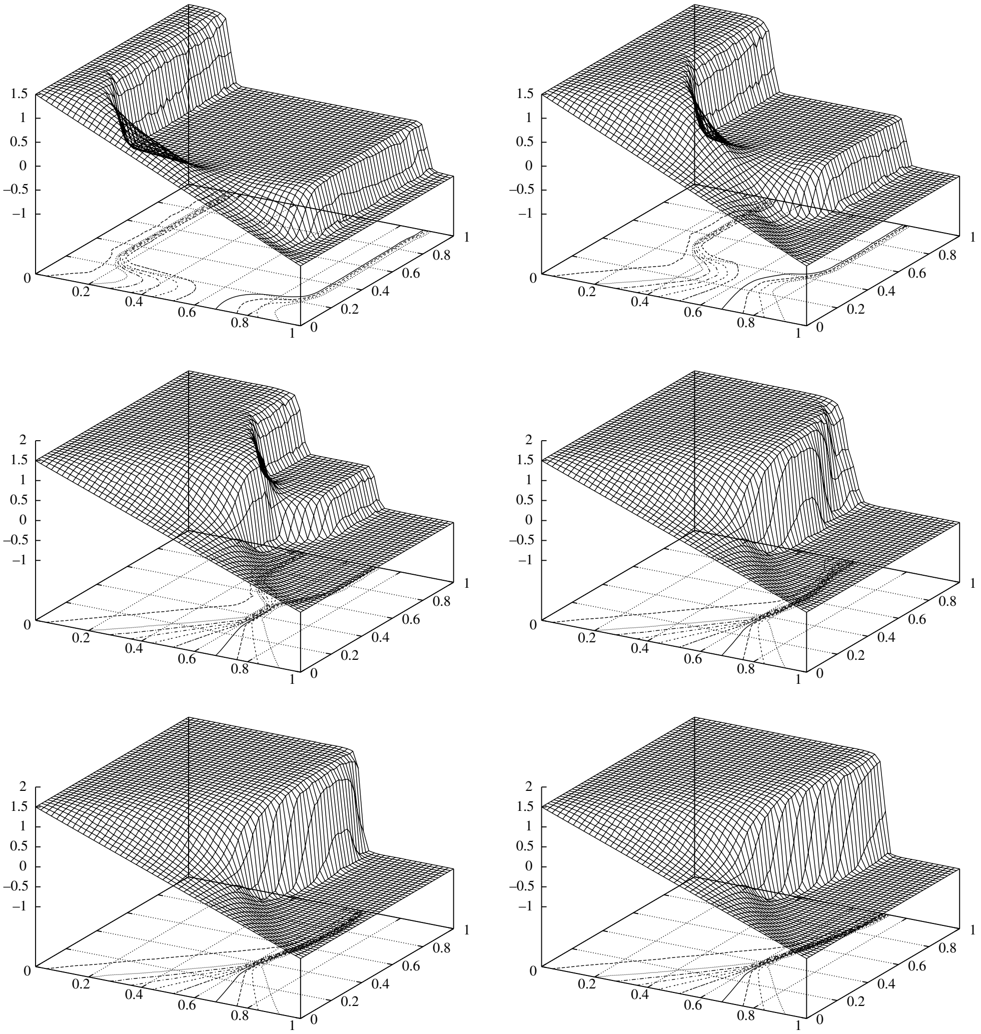


FIG. 9. *Numerical solution after* 50 *and* 100; 150 *and* 200; *and* 250 *and* 300 *time steps.*

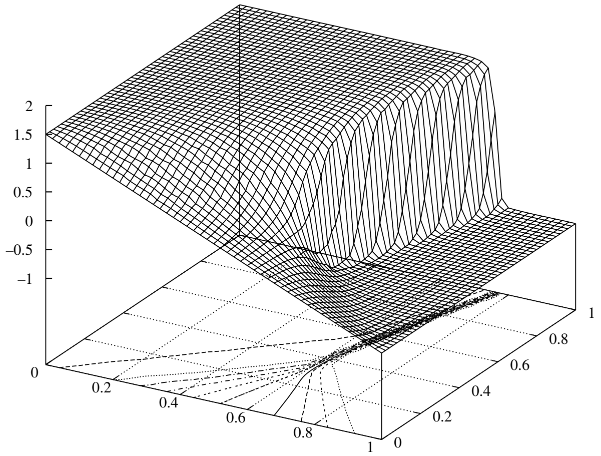


FIG. 10. *Numerical solution after* 1000 *time steps* (*steady state*).

REFERENCES

[1] J. Fürst and Th. Sonar, *On meshless collocation approximations of conservation laws: Positive schemes and dissipation models*, Z. Angew. Math. Mech., 81 (1998), pp. 403–415.

[2] Th. Grahs, A. Meister, and Th. Sonar, *Nonlinear anisotropic artificial dissipation—Characteristic filters for computation of the Euler equations*, in Finite Volumes for Complex Applications II, R. Vielsmeier, F. Benkhaldoun, and D. Hänel, eds., HERMES Science Publication, Paris, 1999, pp. 297–306.

[3] A. Jameson, W. Schmidt, and E. Turkel, *Numerical Solutions of the Euler Equations by Finite Volume Methods Using Runge Kutta Time Stepping Schemes*, AIAA paper 81-1259, 1981.

[4] A. Meister and Th. Sonar, *Finite volume schemes for compressible fluid flow*, Surveys Math. Indust., 8 (1998), pp. 1–36.

[5] J.-M. Morel and S. Solimini, *Variational Methods in Image Segmentation*, Birkhäuser, Basel, 1995.

[6] P. Perona and J. Malik, *Scale space and edge detection using anisotropic diffusion*, IEEE Trans. Pattern Anal. Mach. Intell., 12 (1990), pp. 629–639.

[7] Yu. I. Shokin, *The Method of Differential Approximation*, Springer-Verlag, Heidelberg, 1983.

[8] G.A. Sod, *Numerical Methods in Fluid Dynmaics*, Cambridge University Press, Cambridge, UK, 1985.

[9] Th. Sonar, *Entropy production in second-order three-point schemes*, Numer. Math., 62 (1992), pp. 371–390.

[10] Th. Sonar, *Mehrdimensionale ENO-Verfahren*, B.G. Teubner, Stuttgart, 1997.

[11] E. Tadmor, *The large-time behaviour of the scalar, genuinely nonlinear Lax-Friedrichs scheme*, Math. Comp., 43 (1987), pp. 353–368.

[12] E. Tadmor, *Numerical viscosity and the entropy condition for conservative difference schemes*, Math. Comp., 43 (1987), pp. 369–381.

[13] V. Torre and T.A. Poggio, *On edge detection*, IEEE Trans. Pattern Anal. Mach. Intell., 8 (1986), pp. 147–163.

[14] J. Weickert, *Anisotropic Diffusion in Image Processing*, B.G. Teubner, Stuttgart, 1989.

SIAM J. Sci. Comput.
Vol. 23, No. 5, pp. 1456–1463

THIRD ORDER ACCURATE LARGE-PARTICLE FINITE VOLUME METHOD ON UNSTRUCTURED TRIANGULAR MESHES*

SONG-HE SONG† AND MAO-ZHANG CHEN‡

Abstract. The large-particle fluid in cell (FLIC) method, presented in 1960s, is a numerical method that can be applied to solve unsteady flow. The computational scheme consists of two steps for each time march step: First, intermediate values are calculated for the velocities and energy, taking into account only the effects of acceleration caused by pressure gradients; second, transport effects are calculated. In this paper, we present a third order large-particle finite volume method for unstructured triangular meshes. The key idea of this method is the reconstruction of weighted quadratic interpolation for flow variables, and the two-point Gauss quadrature formula is used on mesh edges. The computational results for two typical flows verify the accuracy of the method.

Key words. large-particle method, unstructured mesh, finite volume method, interpolation polynomial

AMS subject classifications. 65M05, 65M10

S1064827599353105

1. Introduction.

There is a class of important Euler-type numerical methods for dealing with unsteady flow. Their common idea is to split the basic equations of fluid dynamics into two parts. One takes into account the effect of the pressure gradient. The other calculates the distribution of transport. Then the two parts are discretized respectively. So a time marching step consists of two steps. This kind of method includes particle in cell (PIC) [1], fluid in cell (FLIC) [2, 3, 11] (large particle), OIL [4], HELP [5], and so on. They play an important role in computational fluid dynamics (CFD). The idea of these methods has a great influence upon the development of CFD's numerical methods.

These methods, except PIC, are all upwind methods with first order accuracy with low resolution; in order to simulate shock wave or other discontinuity, artificial viscosity has to be added. Giving a general "Riemann" solver to first step (the contribution of pressure gradient), Li and Cao [6] and Li and Qian [7] obtained a large-particle method with second order accuracy by using the idea of Van Leer [8] and solving the "Riemann" problem. As no artificial viscosity is needed, their method has high resolution to discontinuous solution.

In order to simulate a general problem with very complicated computational domain, some methods are presented [9, 10, 11] on arbitrary triangular meshes, arbitrary quadrilateral meshes, or polygonal meshes, i.e., unstructured mesh FLIC methods. These methods have only first order accuracy and low resolution. [12] presents a large-particle finite volume method with second order accuracy on unstructured triangular meshes.

In this paper we will construct a third order accurate large-particle finite volume method on unstructured triangular meshes. The main idea of this method is the reconstruction of weighted quadratic interpolation for flow variables and the application

*Received by the editors January 5, 2000; accepted for publication (in revised form) May 1, 2001; published electronically January 4, 2002. This research was supported by NSF of China grant 10001038.

http://www.siam.org/journals/sisc/23-5/35310.html

†Department of Mathematics and System Science, National University of Defense and Technology, Changsha 410073, China (shsong@nudt.edu.cn).

‡Department of Jet Propulsion, Beijing University of Aeronautics and Astronautics, Beijing 100083, China (cmz@188.net).

of the two-point Gauss quadrature formula on each mesh edge. The “Riemann” problem is solved in the first step, and upwind method is applied in the second step. To make a comparison, the present method and another method [12] are used to simulate plate shock wave reflection and blunt cylinder flow. The computational results show that the transitional width of the shock wave by the present method is narrower than that of the other, and the numerical solution of the new method is nonoscillatory.

2. Decomposition of control equations. Euler equation for two-dimensional cases can be expressed as follows:

$$\left\{\begin{array}{rcl} \dfrac{\partial\rho}{\partial t}+\dfrac{\partial\rho u}{\partial x}+\dfrac{\partial\rho v}{\partial y} &=& 0,\\ \dfrac{\partial\rho u}{\partial t}+\dfrac{\partial(\rho u^2+p)}{\partial x}+\dfrac{\partial\rho uv}{\partial y} &=& 0,\\ \dfrac{\partial\rho v}{\partial t}+\dfrac{\partial\rho uv}{\partial x}+\dfrac{\partial(\rho v^2+p)}{\partial y} &=& 0,\\ \dfrac{\partial\rho E}{\partial t}+\dfrac{\partial(\rho uE+pu)}{\partial x}+\dfrac{\partial(\rho vE+pv)}{\partial y} &=& 0, \end{array}\right. \tag{2.1}$$

$$E=e+\frac{1}{2}(u^2+v^2),$$

where E is the specific total energy, e is the specific internal energy, and the equation of state is $p=(r-1)\rho e$.

The large-particle method consists of two steps for solving system (2.1).

Step 1. In this step, the density ρ is unchanged with time, and the intermediate values of velocities and specific total energy can be obtained by taking into account the contribution of pressure gradients. The differential equations of this step are as follows:

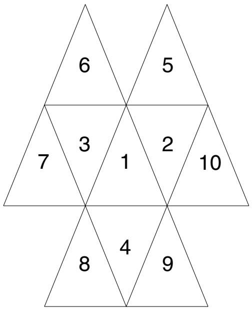
$$\left\{\begin{array}{rcl} \dfrac{\partial\rho u}{\partial t}+\dfrac{\partial p}{\partial x} &=& 0,\\ \dfrac{\partial\rho v}{\partial t}+\dfrac{\partial p}{\partial y} &=& 0,\\ \dfrac{\partial\rho E}{\partial t}+\dfrac{\partial pu}{\partial x}+\dfrac{\partial pv}{\partial y} &=& 0. \end{array}\right. \tag{2.2}$$

Step 2. In this step, the contribution of the transport of fluid is calculated, and the values of state variables are obtained at time $(n+1)\Delta t$. The differential equations are as follows:

$$\left\{\begin{array}{rcl} \dfrac{\partial\rho}{\partial t}+\dfrac{\partial\rho u}{\partial x}+\dfrac{\partial\rho v}{\partial y} &=& 0,\\ \dfrac{\partial\rho u}{\partial t}+\dfrac{\partial\rho u^2}{\partial x}+\dfrac{\partial\rho uv}{\partial y} &=& 0,\\ \dfrac{\partial\rho v}{\partial t}+\dfrac{\partial\rho uv}{\partial x}+\dfrac{\partial\rho v^2}{\partial y} &=& 0,\\ \dfrac{\partial\rho E}{\partial t}+\dfrac{\partial\rho uE}{\partial x}+\dfrac{\partial\rho vE}{\partial y} &=& 0. \end{array}\right. \tag{2.3}$$

The computational region is divided into many triangular cells. Every state variable is defined as an averaged value at the center of each triangular mesh. So the numerical result will give the averaged value at mesh for every variable.

3. Reconstruction of quadratic polynomial. Let the averaged value of each variable be given on every mesh; we will make use of these given conditions to construct a weighted quadratic polynomial on every triangular cell. At first we construct a "smoothest" linear polynomial on each cell. Based on each linear polynomial, we determine the cells required by constructing quadratic polynomials. The weighted quadratic reconstruction is carried out by making all quadratic polynomials on every cell weighted. The weighted factor is obtained by considering smoothness of the solution. Since only some parts of cells are used in this process, the process of interpolation is simplified.



Let c_1 be the center cell of the above figure; to construct a linear polynomial, we can choose any two cells from c_2, c_3, c_4. Let the constructed linear polynomial be

$$p_k^1(x,y) = a_0 + a_1\, x + a_2\, y \qquad (k = 1,2,3) \tag{3.1}$$

under the constraining condition

$$\frac{1}{|c_i|}\int_{c_i} p_k^1(x,y)\; dx\; dy = \bar{u}_i, \tag{3.2}$$

where i is the number of cells which determine $p_k^1(x,y)$.

We select a "smoothest" linear polynomial from (3.1), i.e., the value $|a_1| + |a_2|$ of the selected linear polynomial is the least among the three linear polynomials [15]. For convenience, we assume the selected linear polynomial is obtained on cells c_1, c_2, c_4. To construct a quadratic polynomial, we need another three constraining conditions (i.e., 3 cells). Since c_5 and c_{10} have common edges with c_2, c_8 and c_9 have common edges with c_4, and c_3 has a common edge with c_1, we select c_3 and any two cells from c_5, c_{10}, c_8, and c_9 as another three cells; then six quadratic polynomials can be obtained. We note quadratic polynomials by $p_k^2(x,y)$, $k = 1,2,\ldots,6$, with the constraining condition

$$\frac{1}{|c_i|}\int_{c_i} p_k^2(x,y)\; dx\; dy = \bar{u}_i, \tag{3.3}$$

$k = 1,2,\ldots,6$; c_i is the six corresponding cells in determining $p_k^2(x,y)$.

Let us define an indicator of smoothness IS_k for $p_k^2(x,y)$. Let Ω_k denote the connected region consisting of the six cells which are used to construct $p_k^2(x,y)$; we define IS_k as the summation of all square values of average difference of every two cells with common edge in Ω_k.

Let h be the maximum radius of the circumcircle of the triangles; we observe that if u is continuous on Ω_k, then $IS_k = O(h^2)$; and if u is discontinuous on Ω_k, then $IS_k = O(1)$.

Let

$$\alpha_k = \frac{1}{(\varepsilon + IS_k)^2}, \qquad \varepsilon = 10^{-5}; \tag{3.4}$$

define the weighted quadratic polynomial (WQP)

$$p^2(x,y) = \sum \frac{\alpha_k}{\sum \alpha_j} p_k^2(x,y); \tag{3.5}$$

then $\alpha_k / \sum \alpha_j$ is the weighted factor of $p_k^2(x,y)$. In smooth regions, we observe

$$\frac{\alpha_k}{\sum \alpha_j} = O(1), \tag{3.6}$$

while in discontinuous regions

$$\frac{\alpha_k}{\sum \alpha_j} = \frac{\frac{1}{(\varepsilon+O(1))^2}}{\sum \alpha_j} = \frac{O(1)}{\sum \alpha_j} \leq \max(O(\varepsilon^2), O(h^4)). \tag{3.7}$$

So the weighted quadratic polynomial requires that the interpolating polynomials on the smooth regions have contribution to WQP, while on the discontinuous regions they have essentially no contribution to WQP.

If $u(x,y)$ is an exact solution on a cell, $p^2(x,y)$ is a quadratic polynomial on the cell and it satisfies (3.3), then

$$p^2(x,y) - u(x,y) \;=\; O(h^3). \tag{3.8}$$

4. Discretization of equations. The large-particle method will be discussed in the following two steps.

Step 1. The integral form of (2.2) can be expressed as follows:

$$\begin{cases} \dfrac{\partial}{\partial t}\displaystyle\int_\Omega \rho u d\Omega + \int_{\partial\Omega} p \cdot n_x ds &= 0, \\ \dfrac{\partial}{\partial t}\displaystyle\int_\Omega \rho v d\Omega + \int_{\partial\Omega} p \cdot n_y ds &= 0, \\ \dfrac{\partial}{\partial t}\displaystyle\int_\Omega \rho E d\Omega + \int_{\partial\Omega} p \cdot (un_x + vn_y) ds &= 0. \end{cases} \tag{4.1}$$

The value of state variables on the edge of the triangular cell can be obtained by solving the “Riemann” problem [6]. We discretize (4.1) on triangle A:

$$\begin{cases} \tilde{u}_A = & \bar{u}_A - \frac{\Delta t}{|A|\bar{\rho}_A} \displaystyle\sum_{k=1}^{3} \left(\sum_{q=1}^{2} \omega_q \cdot p^*_{A_{k,q}} \right) \cdot n_x^k s_k, \\ \tilde{v}_A = & \bar{v}_A - \frac{\Delta t}{|A|\bar{\rho}_A} \displaystyle\sum_{k=1}^{3} \left(\sum_{q=1}^{2} \omega_q \cdot p^*_{A_{k,q}} \right) \cdot n_y^k s_k, \\ \tilde{E}_A = & \bar{E}_A - \frac{\Delta t}{|A|\bar{\rho}_A} \displaystyle\sum_{k=1}^{3} \left(\sum_{q=1}^{2} \omega_q \cdot p^*_{A_{k,q}} \right) \cdot U^{n*}_{A_{k,q}} s_k, \end{cases} \tag{4.2}$$

where $\bar{u}_A$ is the approximation of $\frac{1}{|A|}\int_A u d\Omega$, $p^*_{A_{k,q}}$ and $U^*_{A_{k,q}}$ are the pressure and component of the velocity vector along the direction normal to the triangular edge k, respectively, q is the qth Gauss quadrature point at the kth edge of triangle A, and s_k is the length of the kth edge of triangle A. $p^*_{A_{k,q}}$ and $U^*_{A_{k,q}}$ can be obtained by solving the "Riemann" problem. Now we describe the process briefly.

Along the normal direction at the Gauss quadrature point to a common edge, one considers the one-dimensional "Riemann" problem. The states on the left and right sides of the edge are as follows:

$$(\rho, U^n, p)_{L,q}, \quad (\rho, U^n, p)_{R,q}, \tag{4.3}$$

where U^n is the component of the velocity vector in the direction normal to the edge at the qth Gauss quadrature point, i.e., $U^n = n_x \cdot u + n_y \cdot v$. The value of flow variables at (4.3) can be calculated by WQP; then p^* and U^* can be obtained by solving the one-dimensional "Riemann" problem similar to [6].

Step 2. The integral form of (2.3) is as follows:

$$\left\{\begin{array}{rcl} \dfrac{\partial}{\partial t}\displaystyle\int_\Omega \rho d\Omega + \int_{\partial\Omega} \rho(un_x + vn_y)ds & = & 0, \\ \dfrac{\partial}{\partial t}\displaystyle\int_\Omega \rho u d\Omega + \int_{\partial\Omega} \rho u(un_x + vn_y)ds & = & 0, \\ \dfrac{\partial}{\partial t}\displaystyle\int_\Omega \rho v d\Omega + \int_{\partial\Omega} \rho v(un_x + vn_y)ds & = & 0, \\ \dfrac{\partial}{\partial t}\displaystyle\int_\Omega \rho E d\Omega + \int_{\partial\Omega} \rho E(un_x + vn_y)ds & = & 0. \end{array}\right. \tag{4.4}$$

We will get the approximate solution of the equations by means of the upwind method; for example, the approximation of the first equation in (4.4) is expressed as follows

$$\rho_A^{n+1} = \tilde{\rho}_A - \frac{\Delta t}{|A|}\sum_{k=1}^{3}\left(\sum_{q=1}^{2}\omega_q \tilde{\rho}_{A'_{k,q}}\tilde{U}_{A'_{k,q}}\right)s_k, \tag{4.5}$$

where $\tilde{U}_{A'_{k,q}} = \tilde{u}_{A'_{k,q}} \cdot n_x^k + \tilde{v}_{A'_{k,q}} \cdot n_y^k$, $\tilde{\rho}_{A'_{k,q}}$, $\tilde{u}_{A'_{k,q}}$, and $\tilde{v}_{A'_{k,q}}$ are the approximate values at the qth Gauss quadrature point of the kth edge in triangle A'; their values can be obtained by WQP. The fact that the cell A' represents cell A or its adjacent should be determined by the direction of normal velocity vector.

5. Numerical tests. Two typical flows were computed by use of the present method and another method [12].

Test 1. *Plate shock wave reflection.* The typical parameters of this example are as follows: Mach number is 2.9, the projection angle of shock wave is 29°, the length of x direction and y direction are 4 and 1, respectively. Figure 1 shows the Mach contours and Figure 2 shows the pressure distribution along the x direction at $y = 0.5$. Obviously, the transitional width of shock wave of this method is narrower than that of the other method [12]. By the use of the present method, the number of points in the transitional region of shock wave decreases from 10 to 6 and from 12 to 6 in the projection and reflection shock waves, respectively.

Test 2. *Blunt cylinder flow.* Mach number is 6.0, the radius of cylinder is 0.3. Figure 3 shows the Mach contours. Figure 4 shows the pressure distribution along flow direction at front of cylinder. Figure 5 shows the pressure distribution across the upper shock wave at $x = 1$. From Figures 4 and 5, we see that the calculated transitional width of shock wave by the new method is narrower than that of [12], and this new method is nonoscillatory.

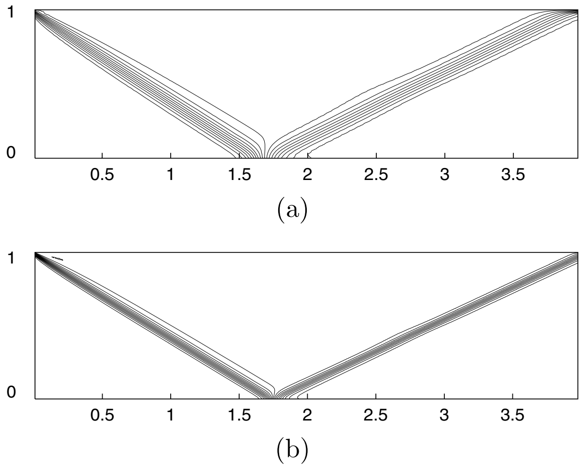


FIG. 1. *Mach contours for plate shock wave reflection in* (a) [12], (b) *this paper.*

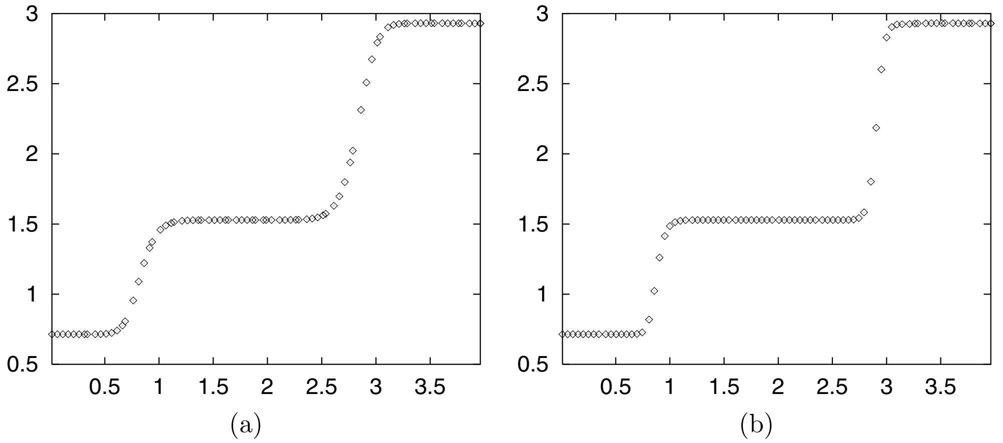


FIG. 2. *Pressure distribution along x direction at* $y = 0.5$ *in* (a) [12], (b) *this paper.*

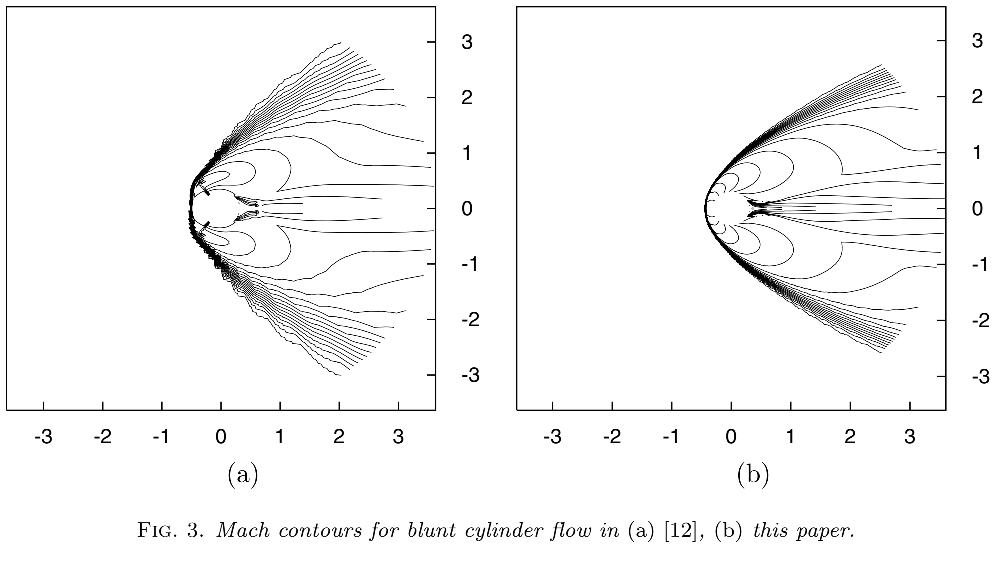


FIG. 3. *Mach contours for blunt cylinder flow in* (a) [12], (b) *this paper.*

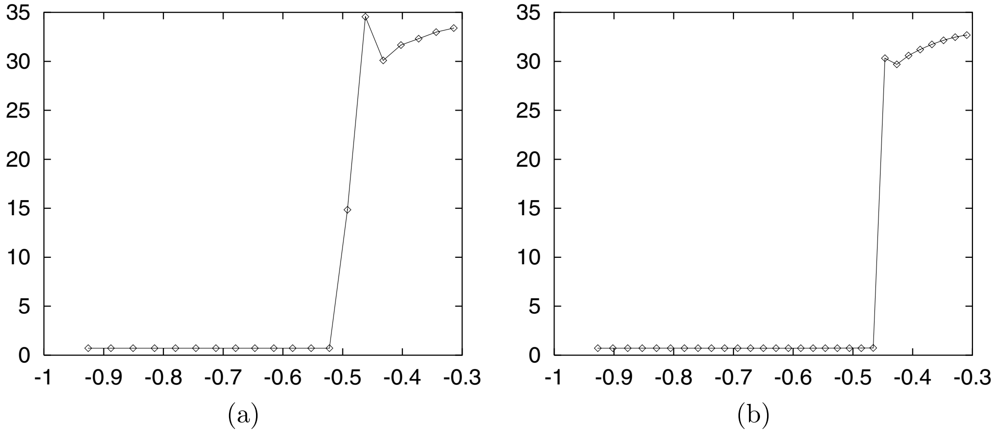


FIG. 4. *Pressure distribution at front of cylinder at $y = 0$ in* (a) [12], (b) *this paper.*

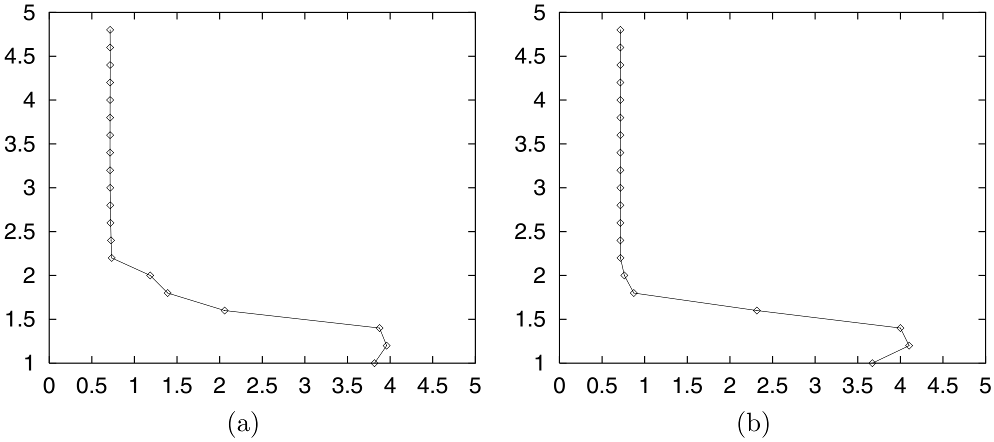


FIG. 5. *Pressure distribution across the upper shock wave at $x = 1$ in* (a) [12], (b) *this paper.*

6. Conclusion. According to the two tests, we can conclude that the accuracy of the present method is higher than that of the other method [12]. The calculated shock wave transitional region of the present method is narrower than that of [12], and it is nonoscillatory. The CPU time and memory of the new method are 280% and 85% more than that of [12], respectively.

REFERENCES

[1] F. H. HARLOW, *A Machine Calculation Method for Hydrodynamic Problems*, Report LAMS-1956, Los Alamos Science Laboratory, 1955.

[2] M. RICH, *A Method for Eulerian Fluid Dynamics*, Report LAMS-2826, Los Alamo Science Laboratory, 1963.

[3] R. A. GENTRY, R. E. MARTIN, AND B. J. DALY, *An Eulerian differencing method for unsteady compressible flow problems*, J. Comput. Phys., 1 (1966), pp. 87–118.

[4] W. E. JOHNSON, *OIL: A Continuous* 2*-Dimensional Eulerian Hydrodynamic Code*, LA-04-495-AMC-116(x), 1965.

[5] L. J. Hageman and J. M. Walsh, *HELP: A Multi-Material Eulerian Program for Compressible Fluid and Elastic-Plastic Flows in Two Space Dimension and Time*, Report 3SR-350, Ballistic Research Laboratory, 1971.

[6] Y. F. Li and Y. M. Cao, *"Large-particle" difference method with second-order accuracy in gasdynamics*, Sci. Sinica Ser. A, 28 (1985), pp. 1024–1035.

[7] Y. F. Li and E. P. Qian, *A "large-particle" difference method with second order accuracy computation of two-dimensional unsteady flows*, in Proceedings of the Tenth International Conference on Numerical Methods in Fluid Dynamics, Lecture Notes in Phys. 264, Springer-Verlag, Berlin, 1986, pp. 416–421.

[8] L. Van Leer, *Towards the ultimate conservative difference scheme: A second order sequel to Godunov method*, J. Comput. Phys. 32 (1979), pp. 101–136.

[9] U. Nobuhiro, S. Hiroshi, and A. Takesh, *On the numerical analysis of compressible flow problems by the "modified FLIC method,"* Comput. & Fluids, 8 (1980), pp. 251–262.

[10] Y. F. Li and Y. M. Cao, *On the "fluid in cell method" for arbitrarily triangular (or quadrilateral) mesh*, Math. Numer. Sin., 3 (1981), pp. 381–385.

[11] G. R. Xu, *An unsteady Eulerian difference scheme for arbitrary polygonal grids: A modified FLIC method*, Math. Numer. Sin., 6 (1984), pp. 429–433.

[12] S. H. Song and Y. F. Li, *High resolution large-particle finite volume method for* 2*-d unstructured triangular mesh*, Chinese J. Numer. Math. Appl., 19 (1997), pp. 8–14.

[13] S. H. Song, *Research on Finite Volume Method for Unstructured Meshes*, Ph.D. thesis, ICMSEC, Chinese Academy of Science, 1996.

[14] T. J. Barth, *Aspects of unstructured grids and finite volume solvers for the Euler and NS equations*, von Karman Institute for Fluid Dynamics, Lecture Series 1994-05, 1994.

[15] R. Abgrall, *On essentially non-oscillatory schemes on unstructured meshes: Analysis and implementation*, J. Comput. Phys. 114 (1994), pp. 45–58.

SIAM J. SCI. COMPUT.
Vol. 23, No. 5, pp. 1464–1484

HYBRID CURVE POINT DISTRIBUTION ALGORITHMS*

AHMED KHAMAYSEH† AND ANDREW KUPRAT‡

Abstract. Effective discretization of parametric curves has many applications in computational fields, ranging from evaluation and representation of complex geometries to geometric processing, grid generation, and numerical simulation. This paper presents a study of methods for point distribution on curves. Two direct approaches, *adaptive resolution refinement* and *hybrid grid point distribution*, are presented that generate high fidelity discrete approximations of parametric curves. Additionally, two smoothing techniques, *parametric* and *nonparametric curve grid smoothing*, are presented for the optimization of discrete curve data. This paper discusses these techniques in detail, presenting effective algorithms as well as a theoretical development of the methods.

Key words. point distribution density, adaptive resolution refinement, hybrid point distribution, curve grid smoothing

AMS subject classifications. 53A04, 65D10, 65M50, 68U05

PII. S1064827500367592

1. Introduction.

Many potential applications of effective curve point distribution algorithms exist, including support for geometric evaluation and representation such as curve-curve intersection operations, surface trimming, curve mapping and comparison, and as boundary discretization methods to support grid generation and field simulation. Computational applications based on discrete curves often require that the curve approximation is faithful to the actual curve within a tolerance bound with a computational representation that can be stored and evaluated effectively. Intuitively, it is not necessary to use many discrete points to represent a line or a curve with minimum curvature. However, to accurately capture a complex curve with a large variation in curvature requires many more points. Ideally, one desires an adaptive approach where few points are used in linear regions allowing the concentration of points in the areas of high curvature without using an excessive number of points across the global representation.

In the literature, the problem of sampling a given set of points to capture the physical features of the curve has been approached from several different angles. In particular, the curvature-based sampling of Kosters [1] minimizes the angular deviation of the surface normal along parametric curves. Li [2] develops an adaptive sampling technique based on solving a system of nonlinear equations derived from a minimization principle. Anderson, Khamayseh, and Jean [3] and Cuilliere [4] develop recursive refinement algorithms based on feature recognition and a local node placement strategy for adaptive sampling of high fidelity continuum parameterizations. Also in [4], Cuilliere presents a direct (i.e., noniterative) algorithm for node placement; however, this algorithm requires the curve to be twice differentiable. If

*Received by the editors February 11, 2000; accepted for publication (in revised form) August 16, 2001; published electronically January 4, 2002. The U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes. Copyright is owned by SIAM to the extent not limited by these rights.

http://www.siam.org/journals/sisc/23-5/36759.html

†Computer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831 (khamaysehak@ornl.gov).

‡Theoretical Division, Los Alamos National Laboratory, Los Alamos, NM 87544 (kuprat@lanl.gov).

the second derivative does not exist or is erratic, the curve discretization will not be guaranteed to be within the desired tolerance. This is a severe requirement since in many cases of geometric modeling, parametric curves are represented as B-splines with first derivatives discontinuous.

The accuracy of a numerical simulation on a field problem depends not only on the formal order of the approximation but also on the distribution of the grid points. An error analysis reveals that the accuracy of the approximation is related to the quality of the grid. Hoffman [5] and Vinokur [6] have analyzed the effect of the grid on truncation error for one-dimensional problems. In addition, curve point distribution (or stretching) functions are often used for distributing grid points along boundary curves of planar regions and surfaces, and along edges of three-dimensional regions. Boundary point distribution is often regarded as the initial step for algebraic or elliptic grid generation methods (see Mastin [7], Khamayseh and Kuprat [8], and Soni and Yang [9]).

In certain applications, curves obtained from physics-based simulations are frequently "jagged" or "nonsmooth," and are often unsuitable as direct input for subsequent simulations. For example, Potts model simulations of metallic grain growth describe interfaces between differing grains as a series of "stair-steps." The jagged stair-step interfaces are an artifact of the simulation and will produce incorrect results in subsequent simulations unless the interfaces are smoothed. Another example would be the Lagrangian surface motion of a fluid flow which could leave surfaces convoluted after a transient evolution; these surfaces exhibit high frequency artifacts that render the surfaces unsuitable for direct use in further calculations.

A popular approach to curve grid smoothing was developed by Taubin [10]. This method uses a low-pass filtering technique which removes large curvature variations and prevents shrinkage. Another approach to smoothing relies on the evolution of the curve grid by mean curvature, such as in Miller [11]. These methods can easily mitigate high frequency effects and stair-step phenomena in the grid but conserve neither the shape of the curve nor the area bounded by the curve.

This paper presents two efficient algorithms for the discretization of parametric curves. The first approach, *adaptive resolution refinement*, seeks to accurately decompose the curve for the purposes of geometric evaluation, processing, and visualization, while minimizing the total number of nodes needed for its representation. For grid generation and field simulation, the approach of *hybrid grid point distribution* is proposed. This algorithm supports the generation of grids on parametric curves allowing point distribution to be focused on parameters measuring the arclength, curvature, or specified attractors, alone or in unison, to achieve a suitable curve discretization.

In addition, two techniques for curve grid smoothing are examined, *parametric* and *nonparametric*. Parametric grid smoothing is used to obtain a smooth variation in discrete curve edge length along the curve without sacrificing the fidelity between the discrete approximation and the parametric basis of the curve. This method provides a smooth boundary edge discretization for surface grid generation applications.

The second technique, nonparametric area conserving smoothing, is targeted at applications where the input discrete approximation is suspect, but the integrated area enclosed is known with confidence. This approach, for example, will rapidly deform a "stair-stepped" closed curve into a smoothed curve enclosing the specified area (conserving that area to round-off). This technique of grid smoothing is used to obtain a smooth curve approximation by reducing high frequency variation of the input discrete data and the represented geometry. This development explores these techniques in some detail, presenting algorithms as well as the theoretical basis.

2. Curve adaptive resolution refinement. This section presents an automatic adaptive method for sampling the parametric domain to construct a grid with m segments on the represented curve in physical space. This discretization method seeks to produce a high fidelity approximation of the parametric curve. The final goal is to compute a minimal m such that the deviation of the discretized curve should be within a specified tolerance of the actual curve. The deviation is defined to be the distance of the piecewise linear polygonized curve from the true curve. It is desirable that the polygonized curve be close to the true curve, since it is usually the polygonized curve that is used in graphics applications such as rendering and region selection.

The development of this algorithm requires the specification of the mathematical form of the curve geometry. The following discussion assumes that the curve geometry is available as a *parametrically defined curve* such as a *conic curve, bezier curve, or B-spline curve.* This equates to the existence of a curve geometry definition in the form of a mapping $\mathbf{x}(u) = (x(u), y(u), z(u))$ from a *parametric u-domain* to a *physical (x, y, z)-domain.*

In curve point distribution, the discretization process may be described as the generation of a mapping from the discrete *computational ξ-domain* to the parametric u-domain, resulting in the composite map $\mathbf{x}(\xi) = (x(\xi), y(\xi), z(\xi))$. Physical space is a subset of $\mathbb{R}^3$, with parametric space a subset of $\mathbb{R}$, which may be defined to be the $[0, 1]$ unit interval. Traditionally, computational space is denoted as a discrete linear set of points $\xi \in \{0, 1, \ldots, m\}$, extended to the continuum $[0, m]$ for mathematical convenience. The task of curve point distribution is the task of specifying the distribution of the curve in the parametric domain

$$\big\{u(\xi) \;\big|\; 0 \leq \xi \leq m\big\}.$$

This curve in parametric space corresponds to the curve $\{\mathbf{x}(u) \mid 0 \leq u \leq 1\}$ in physical space. If we select a continuum parameterization $\{u(\xi) \mid 0 \leq \xi \leq m\}$, then the image of the curve $\mathbf{x}(u(\xi)) : [0, m] \to \mathbb{R}^3$ is, of course, equivalent to the image of the curve $\mathbf{x}(u) : [0, 1] \to \mathbb{R}^3$. However a "nearby" curve $\mathbf{x}^{\mathrm{PL}}(u(\xi)) : [0, m] \to \mathbb{R}^3$ can be constructed by defining

$$\mathbf{x}^{\mathrm{PL}}(\xi) = \mathbf{x}(\xi) \quad \text{if } \xi \in \{0, 1, \ldots, m\},$$

and defining $\mathbf{x}^{\mathrm{PL}}(\xi)$ to be the piecewise linear interpolant of these points if ξ is not an integer. $\mathbf{x}^{\mathrm{PL}}(\xi)$ is the "polygonized" version of $\mathbf{x}(u)$. It is our desire to find an integer m and a mapping $u(\xi) : [0, m] \to [0, 1]$ such that the greatest distance between $\mathbf{x}^{\mathrm{PL}}(\xi)$ and $\mathbf{x}(\xi)$ is bounded by a user-supplied tolerance ϵ.

We define deviation on an interval $[\xi, \xi + 1]$ by

$$\mathrm{dev}([\xi, \xi + 1]) = \max_{u(\xi) \leq u \leq u(\xi+1)} \min_{0 \leq t \leq 1} \|\mathbf{x}(u) - [t\mathbf{x}(u(\xi)) + (1 - t)\mathbf{x}(u(\xi + 1))]\|.$$

In order to find the desired function $u(\xi)$—or, equivalently, $\xi(u)$—we define the point density function $\rho \equiv \frac{d\xi}{du}$ so that

$$\xi(u) = \int_0^u \rho(w)\, dw. \tag{2.1}$$

Now relative to a line tangent to $\mathbf{x}(u)$ at u, we typically have that to leading order the distance from $\mathbf{x}(u)$ to the line is quadratic in u. This means the deviation on a typical interval obeys

$$\begin{aligned}\operatorname{dev}([\xi,\xi+1]) &= \max_{u(\xi)\le u\le u(\xi+1)} \min_{0\le t\le 1} \|\mathbf{x}(u) - [t\mathbf{x}(u(\xi)) + (1-t)\mathbf{x}(u(\xi+1))]\| \\ &\approx \left\|\mathbf{x}\left(u\left(\xi+\frac{1}{2}\right)\right) - \left[\frac{1}{2}\mathbf{x}(u(\xi)) + \frac{1}{2}\mathbf{x}(u(\xi+1))\right]\right\| \\ &\approx C\left(u\left(\xi+\frac{1}{2}\right)\right)\|\mathbf{x}_\xi\|^2.\end{aligned}$$

That is, the deviation in an interval is proportional to the square of the grid spacing with $C(u)$, the constant of proportionality, depending on the curve parameter u.

We wish to construct $u(\xi)$ so that $\operatorname{dev}([\xi,\xi+1])$ is the same on each interval $[\xi,\xi+1]$. Thus

$$\sqrt{\operatorname{dev}([\xi,\xi+1])} = K \tag{2.2}$$

$$= \sqrt{C(u)}\|\mathbf{x}_\xi\|. \tag{2.3}$$

Consider a uniform fine grid parameterization $u = \tilde{u}(\tilde{\xi}) = \frac{\tilde{\xi}}{M}$ with M large. Suppose $u(\xi) = \tilde{u}(\tilde{\xi})$. Then if $\widetilde{\operatorname{dev}}([\tilde{\xi},\tilde{\xi}+1])$ represents the maximum deviation suffered by the finely discretized curve in the interval $[\tilde{\xi},\tilde{\xi}+1]$, we have

$$\begin{aligned}\sqrt{\widetilde{\operatorname{dev}}([\tilde{\xi},\tilde{\xi}+1])} &= \sqrt{C\left(u\left(\tilde{\xi}+\frac{1}{2}\right)\right)}\|\mathbf{x}_{\tilde{\xi}}\| \\ &= \sqrt{C(u)}\|\mathbf{x}_u\|\tilde{u}_{\tilde{\xi}} \\ &= \sqrt{C(u)}\|\mathbf{x}_u\|\frac{1}{M}.\end{aligned} \tag{2.4}$$

Solving for $\sqrt{C(u)}$ in (2.4) and (2.3) and equating the expressions, we obtain

$$\begin{aligned}\frac{\sqrt{\widetilde{\operatorname{dev}}([\tilde{\xi},\tilde{\xi}+1])}}{\|\mathbf{x}_u\|\frac{1}{M}} &= \frac{K}{\|\mathbf{x}_\xi\|} \\ &= \frac{K}{\|\mathbf{x}_u\|u_\xi}.\end{aligned}$$

Therefore,

$$\frac{1}{u_\xi} = \frac{M}{K}\sqrt{\widetilde{\operatorname{dev}}([\tilde{\xi},\tilde{\xi}+1])}.$$

Since $\rho = \frac{1}{u_\xi}$, this says that we should set

$$\rho \propto \sqrt{\widetilde{\operatorname{dev}}([\tilde{\xi},\tilde{\xi}+1])}. \tag{2.5}$$

Our algorithm is thus to compute deviations on a fine uniform parameterization with M intervals, to set

$$\rho = \sqrt{\widetilde{\operatorname{dev}}([\tilde{\xi},\tilde{\xi}+1])}$$

on each interval $[\tilde{\xi}, \tilde{\xi}+1]$, $0 \le \tilde{\xi} \le M-1$, and then to normalize ρ so that

$$\int_0^1 \rho(w)\,dw = m, \tag{2.6}$$

where $m \ll M$ will be the actual number of points on the polygonized curve. With the determination of ρ, we obtain $\xi(u)$, which upon inversion gives us $u(\xi)$, $0 \le \xi \le m$, the locations of the nodes in parametric space for the polygonized curve.

Finally, we must determine an m such that the deviation on each interval is bounded by ϵ. Consider the cumulative distribution functions

$$\operatorname{dev}(\xi) = \sum_{i=0}^{\xi-1} \operatorname{dev}([i, i+1]), \quad 0 \le \xi \le m,$$

and

$$\widetilde{\operatorname{dev}}(\tilde{\xi}) = \sum_{i=0}^{\tilde{\xi}-1} \widetilde{\operatorname{dev}}([i, i+1]), \quad 0 \le \tilde{\xi} \le M.$$

Assume for the moment that $u(\xi)$ is a uniform parameterization (i.e., $u(\xi) = \frac{\xi}{m}$). In this case,

$$\operatorname{dev}(m) = \sum_{\xi=0}^{m-1} C\left(\frac{\xi+\frac{1}{2}}{m}\right)\left(\|\mathbf{x}_u\|\frac{1}{m}\right)^2$$

and

$$\begin{aligned}
\widetilde{\operatorname{dev}}(M) &= \sum_{\tilde{\xi}=0}^{M-1} C\left(\frac{\tilde{\xi}+\frac{1}{2}}{M}\right)\left(\|\mathbf{x}_u\|\frac{1}{M}\right)^2 \\
&\approx \frac{M}{m}\sum_{\xi=0}^{m-1} C\left(\frac{\xi+\frac{1}{2}}{m}\right)\left(\|\mathbf{x}_u\|\frac{1}{M}\right)^2 \\
&= \frac{m}{M}\sum_{\xi=0}^{m-1} C\left(\frac{\xi+\frac{1}{2}}{m}\right)\left(\|\mathbf{x}_u\|\frac{1}{m}\right)^2 \\
&= \frac{m}{M}\operatorname{dev}(m).
\end{aligned}$$

Hence,

$$M\,\widetilde{\operatorname{dev}}(M) \approx m\,\operatorname{dev}(m). \tag{2.7}$$

If we require that the average deviation in the polygonized curve satisfies

$$\frac{\operatorname{dev}(m)}{m} = \epsilon,$$

then in light of (2.7), we set

$$M\,\widetilde{\operatorname{dev}}(M) = \epsilon m^2,$$

so that

$$m = \sqrt{\frac{M\widetilde{\mathrm{dev}}(M)}{\epsilon}}. \tag{2.8}$$

If we use this value for m and compute $u(\xi)$ with (2.1), (2.5), (2.6) rather than $u(\xi) = \frac{\xi}{m}$ as supposed in the above estimate, we do not expect the average deviation $\frac{\mathrm{dev}(m)}{m}$ to increase because the distribution $u(\xi)$ is generated with the intent to attract grid nodes to regions of curvature and thus will probably decrease average deviation, given a fixed number of nodes. Given this, by (2.2) we have that $u(\xi)$ equidistributes deviation, and so we expect that

$$\max_{0\le\xi\le m-1} \mathrm{dev}([\xi,\xi+1]) \approx \frac{1}{m}\mathrm{dev}(m) \le \epsilon.$$

Thus, with the choice of m given by (2.8), we expect maximum deviation to be bounded by the user-supplied tolerance ϵ.

In the interest of formulating a general implementation, this development will further assume that the curve to be discretized is provided in the form of a nonuniform rational B-spline (NURBS) curve representation. A detailed discussion of NURBS is available in Piegl and Tiller [12]. A NURBS curve $\mathbf{x}(u) = (x(u), y(u), z(u))$ is a piecewise rational curve given as

$$\mathbf{x}(u) \;=\; \frac{\sum_{i=0}^{m} \;\omega_i\; \mathbf{d}_i\; N_i^k(u)}{\sum_{i=0}^{m}\; \omega_i\; N_i^k(u)}, \qquad u \in [u_{k-1}, u_{m+1}],$$

and defined by

- an order k,
- control points $\mathbf{d}_i = (x_i, y_i, z_i)$, $i = 0, \dots, m$,
- real weights ω_i, $i = 0, \dots, m$,
- a set of real knots $\{u_0, \dots, u_{m+k} \mid u_i \le u_{i+1},\ i = 0, \dots, (m+k-1)\}$,
- B-spline basis functions $N_i^k(u)$, $u \in [u_i, u_{i+k}]$, $i = 0, \dots, m$, where

$$N_i^k(u) = \frac{u - u_i}{u_{i+k-1} - u_i}\; N_i^{k-1}(u) + \frac{u_{i+k} - u}{u_{i+k} - u_{i+1}}\; N_{i+1}^{k-1}(u),$$

$$N_i^1(u) \;=\; \begin{cases} 1 & \text{if } u_i \le u < u_{i+1}, \\ 0 & \text{otherwise,} \end{cases} \qquad i = 0, \dots, m, \qquad \text{and}$$

- curve segments $\mathbf{x}_i(u)$, $u \in [u_i, u_{i+1}]$, $i = (k-1), \dots, m$.

Given this supplementary information, it is now possible to express the adaptive point distribution algorithm.

ALGORITHM 2.1. *Adaptive resolution refinement.*

Assume a NURBS curve $\mathbf{x}(u)$ with unique knots $\{u_l \mid 0 \le l \le n\}$. For each knot interval $I = [u_{l-1}, u_l]$ create an initial distribution of $(M+1)$ points $u_0^I = u_{l-1}, u_1^I, \dots, u_M^I = u_l$ that are used to compute the total deviation from the actual parametric curve. The user must specify a large enough M and maximum deviation tolerance ϵ. Typical values are $\epsilon = 10^{-2}$ for visualization purposes and $\epsilon = 10^{-5}$ for geometric computations.

1. Initialize the first adaptive grid point $u_0^* \leftarrow u_0$ and a point counter $k \leftarrow 1$.
2. For each unique knot interval $I = [u_{l-1}, u_l]$ do

a) Initialize grid function ξ to zero.

```
Do $i = 0, \dots, M$
  $\xi_i \leftarrow 0$
```

b) Subdivide the curve over the interval I.

```
Do $i = 0, \dots, M$
  $u_i^I \leftarrow \left(\frac{i}{M}\right) u_l + \left(1 - \frac{i}{M}\right) u_{l-1}$
```

c) Compute the grid function ξ by summing square roots of deviations.

```
$totaldev \leftarrow 0$
Do $i = 1, \dots, M$
  $\tilde{\mathbf{x}} \leftarrow \mathbf{x}(\frac{1}{2}(u_{i-1}^I + u_i^I))$
  $\mathbf{a} \leftarrow \mathbf{x}(u_i^I) - \mathbf{x}(u_{i-1}^I)$
  $\mathbf{b} \leftarrow \tilde{\mathbf{x}} - \mathbf{x}(u_{i-1}^I)$
  $t \leftarrow \frac{\mathbf{a} \cdot \mathbf{b}}{\mathbf{a} \cdot \mathbf{a}}$   (Usually $t \approx \frac{1}{2}$)
  $dev \leftarrow \| t\, \mathbf{x}(u_i^I) + (1-t)\, \mathbf{x}(u_{i-1}^I) - \tilde{\mathbf{x}} \|$
  $\xi_i \leftarrow \xi_{i-1} + \sqrt{dev}$
  $totaldev \leftarrow totaldev + dev$
```

d) Compute the number of adaptive refinement points over this interval using the total deviation and the user-specified epsilon.

```
$m_I \leftarrow \sqrt{\frac{M\ totaldev}{\epsilon}} + 1$
```

e) Increase the size of the array u^* by m_I to hold the additional adaptive refinement points.

f) Normalize the grid function ξ to have $\xi_M = m_I$.

```
Do $i = 1, \dots, M-1$
  $\xi_i \leftarrow m_I \frac{\xi_i}{\xi_M}$
$\xi_M \leftarrow m_I$
```

g) Obtain the point distribution by inverting the grid function.

```
$j \leftarrow 1$
Do $i = 1, \dots, M$
  Do while $(j \le \xi_i)$
    $u_k^* \leftarrow u_{i-1}^I \frac{\xi_i - j}{\xi_i - \xi_{i-1}} + u_i^I \frac{j - \xi_{i-1}}{\xi_i - \xi_{i-1}}$   (Obtain $u_{i-1}^I < u_k^* \le u_i^I$ using linear interpolation)
    $k \leftarrow k + 1$
    $j \leftarrow j + 1$.
```

Figure 2.1 is an example which exhibits the effectiveness of this algorithm when applied to a NURBS curve with varying curvature. This curve was formed by joining linear, quadratic, and cubic segments into a single NURBS curve. In this example, a total number of $m = 220$ intervals are used in order to limit the maximum deviation to less than the $\epsilon = 10^{-3}$ tolerance. The discretization method clearly concentrates points in areas of high curvature while economizing the number of points used in linear and low curvature regions of the curve. Our method is suitable for curves with first derivatives discontinuous as shown in the figure. It computes a sufficient number of nodes needed in order for the polygonized discretization of the curve to be accurate within a tolerance, and the method requires only the evaluation of the curve function and not its derivatives.

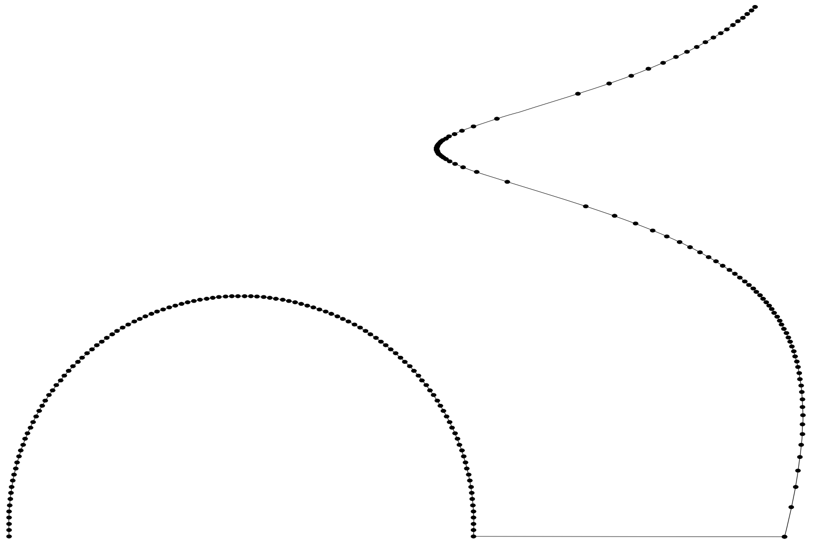


FIG. 2.1. *Adaptive point distribution on a* NURBS *curve.*

3. Hybrid grid point distribution. The adaptive discretization method was targeted at obtaining a high degree of geometric fidelity of the approximation when compared with the original parametric curve. The hybrid approach is targeted at achieving a compromise between geometric fidelity and external application constraints. Depending on the eventual use of the discretization, numerical considerations may favor relaxing conformality of the discretization to achieve enhanced accuracy within the target application.

This section again assumes that the curve is defined as a differentiable parametric curve $\mathbf{x}(u)$. In addition to requiring that $u(\xi)$ is an isomorphism, it is often constructive to constrain the $u(\xi)$ mapping such that the composite map $\mathbf{x}(\xi)$ has the following additional properties to reduce the potential for numerical error: (1) grid points should be closely spaced in the physical domain where large numerical errors are expected, and (2) the angular deviation of grid edges should vary smoothly along the curve.

The methodology of constructing a grid of m intervals on a physical curve begins with the specification of the point distribution along the curve. This is equivalent to specifying the distribution of the curve in the parametric domain $\big\{u(\xi) \bigm| 0 \le \xi \le m\big\}$ which corresponds to the curve $\{\mathbf{x}(u) \mid 0 \le u \le 1\}$ in physical space. The final task is to derive $\{u(\xi) \mid 0 \le \xi \le m\}$ such that $\{\mathbf{x}(u(\xi)) \mid 0 \le \xi \le m\}$ is a "good" parameterization of the boundary curve $\mathbf{x}(u)$.

Again, obtaining $u(\xi)$ is equivalent to finding $\xi(u)$. With the definition $\rho(u) \equiv \frac{d\xi}{du}$,

$$\xi(u) = \int_0^u \rho(w)\, dw.$$

For the hybrid case, a more complex set of physical considerations are typically involved in the construction of the grid point density function, ρ. Possible approaches include the following:

1. Equal arclength spacing where points are separated by equal distances in physical space. For this case, grid point density should be proportional to

the *rate of change of arclength*, or $\rho \propto \|\mathbf{x}'\|$.

2. Curvature weighted arclength spacing, where points are concentrated in areas of large curvature. For this case,

$$\rho \propto \kappa(u)\|\mathbf{x}'\|,$$

where $\kappa(u)$ is the curvature of the curve $\mathbf{x}(u)$ at u.

3. Grid attraction to an *attractor point* $u^\star$ in parametric space (corresponding to a point $\mathbf{x}^\star = \mathbf{x}(u^\star)$ in physical space). A typical case is $u^\star = 0$ or $u^\star = 1$, when one desires to capture a small length scale (such as a Navier–Stokes boundary layer) at one end of the boundary curve. Alternatively, $0 < u^\star < 1$ would focus refinement near a point in the interior of the curve. For these cases, a good choice for ρ is

$$\rho_{u^\star}(u) \propto \frac{1}{\sqrt{\big(k(u-u^\star)\big)^2+1}},$$

where k is a *strength factor* which determines the degree of attraction to $u^\star$.

3.1. Hybrid grid density functions. In practice, the user will likely desire a hybrid grid density function which is a linear combination of several of the above considerations. In this case, multiple density functions ρ_i may be combined, each normalized such that $\int_0^1 \rho_i \, du = \xi(1) - \xi(0) = m$. Given positive constants λ_i such that $\sum \lambda_i = 1$, $\rho = \sum \lambda_i \rho_i$ expresses a grid density function with suitable normalization. This hybrid density function will migrate grid points into regions where any one of the functions ρ_i mandate refinement. Using this concept, it is possible to distribute grid points based on the hybrid criteria of arclength, curvature, and attraction to a set $\{u_i^\star\}$ of distinct points.

This section presents an algorithm for grid point distribution along boundary curves based on a hybrid grid density function. The principle of the algorithm is to construct $\rho(u)$ on a relatively fine grid of points $\tilde{u}_i = \frac{i}{M}$, $0 \le i \le M$, where M is 5–10 times larger than m. The *grid function* $\xi(u) = \int_0^u \rho(w)\, dw$ is then evaluated by integrating ρ on the fine grid with the curve points $u(\xi)$ generated in the parametric space of the curve by inverting the grid function $\xi(u)$.

This algorithm is based on the following quantitative considerations that provide control over point migration:

1. The grid density function for arclength is given by

$$\rho_s(u) = \frac{m\|\mathbf{x}'(u)\|}{\int_0^1 \|\mathbf{x}'(w)\|\, dw}.$$

In this case, $\frac{m}{\int_0^1 \|\mathbf{x}'(w)\|\, dw}$ is the normalization required such that $\int_0^1 \rho_s(u)\, du = m$. If $u = \tilde{u}_i$ and $du = \tilde{u}_i - \tilde{u}_{i-1}$, the approximation

$$\|\mathbf{x}'(\tilde{u}_i)\|\, du \approx \|\mathbf{x}(\tilde{u}_i) - \mathbf{x}(\tilde{u}_{i-1})\|$$

may be employed.

2. The grid density function for curvature weighted arclength is

$$\rho_\kappa(u) = \frac{m\kappa(u)\|\mathbf{x}'(u)\|}{\int_0^1 \kappa(w)\|\mathbf{x}'(w)\|\, dw}.$$

By definition, $\kappa(u) = \frac{d\theta}{ds}$, where $d\theta$ is the angular change in the direction of the tangent of the curve during a small traversal of arclength ds along the curve. Thus,

$$\kappa(u)\|\mathbf{x}'(u)\| = \frac{d\theta}{ds}\frac{ds}{du} = \frac{d\theta}{du}.$$

If $u = \tilde{u}_i$, a simplifying approximation may be used,

$$\begin{aligned}\kappa(\tilde{u}_i)\|\mathbf{x}'(\tilde{u}_i)\|\ du &\approx \theta_i - \theta_{i-1}\\ &= \|\mathbf{t}_i - \mathbf{t}_{i-1}\|,\end{aligned}$$

where $\mathbf{t}_i \equiv \frac{\mathbf{x}'(\tilde{u}_i)}{\|\mathbf{x}'(\tilde{u}_i)\|}$ is the unit tangent vector to the physical curve at $\tilde{u}_i$. If the total integrated curvature $\int_0^1 \kappa(u)\|\mathbf{x}'(u)\|\ du \approx \sum_{i=1}^{M} \|\mathbf{t}_i - \mathbf{t}_{i-1}\|$ is less than some minimal angular tolerance (say $\epsilon_\kappa = .01$ radian), then the curvature weighted arclength is replaced with a simple arclength expression as a criterion for grid point distribution. This substitution avoids distributing points based on a quantity which is numerically small, which could lead to a nonsmooth distribution.

3. The grid density function for attraction (with strength k) to a point $u^\star$ is given by

$$\rho_{u^\star}(u) = m\,\frac{1}{\sqrt{\big(k(u-u^\star)\big)^2+1}}\bigg/\int_0^1 \frac{1}{\big(k(w-u^\star)\big)^2+1}\,dw, \tag{3.1}$$

$$\int_o^u \rho_{u^\star}(w)\ dw = m\,\frac{\operatorname{arcsinh}\big(k(u-u^\star)\big)+\operatorname{arcsinh}(ku^\star)}{\operatorname{arcsinh}\big(k(1-u^\star)\big)+\operatorname{arcsinh}(ku^\star)}.$$

If $u^\star = 0$,

$$\int_0^u \rho_0(w)\ dw = m\,\frac{\operatorname{arcsinh}(ku)}{\operatorname{arcsinh}(k)}.$$

This leads to a grid distribution of the form

$$u(\xi) = \frac{\sinh(\frac{\alpha\xi}{m})}{\sinh\alpha}.$$

It has been noted that the smoothness of this distribution in the vicinity of $u^\star = 0$ results in a smaller truncation error (in finite difference expressions) than "exponential" functions that often approach the attractor point in a more severe fashion (Thompson, Warsi, and Mastin [13]).

Algorithm 3.1. *Hybrid grid point distribution.*

Assume a physical curve $\mathbf{x}(u)$, $0 \le u \le 1$. Given weights λ_s, λ_κ, points $\{u_i^\star \mid 0 \le u_i^\star \le 1,\ 1 \le i \le p\}$, weights $\{\lambda_i \mid 1 \le i \le p\}$, and strengths $\{k_i | k_i \ge 0,\ 1 \le i \le p\}$ where $\lambda_s + \lambda_\kappa + \sum_{i=1}^{p} \lambda_i = 1$, create a distribution of $m+1$ points $u_0, u_1, \ldots, u_m$ that are simultaneously attracted to each of the points in $\{u_i^\star\}$, placed in regions of high curvature, and placed to avoid large gaps in arclength. The user must also specify a parametric grid resolution $M \ge m$ and a minimum integrated curvature tolerance ϵ_κ. (The authors suggest $M = 5m$ and $\epsilon_\kappa = 10^{-3}$.)

1. Initialize the grid function ξ to zero.

```
Do $i = 1, \ldots, M$
  $\xi_i \leftarrow 0$
```

2. Compute arclengths. Rescale such that the maximum scaled arclength is m. Add to ξ, weighted by λ_s.

```
$s_0 \leftarrow 0$
Do $i = 1, \ldots, M$
  $s_i \leftarrow s_{i-1} + \|\mathbf{x}(\frac{i}{M}) - \mathbf{x}(\frac{i-1}{M})\|$
Do $i = 1, \ldots, M$
  $s_i \leftarrow m \frac{s_i}{s_M}$
  $\xi_i \leftarrow \xi_i + \lambda_s s_i$
```

3. Compute curvature weighted arclengths on fine grid. Check if curve has a nontrivial amount of curvature. If so, normalize to m and add into ξ, weighted by λ_κ. Otherwise, use arclength instead.

```
Do $i = 0, \ldots, M$
  $\mathbf{t}(i) \leftarrow \mathbf{x}'(\frac{i}{M}) / \|\mathbf{x}'(\frac{i}{M})\|$
$\theta_0 \leftarrow 0$
Do $i = 1, \ldots, M$
  $\theta_i \leftarrow \theta_{i-1} + \|\mathbf{t}(i) - \mathbf{t}(i-1)\|$
If ($\theta_M \geq \epsilon_\kappa$), then
  Do $i = 1, \ldots, M$
    $\theta_i \leftarrow m \, \frac{\theta_i}{\theta_M}$
    $\xi_i \leftarrow \xi_i + \lambda_\kappa \theta_i$
Else
  Do $i = 1, \ldots, M$
    $\xi_i \leftarrow \xi_i + \lambda_\kappa s_i$
```

4. Add in contributions to grid function due to attractor points.

```
Do $j = 1, \ldots, p$
  Do $i = 1, \ldots, M$
    $\xi_i \leftarrow \xi_i + \lambda_j m \, \frac{\operatorname{arcsinh}\left(k_j(\frac{i}{M} - u_j^\star)\right) + \operatorname{arcsinh}(k_j u_j^\star)}{\operatorname{arcsinh}\left(k_j(1 - u_j^\star)\right) + \operatorname{arcsinh}(k_j u_j^\star)}$
```

5. Obtain point distribution by inverting grid function.

```
$\xi_M \leftarrow m$     *(Force final grid function value to be* exactly *$m$)*
$u_0 \leftarrow 0$
$j \leftarrow 1$
Do $i = 1, \ldots, M$
  Do while ($j \leq \xi_i$)
    $u_j \leftarrow \frac{i}{M} - \frac{1}{M} \frac{\xi_i - j}{\xi_i - \xi_{i-1}}$     *(Obtain $\frac{i-1}{M} < u_j \leq \frac{i}{M}$ using linear interpolation)*
    $j \leftarrow j + 1$.
```

3.2. Determination of weights $\boldsymbol{\lambda_s, \lambda_\kappa, \lambda_i}$ and strengths $\boldsymbol{k_i}$. When using the boundary point distribution algorithm, one must choose weights $\lambda_s, \lambda_\kappa, \lambda_i$ and strengths k_i. As a rough estimate, it is sufficient to select these weights to be equal with a unity summation constraint. For example, if one desires a distribution on *arclength* and *two attractor points*, set $\lambda_s = \lambda_1 = \lambda_2 = \frac{1}{3}$. (In this case, $\lambda_\kappa = 0$.)

When expressing the strengths k_i on the attractor points $u_i^\star$, one must further consider the degree of concentration required by the particular application. Given the case of a single attractor point $u^\star = u_1^\star$, (3.1) suggests that

$$\rho(u) = m \, \frac{k \Big/ \sqrt{\big(k(u-u^\star)\big)^2 + 1}}{\operatorname{arcsinh}\big(k(1-u^\star)\big) + \operatorname{arcsinh}(ku^\star)}$$

or

$$\begin{aligned} \rho(u^\star) &= m \, \frac{k}{\operatorname{arcsinh}\big(k(1-u^\star)\big) + \operatorname{arcsinh}(ku^\star)} \\ &\geq m \, \frac{k}{2 \operatorname{arcsinh}(\frac{k}{2})}. \end{aligned} \tag{3.2}$$

For example, choosing $k = 100$ would provide $\rho(u^\star) \geq 10m$, which results in grid points packed in the neighborhood of $u^\star$ with a density in excess of 10 times of the average grid density $\rho_{\text{ave}} = m$. As a second example, suppose one desires to construct a grid with a specified value of $\frac{\rho(u^\star)}{m}$, or, alternatively, specifies an excess grid density at the attractor $u^\star$. For this case, a heuristic expression

$$k = 15 \, \frac{\rho(u^\star)}{m} \tag{3.3}$$

may be used, adjusted as necessary to provide the desired result. One could alternatively solve the nonlinear equation (3.2) for k exactly. The presence of other criteria (such as arclength, curvature, or other attractor points) in this expression may support the desire to predict a solution using (3.3) and correcting this result, iteratively, by the adjustment of k as necessary.

If one desires a certain grid spacing Δx in the region near $\mathbf{x}^\star = \mathbf{x}(u^\star)$, consider that

$$\Delta x = \|\mathbf{x}_\xi\| \Bigg|_{u=u^\star} = \|\mathbf{x}' \cdot u_\xi\| \Bigg|_{u=u^\star} = \frac{\|\mathbf{x}'(u^\star)\|}{\rho(u^\star)}.$$

Equation (3.3) yields

$$k = 15 \, \frac{\|\mathbf{x}'(u^\star)\|}{m\Delta x},$$

an estimate for the strength k required to obtain a grid with the desired spacing Δx near the attractor $\mathbf{x}^\star = \mathbf{x}(u^\star)$ on the physical curve $\mathbf{x}(u(\xi))$.

Figure 3.1 illustrates an example of the hybrid method applied to a parametric curve. This illustration was produced with Algorithm 3.1, where $\lambda_s = \lambda_\kappa = \frac{1}{2}$. In this case, the points are distributed equally according to both arclength and curvature weighting. The effect of curvature distribution is clearly seen; the grid points are clustered in areas of high curvature. The fact that arclength is additionally considered is evidenced by the nonzero density of grid points in areas where curvature is small or absent.

Figure 3.2 illustrates an example where the influence of two attractor points on the curve is considered in addition to arclength and curvature weighting. These attractor points are located at both endpoints of the curve. The parameters used for

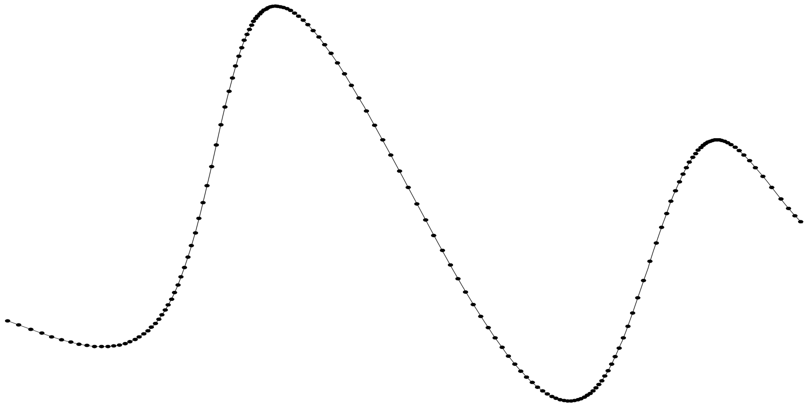


FIG. 3.1. *Point distribution based on arclength and curvature.*

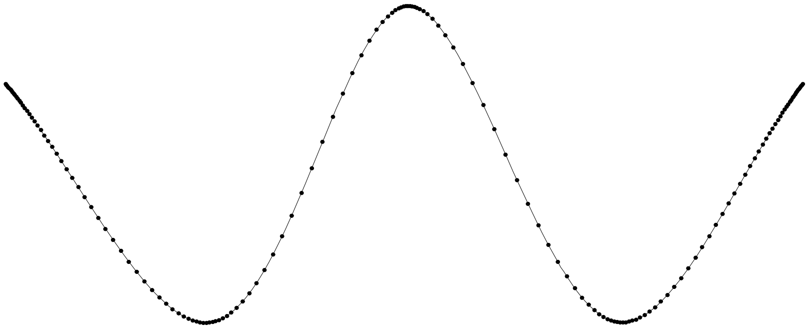


FIG. 3.2. *Curve grid with point distribution based on arclength, curvature, and two attractor points.*

the curve point distribution in this result are $\lambda_s = \lambda_\kappa = \lambda_1 = \lambda_2 = \frac{1}{4}$, $k_1 = k_2 = 120$, and $u_1^\star = 0$ and $u_2^\star = 1$. Examining (3.3), $k_i = 120$ implies that the grid should be packed in the neighborhood of the attractors $u_i^\star$ at a density approximately eight times higher than the average grid point density. Close examination of the figure near the endpoints reveals that this clustering was achieved.

4. Parametric curve grid smoothing.

This section assumes that the topology of the curve grid (i.e., the number of points on the curve) is fixed and presents a technique useful for smoothing these curve grids.

A popular approach to curve grid smoothing is based on the use of the mapping from parametric space to the physical domain; the physical grid is smoothed via operations applied to the parametric representation. This approach assumes that a mapping from parametric space exists to allow smoothing of the curve grid in the parametric domain and to preserve the physical shape of the curve.

This approach often involves solving an elliptic partial differential equation to slide the grid points along the curve, leading to the satisfaction of some equidistribution principle. An initial point distribution must be present to provide a discretization for the elliptic system and an "initial guess" for its iterative solution. This distribution is assumed to be a mapping $\mathbf{x}(\xi)$ from computational space $[0, m]$ to the physical curve. As in previous sections, here *grid points* are the points at $\xi = i$ with $0 \le i \le m$. The initial mapping $\mathbf{x}(\xi)$ is often obtained using an auxiliary uniform point distribution in the parametric domain, $\mathbf{x}_i = \mathbf{x}(u_i)$, where $u_i = \frac{i}{m}$. If the curve geometry is poorly parameterized, this uniform distribution in parametric space will correspond to an unacceptable point distribution in physical space, and the curve point distribution will have to be smoothed.

Our elliptic smoothing method uses the one-dimensional equidistribution principle,

$$ws_\xi = c \quad \text{(constant)},$$

where $w \equiv w(s)$ is the spacing weight function and is taken as a function of the arc length s of the curve. The grid point distribution resulting from this equidistribution represents the solution of the following elliptic system and, provided that w is smooth, results in a smooth arclength distribution regardless of the parameterization of the curve. The above differential equation is equivalent to

$$s_{\xi\xi} + \phi(s)s_\xi = 0$$

with boundary conditions

$$s_\xi = \|\Delta\mathbf{x}\| \quad \text{for} \quad \xi = 0, m.$$

The one-dimensional spacing function $\phi(s) = \frac{w_\xi}{w}$ is taken to be a function of the arclength s of the curve. (However, it must be noted that $\phi = \phi(s)$ depends on the choice of $u(\xi)$ and that $\phi(s)$ will have to be recomputed whenever $u(\xi)$ is changed in the iterative smoothing process.) In the absence of the spacing function, i.e., $\phi = 0$, the elliptic equation tends to produce the smoothest possible uniform curve grid. The boundary conditions allow for the specified grid spacings at either end of the curve.

The change in the arclength of the curve is given by $ds = du\|\mathbf{x}_u\|$. Therefore,

$$s_{\xi\xi} = \frac{\mathbf{x}_u \cdot \mathbf{x}_{uu}u_\xi^2 + \mathbf{x}_u \cdot \mathbf{x}_u u_{\xi\xi}}{\|\mathbf{x}_u\|},$$

where $s_\xi = s_u u_\xi = \|\mathbf{x}_u\| u_\xi$. Thus, u is a solution of the following quasi-linear elliptic equation:

$$u_{\xi\xi} + \phi u_\xi = -u_\xi^2 \frac{\mathbf{x}_u \cdot \mathbf{x}_{uu}}{\mathbf{x}_u \cdot \mathbf{x}_u}.$$

The boundary conditions are represented by

$$u_\xi = \frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}_u\|} \quad \text{for} \quad \xi = 0, m.$$

To implement the elliptic equation on a given parametric boundary curve, u_0 and u_m are obtained from the initial curve grid. The first two points interior of the curve endpoints, u_1 and u_{m-1}, are computed from the boundary conditions with user-specified spacings $\|\Delta\mathbf{x}^l\|$ and $\|\Delta\mathbf{x}^r\|$ at both ends of the curve,

$$u_1 = u_0 + \frac{\|\Delta\mathbf{x}^l\|}{\|\mathbf{x}_{u_0}\|},$$
$$u_{m-1} = u_m - \frac{\|\Delta\mathbf{x}^r\|}{\|\mathbf{x}_{u_m}\|}.$$

The user-specified point density on the remainder of the curve is provided by the parametric values u_i for $i = 2, \ldots, m-2$. Linear interpolation,

$$u_i = t_i u_{m-1} + (1-t_i)u_1, \quad t_i = \frac{i-1}{m-2} \quad \text{for} \quad i = 2, \ldots, m-2,$$

is used to provide the initial locations of these points. Redistribution of these interpolated points is imposed by adjusting the spacing function ϕ near the boundary, while holding the coordinates of the boundary points constant. To accomplish this, the spacing function is evaluated at $i = 1$ and $i = m-1$ using

$$\phi_i = -\frac{u_{\xi\xi}}{u_\xi}\bigg|_i - u_\xi \frac{\mathbf{x}_u \cdot \mathbf{x}_{uu}}{\mathbf{x}_u \cdot \mathbf{x}_u}\bigg|_i \quad \text{for} \quad i = 1, m-1.$$

The function ϕ is distributed to the remaining points on the curve, again using linear interpolation,

$$\phi_i = t_i \phi_{m-1} + (1-t_i)\phi_1, \quad t_i = \frac{i-1}{m-2} \quad \text{for} \quad i = 2, \ldots, m-2.$$

A standard discretization technique that may be used is central differencing in the ξ domain. In this approach, the difference operator is applied to the interior nodes on the curve to solve for $\mathbf{x}_i = (x_i, y_i, z_i)$ in an iterative manner. Let $(u_\xi)_i$ denote the central difference $\frac{1}{2}(u_{i+1} - u_{i-1})$, and $(u_{\xi\xi})_i \approx u_{i+1} - 2u_i + u_{i-1}$. To solve for u_i, the old parameter value is used to solve for the new u_i to provide $\mathbf{x}_i = \mathbf{x}(u_i)$ along the curve,

$$\begin{aligned} u_i = {} & \frac{1}{2}(u_{i+1} + u_{i-1}) \\ & + \frac{1}{4}(u_{i+1} - u_{i-1})\phi_i \\ & + \frac{1}{8}(u_{i+1} - u_{i-1})^2 \left(\frac{\mathbf{x}_u \cdot \mathbf{x}_{uu}}{\mathbf{x}_u \cdot \mathbf{x}_u} \right)_i \quad \text{for} \quad i = 2, \ldots, m-2. \end{aligned}$$

This difference equation is evaluated for u_i each iterative cycle.

As previously mentioned, the spacing function ϕ_i must be refreshed every iteration prior to the computation of u_i. However, the curve boundary points u_0, u_1, u_{m-1}, and u_m need only to be initialized at the beginning of the relaxation cycle for each curve. This relaxation procedure is applied until convergence is achieved.

The quantities in the difference equation involve two types of approximations. The derivative of the parametric variables with respect to the computational variables u_ξ and $u_{\xi\xi}$ are approximated using a finite difference expression, whereas the derivative terms of the physical variables with respect to the parametric variables $\mathbf{x}_u$ and $\mathbf{x}_{uu}$ are computed analytically from the curve definition $\mathbf{x}(u)$. This development leads directly to the equidistribution algorithm for curve grid smoothing.

Algorithm 4.1. *Parametric curve grid smoothing.*

Assume a physical curve $\mathbf{x}(u)$, $0 \le u \le 1$, with a set of $m+1$ discrete points on $\mathbf{x}(u)$ defined by monotonically increasing parametric values $\{u_i | \; 0 \le i \le m\}$. Assume user specified spacing $\|\Delta\mathbf{x}^l\|$, $\|\Delta\mathbf{x}^r\|$, stopping criteria (number of "sweeps"), and a relaxation parameter $\omega \in (0, 1]$.

```
1. Initialize curve grid distribution.
   $u_0 \leftarrow 0$
```

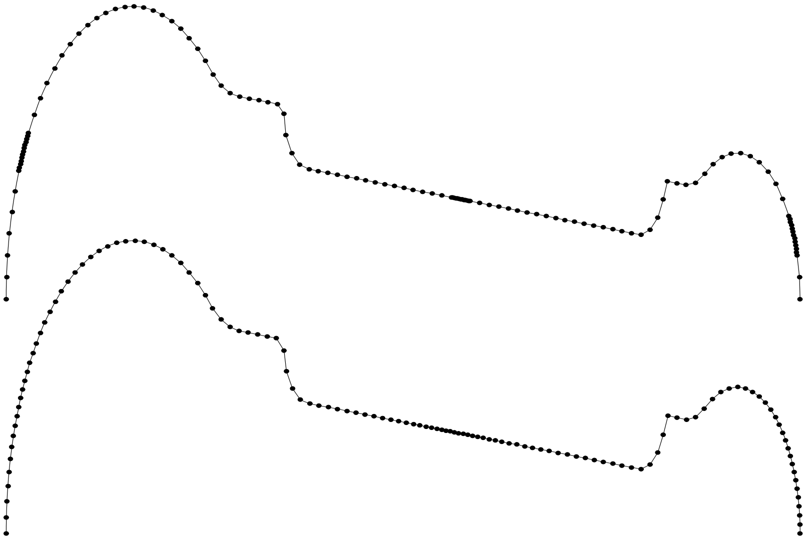


Fig. 4.1. *Initial curve grid with poor point distribution (top); elliptic curve grid with smoothly varying point distribution (bottom).*

```
   $u_m \leftarrow 1$
   $u_1 \leftarrow u_0 + \frac{\|\Delta\mathbf{x}^l\|}{\|\mathbf{x}_{u_0}\|}$
   $u_{m-1} \leftarrow u_m - \frac{\|\Delta\mathbf{x}^r\|}{\|\mathbf{x}_{u_m}\|}$
   Do $i = 2, \ldots, m-2$
      $t_i \leftarrow \frac{i-1}{m-2}$
      $u_i \leftarrow t_i u_{m-1} + (1 - t_i) u_1$
2. Repeat (sweep) until “done”
   $\phi_1 \leftarrow -2\frac{u_2 - 2u_1 + u_0}{u_2 - u_0} - \frac{u_2 - u_0}{2} \left.\frac{\mathbf{x}_u \cdot \mathbf{x}_{uu}}{\mathbf{x}_u \cdot \mathbf{x}_u}\right|_{u_1}$
   $\phi_{m-1} \leftarrow -2\frac{u_m - 2u_{m-1} + u_{m-2}}{u_m - u_{m-2}} - \frac{u_m - u_{m-2}}{2} \left.\frac{\mathbf{x}_u \cdot \mathbf{x}_{uu}}{\mathbf{x}_u \cdot \mathbf{x}_u}\right|_{u_{m-1}}$
   Do $i = 2, \ldots, m-2$
      $t_i \leftarrow \frac{i-1}{m-2}$
      $\phi_i \leftarrow t_i \phi_{m-1} + (1 - t_i)\phi_1$
   Do $i = 2, \ldots, m-2$
      $u_s \leftarrow \frac{1}{2}(u_{i+1} + u_{i-1}) + \frac{1}{4}(u_{i+1} - u_{i-1})\phi_i + \frac{1}{8}(u_{i+1} - u_{i-1})^2 \left.\frac{\mathbf{x}_u \cdot \mathbf{x}_{uu}}{\mathbf{x}_u \cdot \mathbf{x}_u}\right|_{u_i}$
      $u_i \leftarrow (1 - \omega) u_i + \omega u_s.$
```

Figure 4.1 illustrates the effect of the elliptic grid smoothing algorithm applied to a curve geometry. The initial grid distribution clearly exhibits a nonuniform distribution of grid points in several regions. This grid defect could conceivably lead to unacceptable artifacts in a computation involving the grid. The elliptically smoothed grid exhibits a smooth point distribution after a few iterations. In this example, the smoothing was performed with specified physical spacings $\|\Delta\mathbf{x}^l\| = 0.08$ and $\|\Delta\mathbf{x}^r\| = 0.04$. We observe that the grid point spacings are enforced near the ends of the curve, while the spacings in the middle of the curve are smoothly varying. Naturally, the spacings in the interior of the curve are determined more by the fixed number of points on the curve than by the endpoint spacing boundary conditions.

5. Nonparametric curve grid smoothing. The last smoothing algorithm considered in this paper is a method that seeks to simultaneously equidistribute points along a geometric curve while reformulating this curve based on a set of physical constraints. In fact, the development here generalizes [14] by introducing for nonplanar curves the notion of smoothing while locally conserving area in the osculating plane and simultaneously conserving torsion.

Curves in the space obtained from field simulations frequently display "nonsmooth" artifacts and may be generally unsuitable as input for subsequent simulations. However, one desires that any smoothing operation be physically relevant; i.e., one may desire that the smoothing process be area- or mass-conservative with respect to an enclosed region.

In contrast with the previous smoothing approach, this method does not require a mapping from parametric space to the physical curve and vice versa. This method was developed in an attempt to preserve the area (or mass) that the curve grid encloses while sacrificing that the exact *shape* of the curve be preserved. In certain situations, exact preservation of the curve shape may be undesirable (e.g., in the case of "stair-step" curves).

This section develops a nonparametric area-conserving smoothing approach of planar curve grids (with generalization to space curve grids). This approach is designed to rapidly deform a "stair-stepped" closed curve into a smoothed curve which encloses the same area as the original to machine round-off error. To accomplish this, the area-conserving approach allows small deformations in the *shape* of the curve geometry. However, the degree of curve deformation can be limited by controlling the number of smoothing iterations performed. Additionally, the notion of area conservation is generalized to allow for the extension of the scheme to nonclosed curve grids in the plane and to space curves.

Consider a non-self-intersecting piecewise linear curve $\Gamma = (\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{n-1}, \mathbf{x}_n)$ consisting of n line segments in $\mathbb{R}^2$ or $\mathbb{R}^3$. The goal of this section is to develop a smoothing operation on this curve that can be performed locally at each $\mathbf{x}_i$ and that involves slightly altering the position of $\mathbf{x}_i$ based on nearby or adjacent data points ($\{\mathbf{x}_{i-m}, \mathbf{x}_{i-m+1}, \ldots, \mathbf{x}_{i+m}\}$ with m small). More generally, the smoothing operation may depend on points in $\{\mathbf{x}_{i-m}, \mathbf{x}_{i-m+1}, \ldots, \mathbf{x}_{i+m}\}$ and involve moving one or more points in this neighborhood. The smoothing operation should be constrained such that the area of each triangle $(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1})$ is not modified. The area constraint allows the curve to be smoothed without shrinkage. Given the definition of a *sweep* as one iteration of the local smoothing operation applied on a subset of the points that describe the curve in some sequential order, one desires that only a small number of sweeps through the curve data be required to smooth the overall appearance of the curve.

For a curve in $\mathbb{R}^2$, area conservation is defined as conserving the *signed* area of Γ (given a counter clockwise orientation). A simplistic approach at a smoothing operation providing conservation is depicted in Figure 5.1. For this example, consider the three points $\mathbf{x}_0, \mathbf{x}_1$, and $\mathbf{x}_2$ along Γ. By moving the central point $\mathbf{x}_1$ *parallel* to the line segment $\overline{\mathbf{x}_0\mathbf{x}_2}$, one is assured conservation of area. Further, by moving $\mathbf{x}_1$ such that the projection of $\mathbf{x}_1$ onto $\overline{\mathbf{x}_0\mathbf{x}_2}$ occurs midway between $\mathbf{x}_0$ and $\mathbf{x}_2$, one achieves equal spacing of the segments $\overline{\mathbf{x}_0\mathbf{x}_1}$ and $\overline{\mathbf{x}_1\mathbf{x}_2}$ when projected onto the segment $\overline{\mathbf{x}_0\mathbf{x}_2}$.

Formally, an algorithm based on this one-point smoothing operation is as follows. Let $\mathbf{a} = (\mathbf{x}_2 - \mathbf{x}_0) \times (\mathbf{x}_1 - \mathbf{x}_0)$ be the area vector of triangle $(\mathbf{x}_0, \mathbf{x}_2, \mathbf{x}_1)$. (If Γ is planar, we consider it embedded in the x-y plane of $\mathbb{R}^3$.) The targeted area of the triangle is $\frac{1}{2}\|\mathbf{x}_2 - \mathbf{x}_0\|h$, where h is the height of $\mathbf{x}_1$ above the baseline segment $\overline{\mathbf{x}_0\mathbf{x}_2}$. Then,

$$h = \frac{\|\mathbf{a}\|}{\|\mathbf{x}_2 - \mathbf{x}_0\|}.$$

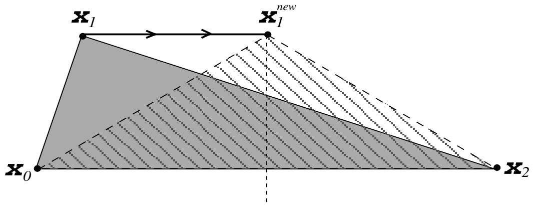


FIG. 5.1. *One-point smoothing operation: Movement of* $\mathbf{x}_1$ *parallel to* $\overline{\mathbf{x}_0\mathbf{x}_2}$ *assures conservation of area under curve* $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2)$.

Let $\hat{\mathbf{n}} = \frac{\mathbf{a}\times(\mathbf{x}_2-\mathbf{x}_0)}{\|\mathbf{a}\times(\mathbf{x}_2-\mathbf{x}_0)\|}$ be the unit normal to the baseline $\overline{\mathbf{x}_0\mathbf{x}_2}$. The smoothing operation then involves repositioning $\mathbf{x}_1$ from its original position to

$$\mathbf{x}_1^{\mathrm{new}} = \frac{1}{2}(\mathbf{x}_0 + \mathbf{x}_2) + h\hat{\mathbf{n}}.$$

If Γ is planar, this action will exactly conserve the area under the curve; if Γ is not planar, this action will at least preserve the local area under the curve in the plane containing the triangle $\{\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2\}$, which corresponds to the osculating plane of a parametric curve. Sweeping through the nodes in sequential order, one obtains the following algorithm.

ALGORITHM 5.1. *Nonparametric curve grid smoothing using node relaxation.*

```
Repeat (sweep) until "done"
  Do $i = 1, \ldots, n-1$
    [Perform smoothing operation on neighborhood $\{\mathbf{x}_i, \mathbf{x}_{i+1}, \mathbf{x}_{i+2}\}$
      (i.e., relax node $\mathbf{x}_{i+1}$)]
    $\mathbf{x}^{\mathrm{old}} \leftarrow \mathbf{x}_{i+1}$
    [Smooth in tangential direction.]
    $\mathbf{x}_{i+1} \leftarrow \frac{1}{2}(\mathbf{x}_i + \mathbf{x}_{i+2})$
    [In nondegenerate case, conserve area.]
    $\mathbf{a} \leftarrow (\mathbf{x}_{i+2} - \mathbf{x}_i) \times (\mathbf{x}^{\mathrm{old}} - \mathbf{x}_i)$
    If ($\|\mathbf{a} \times (\mathbf{x}_{i+2} - \mathbf{x}_i)\| >$ "a tiny number"), then
      $\hat{\mathbf{n}} \leftarrow \frac{\mathbf{a}\times(\mathbf{x}_{i+2}-\mathbf{x}_i)}{\|\mathbf{a}\times(\mathbf{x}_{i+2}-\mathbf{x}_i)\|}$
      $h \leftarrow \frac{\|\mathbf{a}\|}{\|(\mathbf{x}_{i+2}-\mathbf{x}_i)\|}$
      $\mathbf{x}_{i+1} \leftarrow \mathbf{x}_{i+1} + h\hat{\mathbf{n}}$.
```

The simplistic basis of Algorithm 5.1 leads to a serious deficiency. Given the definition of the direction $\overrightarrow{\mathbf{x}_0\mathbf{x}_2}$ shown in Figure 5.1 as "tangential" to Γ and the direction orthogonal to $\overrightarrow{\mathbf{x}_0\mathbf{x}_2}$ as the "normal" direction, the one-point smoothing operation smooths only in the tangential direction. Any smoothing in the normal direction is forbidden by the conservation of area requirement. Because of the absence of normal smoothing, some star-shaped regions will not be addressed by the smoothing operation. It is therefore necessary to design a local smoothing operation that includes normal smoothing.

Consider four sequential points $\mathbf{x}_0$, $\mathbf{x}_1$, $\mathbf{x}_2$, and $\mathbf{x}_3$ along Γ. Again, $\overrightarrow{\mathbf{x}_0\mathbf{x}_3}$ is the direction tangential to the curve, and the direction orthogonal to $\overrightarrow{\mathbf{x}_0\mathbf{x}_3}$ is defined as the normal to the curve. If one simultaneously solves for the positions $\mathbf{x}_1$ and $\mathbf{x}_2$ subject to the constraint of area conservation, normal smoothing is possible. This is feasible, as conservation of area represents a single constraint in the normal direction, but there are two degrees of freedom available (the normal components of $\mathbf{x}_1$ and $\mathbf{x}_2$).

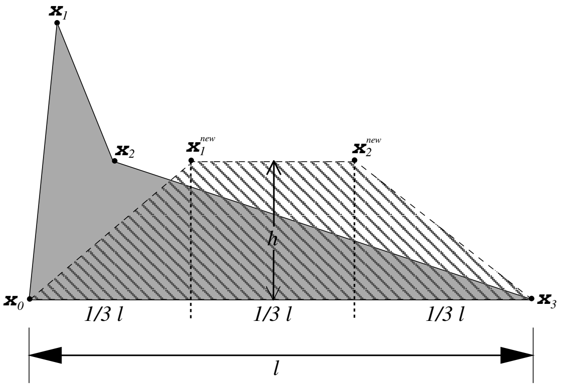


FIG. 5.2. *Edge smoothing operation.* $\overline{\mathbf{x}_1\mathbf{x}_2}$ *moved to be parallel to* $\overline{\mathbf{x}_0\mathbf{x}_3}$ *with projected endpoints at* $\frac{1}{3}l$ *and* $\frac{2}{3}l$*. Choosing* $h = \frac{3}{4}\frac{a}{l}$ *conserves area of the quad* $(\mathbf{x}_0, \mathbf{x}_3, \mathbf{x}_2, \mathbf{x}_1)$*.*

Indeed, development of this idea leads to a new smoothing algorithm. Consider the placement of $\mathbf{x}_1$ and $\mathbf{x}_2$ such that the projection of $\mathbf{x}_1$ onto $\overline{\mathbf{x}_0\mathbf{x}_3}$ is one-third of the way between $\mathbf{x}_0$ and $\mathbf{x}_3$, and the projection of $\mathbf{x}_2$ is two-thirds of the way between $\mathbf{x}_0$ and $\mathbf{x}_3$. Furthermore, the distances of $\mathbf{x}_1$ and $\mathbf{x}_2$ away from $\overline{\mathbf{x}_0\mathbf{x}_3}$ are chosen to be equal with the magnitude of this distance (h in Figure 5.2) constrained to conserve area. In this approach, smoothing occurs in the normal direction as well as in the tangential direction.

The calculation of h is straightforward. To conserve the local area, the vector area of the quadrilateral $(\mathbf{x}_0, \mathbf{x}_3, \mathbf{x}_2, \mathbf{x}_1)$ is computed as $\mathbf{a} = (\mathbf{x}_3 - \mathbf{x}_0) \times (\mathbf{x}_2 - \mathbf{x}_0) + (\mathbf{x}_2 - \mathbf{x}_0) \times (\mathbf{x}_1 - \mathbf{x}_0)$. Repositioning the points $\mathbf{x}_1, \mathbf{x}_2$ in the plane orthogonal to $\mathbf{a}$ such that their projections onto $\overline{\mathbf{x}_0\mathbf{x}_3}$ are equally spaced implies that the area of the quadrilateral $(\mathbf{x}_0, \mathbf{x}_3, \mathbf{x}_2, \mathbf{x}_1)$ will be $\frac{2}{3}hl$, where l is the length of $\overline{\mathbf{x}_0\mathbf{x}_3}$. This requires that

$$h = \frac{3}{4}\frac{a}{l},$$

where $a = \|\mathbf{a}\|$ and $l = \|\mathbf{x}_3 - \mathbf{x}_0\|$.

If Γ is planar, this action will exactly conserve the area under the curve. However, if Γ is not planar, there is a problem in that repositioning of $\mathbf{x}_1, \mathbf{x}_2$ in the plane orthogonal to $\mathbf{a}$ does not conserve the torsion of the curve. (Thus without modification, this smoothing operation would not leave invariant an optimal grid on a helix formed by spacing points equally in arclength.) To conserve torsion, we compute the component of the original edge vector $\overrightarrow{\mathbf{x}_1\mathbf{x}_2}$ that is parallel to $\mathbf{a}$, the normal of the "average" osculating plane. Then we further reposition $\mathbf{x}_1, \mathbf{x}_2$ so that $\mathbf{x}_1, \mathbf{x}_2$ lie at equal distances under and over the averaged osculating plane and maintain their previous separations normal to the osculating plane.

The above node repositioning can be interpreted as being a smoothing operation acting on the *edge* $\overline{\mathbf{x}_1\mathbf{x}_2}$. Thus, to perform a smoothing sweep through Γ, one applies the operation on all the edges of Γ in some order. For example, *sequential order* would perform the smoothing operation on the edge $\overline{\mathbf{x}_1\mathbf{x}_2}$, followed by the edge $\overline{\mathbf{x}_2\mathbf{x}_3}$, and continuing until the last edge is encountered.

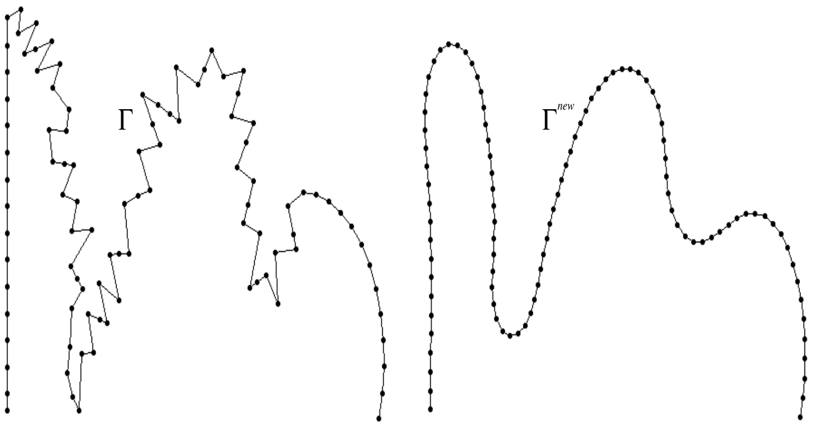


FIG. 5.3. *Before and after smoothing of an open curve using Algorithm* 5.2 *with* 10 *sweeps.*

ALGORITHM 5.2. *Nonparametric curve grid smoothing using edge relaxation.*

```
Repeat (sweep) until "done"
  Do $i = 1, \dots, n-1$
    [Perform smoothing operation on neighborhood $\{\mathbf{x}_i, \mathbf{x}_{i+1}, \mathbf{x}_{i+2}, \mathbf{x}_{i+3}\}$
      (i.e., relax edge $\overline{\mathbf{x}_{i+1}, \mathbf{x}_{i+2}}$)]
    $\mathbf{x}_1^{\text{old}} \leftarrow \mathbf{x}_{i+1}$
    $\mathbf{x}_2^{\text{old}} \leftarrow \mathbf{x}_{i+2}$
    [Smooth edge in tangential direction.]
    $\mathbf{x}_{i+1} \leftarrow \frac{2}{3}\mathbf{x}_i + \frac{1}{3}\mathbf{x}_{i+3}$
    $\mathbf{x}_{i+2} \leftarrow \frac{1}{3}\mathbf{x}_i + \frac{2}{3}\mathbf{x}_{i+3}$
    [In nondegenerate case, conserve area in osculating plane.]
    $\mathbf{a} \leftarrow (\mathbf{x}_{i+3} - \mathbf{x}_i) \times (\mathbf{x}_2^{\text{old}} - \mathbf{x}_i) + (\mathbf{x}_2^{\text{old}} - \mathbf{x}_i) \times (\mathbf{x}_1^{\text{old}} - \mathbf{x}_i)$
    If ($\|\mathbf{a} \times (\mathbf{x}_{i+3} - \mathbf{x}_i)\| >$ "a tiny number"), then
      $\hat{\mathbf{n}} \leftarrow \frac{\mathbf{a} \times (\mathbf{x}_{i+3} - \mathbf{x}_i)}{\|\mathbf{a} \times (\mathbf{x}_{i+3} - \mathbf{x}_i)\|}$
      $h \leftarrow \frac{3}{4} \frac{\|\mathbf{a}\|}{\|\mathbf{x}_{i+3} - \mathbf{x}_i\|}$
      $\mathbf{x}_{i+1} \leftarrow \mathbf{x}_{i+1} + h\hat{\mathbf{n}}$
      $\mathbf{x}_{i+2} \leftarrow \mathbf{x}_{i+2} + h\hat{\mathbf{n}}$
    [In nondegenerate case, conserve torsion.]
    If ($\|\mathbf{a}\| >$ "a tiny number"), then
      $\hat{\mathbf{n}}_\perp \leftarrow \frac{\mathbf{a}}{\|\mathbf{a}\|}$
      $d_\perp \leftarrow (\mathbf{x}_2^{\text{old}} - \mathbf{x}_1^{\text{old}}) \cdot \hat{\mathbf{n}}_\perp$
      $\mathbf{x}_{i+1} \leftarrow \mathbf{x}_{i+1} - \frac{d_\perp}{2}\hat{\mathbf{n}}_\perp$
      $\mathbf{x}_{i+2} \leftarrow \mathbf{x}_{i+2} + \frac{d_\perp}{2}\hat{\mathbf{n}}_\perp$.
```

Figure 5.3 shows the results of Algorithm 5.2, applying 10 sweeps on a plane open curve Γ, holding the first and last points fixed. If Γ were closed by the addition of a segment between the first and last points, the area enclosed by Γ would be conserved to round-off error. Further iterations will continue to deform the curve to increase the convexity of the enclosed region.

6. Summary. This paper provides efficient methods for distributing points on parametric curves. The first two methods, *adaptive resolution refinement* and *hybrid grid point distribution*, are approaches aimed at discretizing a curve to satisfy geometric and computational accuracy constraints. Adaptive resolution refinement is primarily targeted at obtaining a discretization with high geometric fidelity. This method was developed to provide a discretization where the maximum deviation of linear edge segments is never more than some ϵ distance from the true curve. The second method, hybrid grid point distribution, seeks to provide enhanced computational accuracy by concentrating points in regions of the curve to satisfy numerical accuracy requirements (i.e., concentrating points in areas of high curvature or where large gradients are expected in the solution).

The last two approaches presented, *parametric curve grid smoothing* and *nonparametric curve grid smoothing*, focused on redistributing points along the curve to achieve a degree of *smoothness* in the discrete approximation. Parametric curve grid smoothing is a technique used to achieve a smooth edge length variation along the curve while remaining faithful to the original parametric description. Nonparametric curve grid smoothing is an approach to achieve a smooth discretization on a reformulated curve, as it is often known a priori that the original curve is a suboptimal representation of a desired description. In this case, additional information from the physics simulation (such as area or mass conservation) is used to reform the discrete representation to increase its suitability for subsequent simulations.

REFERENCES

[1] M. Kosters, *Curvature dependent parameterization of curves and surfaces*, Comput. Aid. Des., 23 (1991), pp. 569–579.

[2] S. Z. Li, *Adaptive sampling and mesh generation*, Comput. Aid. Des., 27 (1995), pp. 235–240.

[3] J. Anderson, A. Khamayseh, and B. Jean, *Adaptive Resolution Refinement*, LANL Technical Report LA-UR-96-3105, Los Alamos National Laboratory, Los Alamos, NM, 1996.

[4] J. C. Cuilliere, *A direct method for the automatic discretization of 3D parametric curves*, Comput. Aid. Des., 29 (1997), pp. 639–647.

[5] J. D. Hoffman, *Relationship between the truncation errors of centered finite difference approximation on uniform and nonuniform meshes*, J. Comput. Phys., 46 (1982), pp. 469–474.

[6] M. Vinokur, *On one-dimensional stretching functions for finite difference calculation*, J. Comput. Phys., 50 (1983), pp. 215–234.

[7] C. W. Mastin, *Arc length based grid distribution for surface and volume grids*, in Proceedings of the Fifth International Numerical Grid Generation Conference, 1996, pp. 87–96.

[8] A. Khamayseh and A. Kuprat, *Surface grid generation systems*, in Handbook of Grid Generation, J. F. Thompson, B. K. Soni, and N. P. Weatherill, eds., CRC Press, New York, 1999, pp. 9.1–9.29.

[9] B. Soni and S. Yang, *Nurbs-based surface grid redistribution and remapping algorithms*, Comput. Aided Geom. Design, 12 (1995), pp. 675–692.

[10] G. Taubin, *Curve and surface smoothing without shrinkage*, in Proceedings of the Fifth International Conference on Computer Vision, 1995, pp. 852–857.

[11] K. Miller, *A geometrical-mechanical interpretation of gradient-weighted moving finite elements*, SIAM J. Numer. Anal., 34 (1997), pp. 67–90.

[12] L. Piegl and W. Tiller, *The NURBS Book*, Springer-Verlag, Berlin, 1995.

[13] J. F. Thompson, Z. U. A. Warsi, and C. W. Mastin, *Numerical Grid Generation: Foundations and Applications*, North-Holland, New York, 1985.

[14] A. Kuprat, A. Khamayseh, D. George, and L. Larkey, *Volume conserving smoothing for piecewise linear curves, surfaces, and triple lines*, J. Comput. Phys., 172 (2001), pp. 99–118.

SIAM J. SCI. COMPUT.
Vol. 23, No. 5, pp. 1485–1502

MULTIDIMENSIONAL LEAST SQUARES FLUCTUATION DISTRIBUTION SCHEMES WITH ADAPTIVE MESH MOVEMENT FOR STEADY HYPERBOLIC EQUATIONS[*]

M. J. BAINES[†], S. J. LEARY[‡], AND M. E. HUBBARD[§]

Abstract. Optimal meshes and solutions for steady conservation laws and systems within a finite volume fluctuation distribution framework are obtained by least squares methods incorporating mesh movement. The problem of spurious modes is alleviated through adaptive mesh movement, the least squares minimization giving an obvious way of determining the movement of the nodes and also providing a link with equidistribution. The iterations are carried out locally node by node, which yields good control of the moving mesh. For scalar equations an iteration which respects the flow of information in the problem significantly accelerates the convergence.

The method is demonstrated on a scalar advection problem and a shallow water channel flow problem. For discontinuous solutions we introduce a least squares shock fitting approach which greatly improves the treatment of discontinuities at little extra expense by using degenerate triangles and moving the nodes. Examples are shown for a discontinuous shallow water channel flow and a shocked flow in gasdynamics governed by the compressible Euler equations.

Key words. least squares, fluctuation distribution, mesh movement, hyperbolic equations

AMS subject classification. 76M12

PII. S1064827500370202

1. Introduction.

Finite volume schemes of fluctuation distribution type for the approximation of steady first order hyperbolic equations and systems are now well established. In particular, the class of multidimensional upwind schemes on unstructured triangular meshes has been very successful [13]. The least squares methods of finite volume type discussed in this paper also belong to this family, although their properties differ.

Roe was the first to suggest the fluctuation-distribution framework for steady first order hyperbolic PDEs and systems in multidimensions [10]. In this approach a fluctuation (proportional to the PDE residual) is defined on each cell of the mesh and distributed by signals to the nodes of the cell; i.e., weighted fractions of the fluctuation are added to the solution values at the nodes of the cell. This distribution is carried out for each cell, and the cumulative update at a node is the sum of the weighted contributions from cells with that node as a target. To reach steady state the procedure is repeated, updating the solution values until the total increments at every node have become zero, at which point the process is said to have converged.

As pointed out in [11], a descent method applied to the least squares method within a finite volume framework is also a fluctuation-distribution scheme. In the

[*] Received by the editors April 5, 2000; accepted for publication (in revised form) February 21, 2001; published electronically January 4, 2002.

http://www.siam.org/journals/sisc/23-5/37020.html

[†] Department of Mathematics, P.O. Box 220, Reading, RG6 6AX, UK (M.J.Baines@reading.ac.uk). The work of the first author was supported by a research grant from the Engineering and Physical Science Research Council (EPSRC).

[‡] Department of Engineering, University of Southampton, Southampton, S017 IBJ, UK (S.J.Leary@soton.ac.uk). The work of the second author was supported by a Collaborative Awards in Science and Engineering (CASE) studentship.

[§] Department of Computer Science, University of Leeds, Leeds, LS2 9JT, UK (meh@comp.leeds.ac.uk). The work of the third author was supported by a research grant from the Engineering and Physical Science Research Council (EPSRC).

present paper this idea is developed further, using among other things the connection between least squares minimization and equidistribution [1], and in particular is extended to nonlinear systems of PDEs.

For fluctuation distribution schemes in general, even though the total increments at a node may have converged to zero, the individual cell residuals (or fluctuations) need not have vanished but only their weighted sums, leading sometimes to an unsatisfactory solution. One way to alleviate the difficulty is to increase the number of degrees of freedom available by including the mesh locations as additional variables in the least squares minimization and hence moving the mesh. As a consequence, when the total increments at a node converge, the individual fluctuations in a cell are closer to zero and yield a better approximation to the PDE and the solution. In the case of scalar problems, spurious solutions may be eliminated altogether and the outcome identified with an approximate method of characteristics.

Repositioning the nodes in this way leads to conservation and a measure of equidistribution, the latter ensuring that convergence takes place uniformly with respect to the mesh.

In this paper the method is applied to a scalar PDE problem and a shallow water channel flow problem, both of whose solutions are smooth.

For problems with nonsmooth solutions, least squares methods are known to give poor solutions close to discontinuities. Here we take a shock fitting approach and use a least squares moving mesh method to improve the position of the shock. In recent years a great deal of effort has been put into mesh refinement near shocks using mesh subdivision, but substantial improvements in shock resolution can also be obtained by making minor adjustments to the mesh. We introduce degenerate cells in the vicinity of the shock and a least squares shock fitting procedure to adjust its position. A multidimensional upwinding shock capturing scheme [13] is used to generate an initial solution and a first approximation to the position of the shock. A least squares shock fitting approach is then used to improve the position of the shock [4], [7]. This is achieved by a least squares minimization of a measure of the jump condition over nodal positions in degenerate cells. In the smooth regions on either side of the shock the least squares method may then be expected to work well.

Results are shown for a scalar problem with a contact discontinuity, a shallow water problem in a constricted channel with a hydraulic jump, and an Euler gasdynamics problem with an exact solution, including a shock reflection.

The layout of the paper is as follows. In section 2 we give the definition of the fluctuation and its functional form in certain cases. Section 3 describes fluctuation distribution schemes and least squares methods (with descent) in a finite volume framework. In section 4 we discuss the role of node movement in improving the accuracy of solutions and exploiting the link between least squares and equidistribution. Details of the descent methods used for achieving least squares minima are described in section 5, and an upwinding strategy is described in section 6. Results are shown in section 7 for a scalar advection example and a problem involving a nonlinear system of equations, the Shallow Water Equations.

The role of degenerate cells in generating discontinuous solutions is discussed in section 8. Results for some discontinuous scalar problems and nonlinear systems are shown in section 9 with conclusions in section 10.

2. Fluctuations.

We consider the two-dimensional conservation law

$$\operatorname{div}(\underline{f}(u)) = 0 \tag{2.1}$$

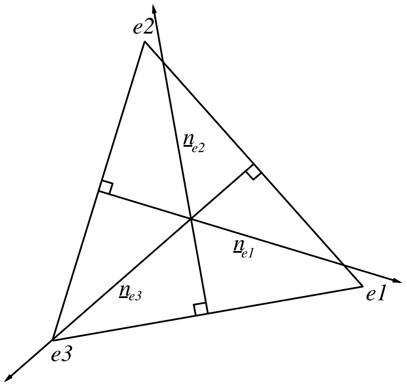


FIG. 1. *A general triangular cell e.*

with integral form

$$\oint_{\Gamma} \underline{f}(u).\underline{\hat{n}}d\Gamma = 0, \tag{2.2}$$

where $\underline{\hat{n}}$ is the inward normal to an arbitrary closed surface Γ in a domain Ω. The boundary condition is an inflow condition over Γ_1, the part of the surface for which $\frac{\partial \underline{f}}{\partial u}.\underline{\hat{n}} \geq 0$.

Let the domain be divided into triangles Ω_e, and let $\underline{f}$ be approximated by a piecewise linear function $\underline{F}$. Then we define the fluctuation in triangle Ω_e to be

$$\phi_e = \oint_{\Gamma_e} \underline{F}.\underline{\hat{n}}d\Gamma, \tag{2.3}$$

where Γ_e is the perimeter of Ω_e.

We also define the average residual

$$\overline{R}_e = \frac{1}{S_e}\int_{\Omega_e} \mathrm{div}\underline{F}d\Omega = \frac{1}{S_e}\oint_{\Gamma_e} \underline{F}.\underline{\hat{n}}d\Gamma = \frac{\phi_e}{S_e}, \tag{2.4}$$

where S_e is the area of triangle e.

Since $\underline{F}$ is linear in the triangle we can use a trapezium rule quadrature to write (2.3) as

$$\phi_e = \frac{1}{2}\left\{(\underline{F}_{e1} + \underline{F}_{e2}).\underline{n}_{e3} + (\underline{F}_{e2} + \underline{F}_{e3}).\underline{n}_{e1} + (\underline{F}_{e3} + \underline{F}_{e1}).\underline{n}_{e2})\right\}, \tag{2.5}$$

where $\underline{n}_{ei}$ $(i = 1, 2, 3)$ is the inward unit normal to the ith edge of triangle e (opposite the vertex ei), as shown in Figure 1, multiplied by the length of that edge. It is easy to verify that, for any triangle,

$$\underline{n}_{e1} + \underline{n}_{e2} + \underline{n}_{e3} = 0, \tag{2.6}$$

so the fluctuation (2.5) may be written as

$$\phi_e = -\frac{1}{2}\left\{\underline{F}_{e1}.\underline{n}_{e1} + \underline{F}_{e2}.\underline{n}_{e2} + \underline{F}_{e3}.\underline{n}_{e3}\right\}, \tag{2.7}$$

or, since $\underline{n}_{ei} = (\Delta Y_{ei}, -\Delta X_{ei})$,

$$\phi_e = -\frac{1}{2}\sum_{ei=1}^{3}\left(F_{ei}\Delta Y_{ei} - G_{ei}\Delta X_{ei}\right), \tag{2.8}$$

where $\underline{F} = (F, G)$ and $\Delta_{e1}X = X_{e2} - X_{e3}$ denotes the difference in X taken across the side opposite node $e1$ in a counterclockwise sense (and similarly for $\Delta_{e2}X$ and $\Delta_{e3}X$). A dual form of the fluctuation is obtained by rewriting (2.7) as

$$\phi_e = \frac{1}{2}\sum_{ei=1}^{3}\left(Y_{ei}\Delta F_{ei} - X_{ei}\Delta G_{ei}\right). \tag{2.9}$$

We aim to set the fluctuations ϕ_e to zero in order to satisfy (2.1).

In the case where $\underline{f}$ is of the form

$$\underline{f} = \underline{a}(\underline{x})u, \tag{2.10}$$

where $\underline{a}(\underline{x})$ is a divergence-free velocity field, the PDE (2.1) reduces to the advection equation

$$\underline{a}(\underline{x}).\underline{\nabla}u = 0. \tag{2.11}$$

Then the fluctuation may be written

$$\phi_e = -\frac{1}{2}\sum_{ei=1}^{3}\left(a_{ei}U_{ei}\Delta Y_{ei} - b_{ei}U_{ei}\Delta X_{ei}\right), \tag{2.12}$$

where $\underline{a} = (a, b) = (a(X_{ei}, Y_{ei}), b(X_{ei}, Y_{ei}))$.

Now consider systems of nonlinear hyperbolic equations

$$\text{div}\underline{\mathbf{f}}(\mathbf{u}) = 0 = \underline{A}(\mathbf{u}).\underline{\nabla}\mathbf{u}, \tag{2.13}$$

where $\underline{A}$ is a vector of the Jacobian matrices $(A, B)^T$. The integral form is

$$\oint_\Gamma \underline{\mathbf{f}}(\mathbf{u}).\underline{\hat{n}}d\Gamma = 0, \tag{2.14}$$

and the fluctuation (with $\mathbf{f}$ approximated by $\mathbf{F}$) is

$$\boldsymbol{\phi}_e = -\frac{1}{2}\left(\underline{\mathbf{F}}_{e1}.\underline{n}_{e1} + \underline{\mathbf{F}}_{e2}.\underline{n}_{e2} + \underline{\mathbf{F}}_{e3}.\underline{n}_{e3}\right) \tag{2.15}$$

$$= -\frac{1}{2}\sum_{ei=1}^{3}\left(\mathbf{F}_{ei}\Delta Y_{ei} - \mathbf{G}_{ei}\Delta X_{ei}\right) \tag{2.16}$$

with dual form

$$\boldsymbol{\phi}_e = \frac{1}{2}\sum_{ei=1}^{3}\left(Y_{ei}\Delta\mathbf{F}_{ei} - X_{ei}\Delta\mathbf{G}_{ei}\right). \tag{2.17}$$

Two systems of interest are the Euler equations of gasdynamics and the Shallow Water Equations.

3. Fluctuation distribution schemes and least squares. In fluctuation distribution schemes we seek to set the fluctuations ϕ_e to zero via an iterative procedure with an index n, say. In this procedure the ϕ_e^n, obtained by substituting an estimate U^n into the (F, G) in (2.8), are distributed to nodes of the mesh in order to give a U^{n+1} for which the ϕ_e^{n+1} are smaller. At each stage of the iteration, for each triangle Ω_e, a weighted amount of ϕ_e is added to the values of the solution at the vertices of the triangle. In the multidimensional upwind schemes [13], [8] the weights are chosen so that the schemes are conservative, positive, and linearity preserving. Conservation is ensured if the weights in each triangle sum to unity.

In the least squares descent method we seek to minimize either the L_2 norm of the average residual (see (2.4)) or the l_2 norm of the vector of fluctuations, using a gradient descent method. This l_2 norm is useful since it is bounded even for the degenerate triangles considered in section 7.

The square of the L_2 norm of the average residual, from (2.4), is

$$\mathcal{F}_1 = \sum_e \int_{\Omega_e} \overline{R}_e^2 d\Omega = \sum_e S_e \overline{R}_e^2 = \sum_e \frac{\phi_e^2}{S_e}, \tag{3.1}$$

or, in the systems case,

$$\mathcal{F}_1 = \sum_e \frac{\boldsymbol{\phi}_e^t \boldsymbol{\phi}_e}{S_e} \tag{3.2}$$

(cf. [11]). For the l_2 norm of the vector of fluctuations we have

$$\mathcal{F}_2 = \sum_e \phi_e^2 \qquad \text{or} \qquad \mathcal{F}_2 = \sum_e \boldsymbol{\phi}_e^t \boldsymbol{\phi}_e \tag{3.3}$$

in the systems case.

Using a gradient descent method to carry out the minimization, we find that each step adds weighted amounts of the ϕ_e in each triangle to the values of the solution at the vertices of the triangle and hence has the form of a fluctuation distribution scheme. For example, in the $\mathcal{F}_2$ case, since the gradient of ϕ_e^2 with respect to the nodal value U_j is

$$\left\{ 2 \frac{\partial \phi_e}{\partial U_j} \right\} \phi_e \tag{3.4}$$

a descent method will add a multiple of ϕ_e to U_j. The weight (in the curly bracket), from (2.8), is

$$w_{je} = 2\frac{\partial \phi_e}{\partial U_j} = -\frac{\partial}{\partial U_j} \sum_{ei=1}^{3} \{F_{ei}\Delta Y_{ei} - G_{ei}\Delta X_{ei}\} \tag{3.5}$$

$$= -\frac{dF_{je}}{dU_{je}}\Delta Y_{je} + \frac{dG_{je}}{dU_{je}}\Delta X_{je} \tag{3.6}$$

$$= -a(U_{je})\Delta Y_{je} + b(U_{je})\Delta X_{je}, \tag{3.7}$$

where je is the node of triangle e corresponding to j and we have used

$$(a(U), b(U)) = \left(\frac{dF}{dU}, \frac{dG}{dU} \right). \tag{3.8}$$

In the case of differentiation with respect to $\underline{X}_j$, the gradient of ϕ_e^2 is

$$\left\{2\frac{\partial \phi_e}{\partial \underline{X}_j}\right\}\phi_e \tag{3.9}$$

and a descent method will add a multiple of ϕ_e to $\underline{X}_j$. This time the vector weights, using (2.17), are

$$\underline{w}_{je} = (0,1)^T \Delta F_{je} + (-1,0)^T \Delta G_{je}. \tag{3.10}$$

For systems of equations the corresponding matrix weights corresponding to (3.7) and (3.10) are

$$W_{je} = -A(\mathbf{U}_{je})\Delta Y_{je} + B(\mathbf{U}_{je})\Delta X_{je} \tag{3.11}$$

and

$$\underline{W}_{je} = (0,1)^T \Delta \mathbf{F}_{je} + (-1,0)^T \Delta \mathbf{G}_{je}. \tag{3.12}$$

For the advection equation (2.10) we have from (2.12) the weights

$$w_{je} = -a(X_{je}, Y_{je})\Delta Y_{je} + b(X_{je}, Y_{je})\Delta X_{je} \tag{3.13}$$

for U variations and, using a dual form of (2.12),

$$\underline{w}_{je} = \frac{\partial}{\partial \underline{X}_j} \sum_{ei=1}^{3} - \left\{\Delta\left(a(X_{ei}, Y_{ei})U_{ei}\right) Y_{ei} + \Delta\left(b(X_{ei}, Y_{ei})U_{ei}\right) X_{ei}\right\} \tag{3.14}$$

for the $\underline{X}$ variations.

Similar sets of weights may be found in the minimization of $\mathcal{F}_1$. In particular, (3.9) generalizes to

$$\frac{\partial}{\partial \underline{X}_j}\left(\frac{\phi_e^2}{S_e}\right) = \left\{\frac{2}{S_e}\frac{\partial \phi_e}{\partial \underline{X}_j} - \frac{\phi_e}{S_e^2}\frac{\partial S_e}{\partial \underline{X}_j}\right\}\phi_e. \tag{3.15}$$

4. Moving the nodes. There are two motivations for moving the nodes. The first is the problem of spurious solutions. The number of equations given by (2.3) is equal to the number of triangles in the mesh, but the number of unknowns is a multiple of the number of nodes. In general these are different. If the number of equations exceeds the number of unknowns it is impossible to satisfy all the equations. For any iteration of fluctuation distribution type in which fluctuations are added to the vertices of the mesh with weights, convergence of the nodal updates does not imply that the fluctuations vanish. In particular, in the least squares descent approach the norms (3.1), (3.3) are not necessarily driven down to zero. However, if we allow the coordinates of the vertices to become additional unknowns of the problem, the number of degrees of freedom is increased and the solution is improved.

For scalar problems the number of unknowns then exceeds the number of equations and there are infinitely many solutions which make the norms zero. A unique solution is obtained if the number of unknowns is equal to the number of equations, and this may be achieved in a scalar problem by including just one coordinate per node in the list of unknowns. The fluctuations may then be driven to zero by a fluctuation-distribution scheme. The result is an approximate method of characteristics, as in Example 1 below. The accuracy of the approximate solution depends only on the coarseness and/or connectivity of the mesh. For a system of two equations in two dimensions, the number of unknowns is equal to the number of equations when the nodes are allowed to move in both directions, and this has been studied in [11]. For systems such as the Shallow Water or the Euler Equations of gasdynamics the number of equations is always less than or equal to the number of unknowns, but the inclusion of nodal variables significantly increases the number of degrees of freedom.

The second motivation comes from a link with equidistribution. As in [1], the identity

$$\left(\sum_e S_e\right)\left(\sum_e S_e \bar{R}_e^2\right) = \left(\sum_e \phi_e\right)^2 + \sum_{e_1>e_2} S_{e_1} S_{e_2} \left(\bar{R}_{e_1} - \bar{R}_{e_2}\right)^2 \tag{4.1}$$

shows that, if the total area of the domain $\sum_e S_e$ is fixed, then driving the norm $\mathcal{F}_1$ (which from (3.1) equals $\sum_e S_e \bar{R}_e^2$) down to zero forces both terms on the right-hand side of (4.1) to zero, resulting in both global conservation and residual "equidistribution." The first follows because of the cancellation property

$$\phi = \sum_e \phi_e = \frac{1}{2} \sum_e \sum_{ei=1}^{3} \left(-F_{ei}\Delta Y_{ei} + G_{ei}\Delta X_{ei}\right) \tag{4.2}$$

$$= \frac{1}{2} \sum_b \left(-F_b \Delta Y_b + G_b \Delta X_b\right), \tag{4.3}$$

so that the total ϕ over the domain is equal to a sum over boundary values b only. Hence the first term on the right-hand side of (4.1) is a measure of global conservation, while the second term is a measure of equidistribution of the average residual $\overline{R}_e$.

In a similar way the identity

$$\left(\sum_e 1\right)\left(\sum_e \phi_e^2\right) = \left(\sum_e \phi_e\right)^2 + \sum_{e_1>e_2} \left(\phi_{e_1} - \phi_{e_2}\right)^2 \tag{4.4}$$

(see [1]) ensures that, provided that the number of triangles $\sum_e 1$ remains fixed, the act of driving the norm $\sum_e \phi_e^2$ down to zero also forces global conservation and a measure of equidistribution of the fluctuations ϕ_e to go to zero. These statements generalize immediately to systems of equations.

The global conservation term (4.3) is evidently unaffected by any adjustment to the values at the interior nodes. Therefore a reduction in the sum of squares term on the left-hand side of (4.1) or (4.4) due to such adjustments simply serves to improve the quality of the equidistribution.

We shall discuss the use of least squares descent methods as fluctuation distribution schemes in this context. Unlike multidimensional upwinding [2], such an approach has the advantage of a norm to minimize which can readily be used to generate the movement of the mesh as well as inducing global conservation and equidistribution in the sense described above.

5. The descent methods. We give now the details of the minimization of $\mathcal{F}_2$ with respect to the nodal values U_j and coordinates $\underline{X}_j$, using a gradient descent method. The steepest descent method generates contributions from the set of triangles je surrounding node j, to be added to the values of U_j and $\underline{X}_j$, of the form

$$\delta U_j = -\tau_2 \sum_{je} \left\{ 2 \frac{\partial \phi_{je}}{\partial U_j} \right\} \phi_{je}, \qquad \delta \underline{X}_j = -\sigma_2 \sum_{je} \left\{ 2 \frac{\partial \phi_{je}}{\partial \underline{X}_j} \right\} \phi_{je} \tag{5.1}$$

(see (3.4) and (3.9)), where τ_2 and σ_2 are suitably chosen relaxation factors, and the negative sign ensures that we go down the gradient. The relaxation parameters control the step taken in the descent direction and are generally chosen via a line search or a local quadratic model. Sometimes, however, it is necessary to take an empirical approach to the choices of these factors.

In this paper we use a splitting technique, first minimizing $\mathcal{F}_2$ with respect to U_j with $\underline{X}_j$ held constant and then minimizing $\mathcal{F}_2$ with respect to $\underline{X}_j$ with U_j held constant. (It is possible, though unlikely, that the constrained nature of the minimization may lead to a saddle point.)

Consequently, for the minimization over U we may construct a quadratic model in which the relaxation parameter is

$$\left(\frac{\partial^2 \mathcal{F}_2}{\partial U_j^2} \right)^{-1} = \left(\frac{\partial^2}{\partial U_j^2} \sum_{je} \phi_{je}^2 \right)^{-1} \tag{5.2}$$

$$= \left(\frac{\partial^2}{\partial U_j^2} \sum_{je} \sum_{ei} \frac{1}{4} \underline{n}_{ei}^T \, \underline{F}_{ei} \underline{F}_{ei}^T \, \underline{n}_{ei} \right)^{-1} \tag{5.3}$$

by (2.7). Let us now linearize $\underline{F}_{ei}$ as $\underline{a}_{ei} U_{ei}$ so that the relaxation factor becomes

$$\left(\frac{\partial^2}{\partial U_j^2} \sum_{je} \sum_{ei} \frac{1}{4} \underline{n}_{ei}^T \, \underline{a}_{ei} U_{ei}^2 \underline{a}_{ei}^T \, \underline{n}_{ei} \right)^{-1} \tag{5.4}$$

$$= \left(\sum_{je} \frac{1}{4} \underline{n}_{je}^T \, \underline{a}_{je} \, \underline{a}_{je}^T \, \underline{n}_{je} \right)^{-1} . \tag{5.5}$$

For the $\underline{X}$ minimization of $\mathcal{F}_2$ the functional is already quadratic, giving the relaxation factor

$$\left(\frac{\partial^2 \mathcal{F}_2}{\partial \underline{X}_j^2} \right)^{-1} = \left(\frac{\partial^2}{\partial \underline{X}_j^2} \sum_{je} \sum_{ei} \frac{1}{4} \underline{F}_{ei}^T \, \underline{n}_{ei} \, \underline{n}_{ei}^T \, \underline{F}_{ei} \right)^{-1} \tag{5.6}$$

which is

$$= \left(- \sum_{je} \left(\underline{F}_{je1}^T \underline{F}_{je1} + \underline{F}_{je2}^T \underline{F}_{je2} \right) \right)^{-1} \tag{5.7}$$

for each coordinate, where $je1, je2$ are the vertices of the triangle je other than j. Alternatively, a line search may be carried out on each $\underline{X}_j$.

For the advection equation (2.10) a quadratic model may be obtained by freezing the advection speed in calculating the second derivative in the quadratic model (see (2.12)).

For the minimization of $\mathcal{F}_1$, rather than $\mathcal{F}_2$, we obtain an approximate quadratic model simply by inserting the factor S_{je}^{-1} between the a_{je}'s or A_{je}'s.

These choices generalize to systems of equations where (5.5) becomes

$$\left(\sum_{je} \frac{1}{4} \underline{n}_{je}^T \ \underline{A}_{je} \ \underline{A}_{je}^T \underline{n}_{je}\right)^{-1}, \tag{5.8}$$

where $\underline{A} = (A, B)$, and where $\underline{\mathbf{F}}$ has been linearized as $\underline{A}\mathbf{U}$.

The iterations are carried out by continually sweeping through the nodes of the mesh in a local manner. The identities (4.1) or (4.4) also hold on each patch of triangles surrounding a node, showing that least squares minimization leads to local conservation over the boundary of the patch and equidistribution over the triangles of the patch.

The sweeps through the nodes of the mesh may be carried out either in a Jacobi or a Gauss–Seidel manner. The local approach is helpful in controlling the mesh quality.

6. Upwinding. Generally, the rate of convergence is slow or very slow. However, we can show that in the scalar case convergence can be accelerated significantly by an awareness of the origin of the problem. One consequence of minimizing the least squares norm of the residual or the fluctuation of the equation $\underline{a}.\nabla u = 0$ is that the original equation is embedded in the second order degenerate elliptic equation $-\underline{a}.\nabla(\underline{a}.\nabla u) = 0$ (see e.g. [9]). The correct solution is picked out from the larger set of solutions by the outflow condition, which is the original differential equation $\underline{a}.\nabla u = 0$ applied at the outflow boundary. Indeed we may write the second order equation as the system

$$\underline{a}.\nabla u = v \tag{6.1}$$

with U given on Γ_2 and

$$-\underline{a}.\nabla v = 0 \tag{6.2}$$

with V given on Γ_1. The first of these is the solution of the original PDE with a source term v, which is the solution of the second equation. For the second equation the analytic solution is $v = 0$, but numerically a nonzero v will be generated building up from the outflow (the characteristics run backwards in (6.2)), forcing a nonzero source term in (6.1).

As befits an elliptic solver, the least squares descent method updates are distributed to all the nodes in a triangle, but it may be argued that, because of the hyperbolic nature of the original equation, the updates should exhibit an upwind bias, as in the case of multidimensional upwinding, and the nonzero v solution should be suppressed.

One way of achieving the upwind bias (see [3], [7]) is to carry out the minimization of the functional within each cell over only *downwind* nodal values. Furthermore, we allow temporary discontinuities in U at each node by letting the solution have one value associated with cells upwind of the node but another with downwind cells. The updates resulting from this minimization still reduce the functional but at the expense of making U discontinuous. However, we may follow this minimization step by a second projection step which resets the *upwind* values of U so as to restore the continuity of U. This is not a descent step and may increase the fluctuation. Nevertheless, we may iterate on the two steps, seeking convergence. If convergence is attained the discontinuities have tended to zero, and we have a continuous U which also minimizes the functional since its gradient is zero. Since the minimization is constrained, a higher value of the functional may result (the two projections cancelling each other out), but further improvement may be found at this point by switching to the full least squares iteration.

By a similar argument on the dual form (2.9) of the fluctuation, the $\underline{X}$ contributions should also be upwinded (although the boundary conditions differ from those on v).

Not surprisingly we find that convergence is much faster, not only for the U variations but also for the $\underline{X}$ variations. The algorithm has a strong upwind bias which reflects the nature of the original problem and its dependence on characteristics. In fact the two steps taken together are equivalent to simply suppressing the upwind updates in the least squares descent method. With an appropriate scaling the U step is simply the Low Diffusion Scheme A (LDA) scheme of multidimensional upwinding [13].

We now give results for two problems in which these techniques are used.

7. Numerical results for continuous solutions.

Example 1. We first consider the scalar two-dimensional advection equation, considered in [11],

$$\underline{a}(\underline{x}).\nabla u = 0, \tag{7.1}$$

where $\underline{a}(\underline{x}) = (y, -x)$ in a rectangle $-1 \leq x \leq 1, 0 \leq y \leq 1$, which generates a semicircular hump swept out by the initial data, here chosen to be

$$U = \begin{cases} 1, & -0.6 \leq x \leq -0.5, \\ 0 & \text{otherwise.} \end{cases} \tag{7.2}$$

Results are shown in Figures 2 and 3 on a fixed and moving mesh, respectively. Fastest convergence occurs when the sweeping is upwinded, taking into account the hyperbolic nature of the equation.

As expected, the solution on a fixed mesh is poor. On the other hand, when the mesh takes part in the minimization the norm $\mathcal{F}_1$ is driven down to machine accuracy. The redistribution effected by the least squares minimization forces global conservation and equidistributes ϕ amongst the triangles [1], leading to more uniform convergence. The cell edges have approximately aligned with characteristics in regions of nonzero ϕ, allowing a highly accurate solution to be obtained.

The left-hand graph in Figure 4 shows the convergence of the solution updating procedure using

(a) steepest descent globally with $\tau_1 = 0.5$;
(b) optimal local updates (quadratic model);
(c) optimal local updates over downwind cells only.

Convergence is improved in (b) and (c). Even though (c) is not monotonic it converges very quickly, albeit to a higher value, due to the minimization being constrained.

The convergence rates obtained when the nodes are allowed to move are shown in Figure 4 (right). Once again we start from the converged solution on the fixed grid and use

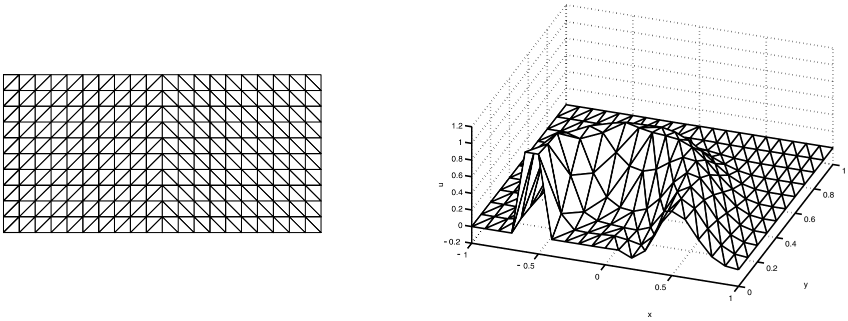


FIG. 2. *Initial grid and solution for Example* 1.

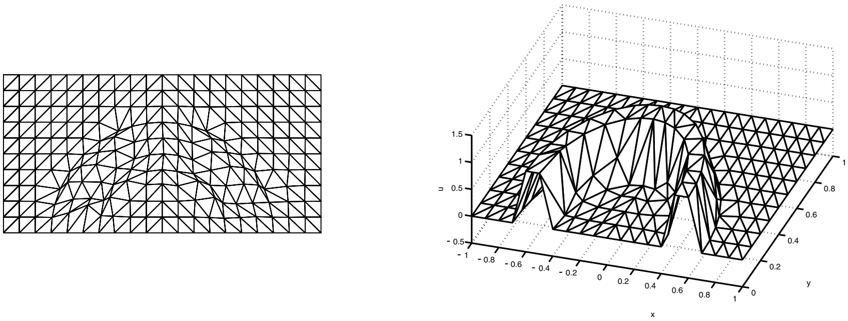


FIG. 3. *Final grid and solution for Example* 1.

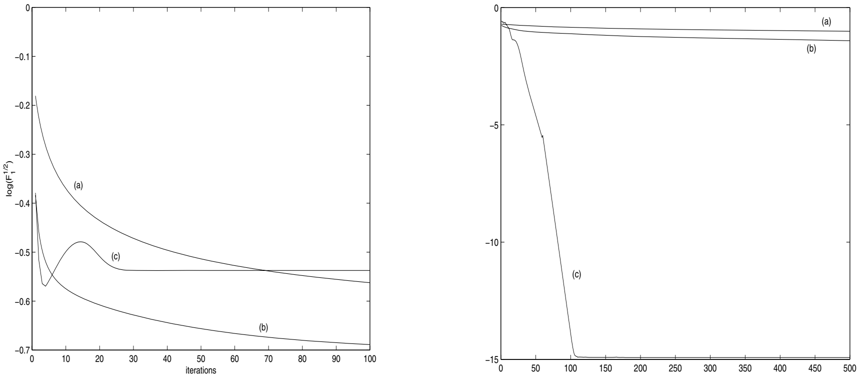


FIG. 4. *Comparisons of convergence histories.*

(a) steepest descent globally with $\tau_1 = 0.5$ and $\sigma_1 = 0.01$;
(b) Hessian local updates;
(c) Hessian local updates over downwind cells only.

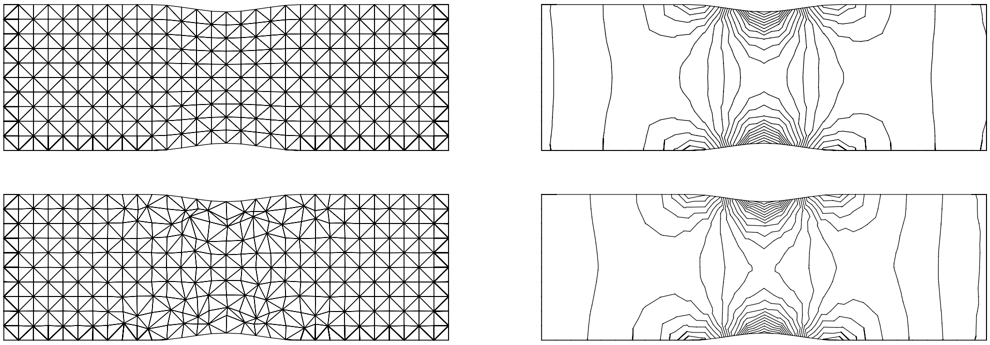


Fig. 5. *Initial grid and solution for Example* 2.

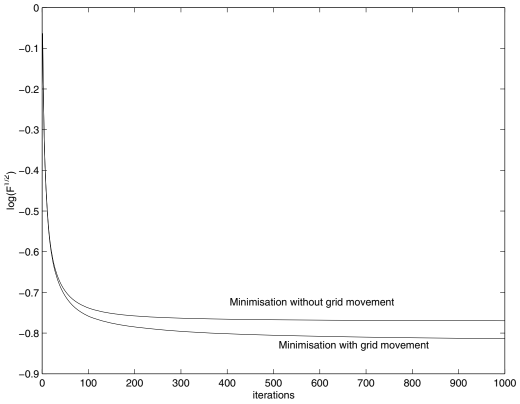


Fig. 6. *Convergence histories with and without mesh movement.*

A small amount of mesh smoothing was included in (b) and (c). In particular, (b) became stuck in a local minimum if more iterations were used. Node locking was a problem with the full least squares approach: node removal or steepest descent updates could be used to alleviate this problem but when tried these still took over 1000 iterations, so they were not competitive when compared to the upwinding approach, which yielded the best result.

Example 2. We now consider the system of equations (2.13) corresponding to a form of the homogeneous Shallow Water Equations written in conserved variables (see [5], [6]).

We shall consider a smooth subcritical constricted channel flow governed by these equations. The computational domain represents a channel of length 3 meters and width 1 meter with a 5% bump in the middle third. The freestream Froude number is defined to be $F_\infty = 0.25$, and the freestream depth is $h_\infty = 1m$. The resulting flow is entirely subcritical and symmetric about the center of the constriction (the narrowest point in the channel).

The fixed mesh is shown at the top of Figure 5 and the least squares descent solution (depth contours) on the mesh beneath it. This is also the initial mesh for the iteration when the mesh is moved. The other pictures in the figure show the adapted mesh and solution on this mesh.

Figure 6 shows convergence histories for this problem with and without mesh movement. An improved minimum is achieved by incorporating mesh movement in the minimization process. However, $\mathcal{F}_1$ is not dramatically decreased in this subcritical problem because there are no particularly sharp features in the flow which can be improved upon by the use of mesh movement.

8. Use of degenerate triangles. In the presence of shocks or contact discontinuities least squares methods give inaccurate solutions which are unacceptable. One way to combat this problem is to divide the region into a number of domains and introduce degenerate triangles at the interface, as suggested in [12]. We may then use a least squares method with moving nodes to adjust the position of the discontinuity, as in shock fitting methods.

Consider again consider the scalar problem (2.1) as a PDE generating a shock or contact discontinuity. We first obtain an initial approximate solution U to this equation by the use of a multidimensional upwinding shock capturing scheme. An initial discontinuous solution may then be constructed by introducing degenerate (vertical) triangles in the regions identified as shocks, using a shock identification technique. In the results shown below this step was carried out manually, but the degenerate triangles can be added automatically using techniques that exist in the shock fitting literature (see for example [16], [15]). The corners of the degenerate triangles are designated as shocked nodes, and these form an internal boundary, on either side of which the least squares method may be applied in two smooth regions where it is known to perform well. The position of the discontinuity can then be improved by minimizing a least squares shock monitor based either on the fluctuation in the degenerate cells or on the jump condition.

Then consider the jump condition at a shock associated with the conservation law (2.1),

$$\underline{f}(u_L).\underline{n}_L + \underline{f}(u_R).\underline{n}_R = 0, \tag{8.1}$$

where $\underline{f}(u_L)$ and $\underline{f}(u_R)$ are the fluxes to the left and right of a discontinuous edge.

We obtain an improved location of the discontinuity in the discretized problem by minimizing an L_2 measure of the residual of the jump condition with respect to node positions, using a piecewise linear approximation F to f. Thus consider minimization of the norm

$$\mathcal{F}_3 = \sum_{Q\in\Omega} \int_{\Gamma_Q} (\underline{F}(U_L).\underline{n}_L + \underline{F}(U_R).\underline{n}_R)^2 d\Gamma \tag{8.2}$$

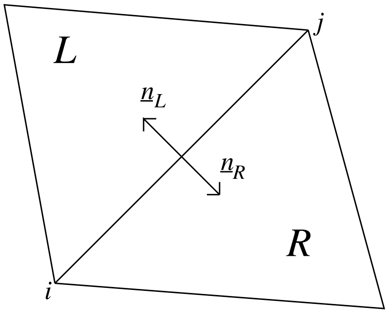


FIG. 7. *Cells on either side of a discontinuous edge.*

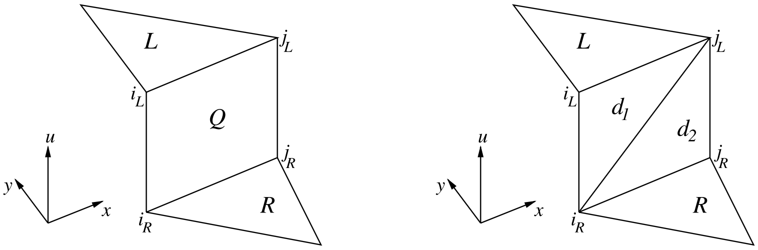


FIG. 8. *Degenerate quadrilaterals Q and triangles d_1, d_2.*

to update the position of the discontinuity where Γ_Q is the edge connecting nodes i and j in Figure 7, and $\underline{F}(U_L), \underline{F}(U_R)$ are the values of $\underline{F}$ at the left and right states.

We could have used an approximation based on degenerate triangles rather than quadrilaterals (see [4]). When updating the nodal positions $\underline{X}_{i_L}$ and $\underline{X}_{i_R}$ we require that they have the same update (so that the cell remains degenerate). The update comes from minimization with respect to their common position vector.

Consider the fluctuations ϕ_{d1} and ϕ_{d2} in the degenerate triangles d_1 and d_2 on the edge containing nodes i and j in Figure 8.

From (2.7) these are

$$\phi_{d_1} = -\frac{1}{2}\left[\underline{F}_i\right] \cdot \underline{n}_{i_L}, \qquad \phi_{d_2} = -\frac{1}{2}\left[\underline{F}_j\right] \cdot \underline{n}_{j_R}, \tag{8.3}$$

where the square bracket denotes the jump across the discontinuity. The contributions from two edges vanish in each case due to the degeneracy of the triangles.

Then

$$\phi_{d_1}^2 + \phi_{d_2}^2 = \frac{1}{4}\left\{\left([\underline{F}_i].\underline{n}_{i_L}\right)^2 + \left([\underline{F}_j].\underline{n}_{j_L}\right)^2\right\}, \tag{8.4}$$

and so we can also use

$$\mathcal{F}_4 = \sum_{e \in \Omega_D} \phi_e^2 \tag{8.5}$$

to improve the position of the shock, where Ω_D is the set of degenerate triangles. (Note that $\mathcal{F}_4$ is bounded because ϕ_e in (2.3) is always bounded, even at shocks where U is discontinuous. On the other hand, the average residual, given by (2.4), is not bounded since $S_e = 0$ at shocks.)

A descent least squares method can then be used on $\mathcal{F}_3$ or $\mathcal{F}_4$ to move the shocked nodes into a more accurate position, keeping the u_L and u_R values fixed. The procedure may be interleaved with a descent least squares method on $\mathcal{F}_1$ or $\mathcal{F}_2$ for the smooth solution on either side.

We now give some numerical results using this technique.

9. Numerical results for discontinuous solutions. We now show results from three problems which exhibit discontinuities, one scalar and the others for different nonlinear systems.

Example 3. The first of these problems is the advection of a contact discontinuity. We consider circular advection as in Example 1 but with initial data

$$U = \begin{cases} 1, & x \leq -0.5, \\ -1, & x \geq -0.5 \end{cases} \tag{9.1}$$

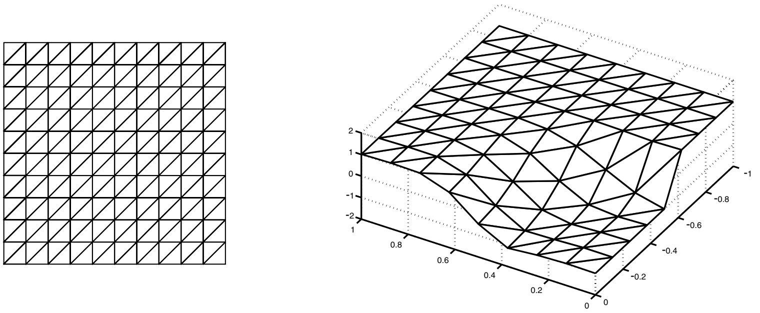


FIG. 9. *Fixed mesh and solution for Example* 3.

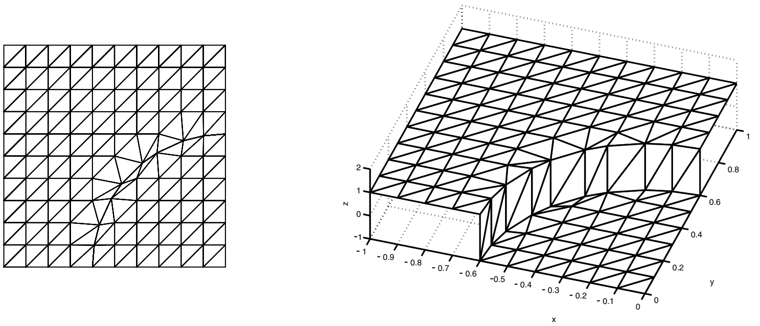


FIG. 10. *Moved mesh and solution for Example* 3.

on the inflow side. This represents the circular advection of a contact discontinuity.

Degenerate triangles are inserted vertically to connect the triangles on either side of the discontinuity. The solution updates come from a least squares descent method taken over nondegenerate elements. (The least squares updates to the solution come from nondegenerate elements.) The shock node adaptation is by the mimimization of F_4 (see (8.5)). Results are shown in Figures 9 and 10 for a fixed mesh and a moving mesh using degenerate triangles. Convergence histories are shown in Figure 11. The contact discontinuity has been accurately located through the use of the degenerate elements.

Example 4. Consider again the Shallow Water Equations system of Example 2. The problem which interests us here is that of a transcritical constricted channel flow which exhibits a hydraulic jump in the constriction. The computational domain represents a channel of length 3 meters and width 1 meter with a 10% bump in the middle third. The freestream Froude number is defined to be $F_\infty = 0.55$, the freestream depth is $h_\infty = 1m$, and the freestream velocity is given by $(u_\infty, v_\infty) = (1.72, 0)$.

An initial solution for the least squares shock fitting approach is found by the elliptic-hyperbolic Lax–Wendroff multidimensional upwinding scheme of Mesaros and Roe; see [8]. This time we seek to locate the hydraulic jump by adding degenerate quadrilaterals at the approximate position of the shock and seeking the best position of the shocked nodes. This is again achieved by using a least squares descent method on $\mathcal{F}_4$ with degenerate triangles to improve the position of the shock. Virtually identical results are obtained using $\mathcal{F}_3$ with quadrilaterals.

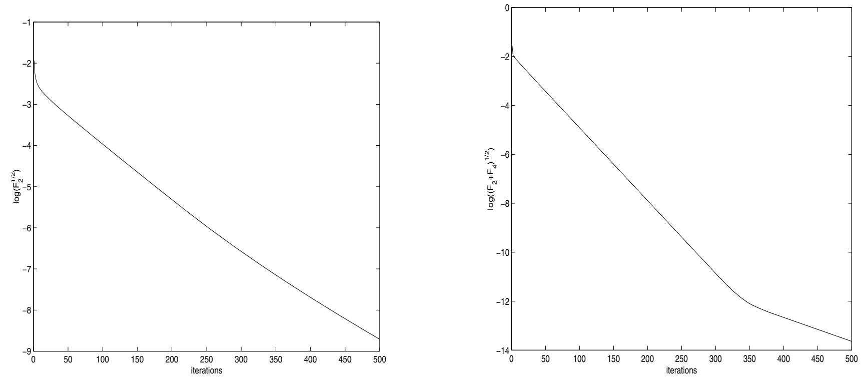


FIG. 11. *Convergence histories.*

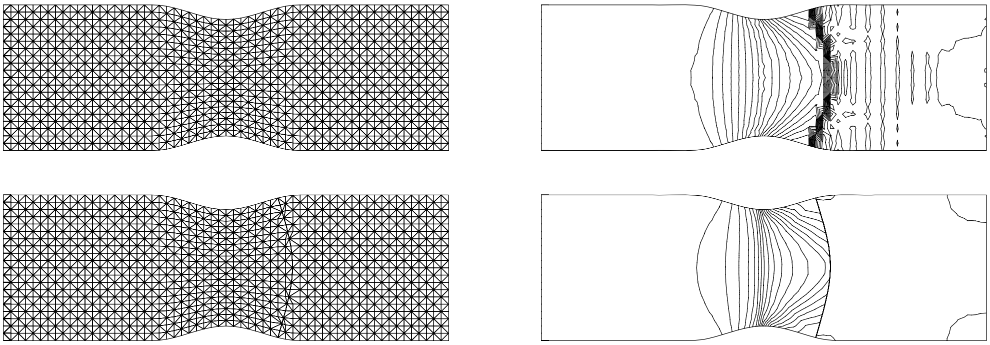


FIG. 12. *Results for Example* 4.

Results are shown in Figure 12, which shows the meshes and solution depth contours obtained. A bow-shaped hydraulic jump which is strongest at the boundaries is predicted, which agrees with solutions obtained using a shock capturing solution on a very fine mesh. Here it is achieved at little cost. Note not only the better shock resolution but also the much cleaner post-shock solution.

Example 5. Finally we consider the system (2.13) again but this time corresponding to the Euler equations of gasdynamics written in conserved variables [5].

This example is chosen to exhibit the shock fitting capabilities of the method for a purely supersonic flow which has an exact solution [17]. The computational domain is of length 3 meters and width 1 meter. Supersonic inflow boundary conditions, given by

$$\begin{aligned} \underline{U}(0,y) &= (1.0, 2.9, 0, 5.99073)^t, \\ \underline{U}(x,1) &= (1.69997, 4.45280, -0.86073, 9.87007)^t, \end{aligned} \tag{9.2}$$

are imposed on the left and upper boundaries, respectively. At the right-hand boundary supersonic outflow conditions are applied, while the lower boundary is treated as a solid wall.

The boundary conditions are chosen so that the shock enters the top left-hand corner at an angle of 29^o to the horizontal and is reflected by a flat plate on the lower boundary. The flow in regions away from shocks is constant. The same strategy is

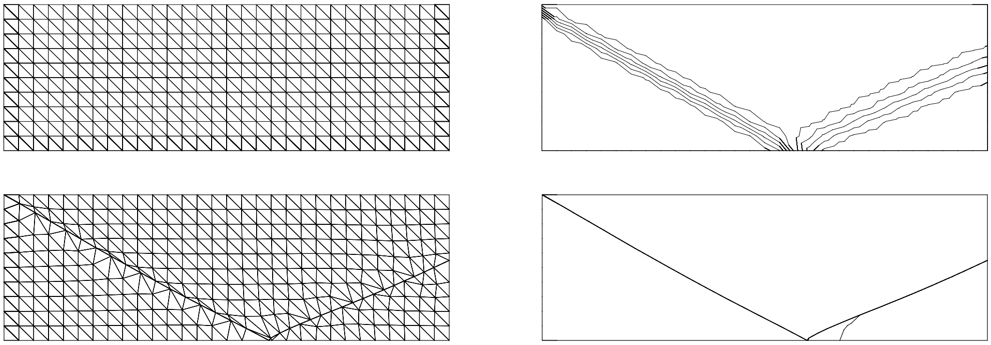


FIG. 13. *Results for Example* 5.

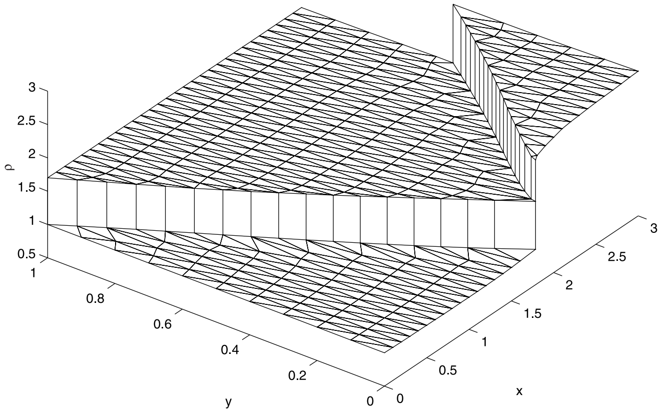


FIG. 14. *Solution (density) in* 3*D*.

employed as in the previous example, including the same shock capturing scheme, with the results shown in Figure 13, where the density contours are plotted. The predicted shock comes in from the top left hand at an angle of 29.2^o to the horizontal, and the solution is virtually constant apart from the discontinuities, in close agreement with the analytic solution (see Figure 14).

The angle made by the reflected shock with the horizontal is also in line with the theory. (See [14], which gives the angle as 23.3°.)

10. Conclusion. In this paper we have considered the approximate solution of steady first order PDEs by a least squares finite volume fluctuation distribution scheme with mesh movement. On fixed meshes, by the nature of the fluctuation distribution technique, the fluctuations on triangular meshes are not driven to zero. The solution may be improved by introducing extra degrees of freedom by adding node locations to the list of unknowns and moving the mesh. As a result, for scalar problems the fluctuations are driven down to zero (to machine accuracy), while for systems of equations the errors are much reduced. The descent least squares procedure with mesh movement also induces global conservation and equidistributes the fluctuation amongst the triangles, thus proceeding down to the steady limit in a uniform way.

For scalar problems convergence can be greatly accelerated by carrying out the iterations in an upwind manner.

For problems with discontinuities the descent least squares method does not give good solutions, but the mesh movement technique enables improvement of the location of the discontinuity in a manner akin to shock fitting. By minimizing a measure of the jump condition an approximate position of the shock can be maneuvered into an accurate position. This allows the descent least squares method to be used on either side of the shock to gain a good approximation of the smooth regions of the flow.

Acknowledgment. The authors wish to thank Phil Roe of the University of Michigan for many helpful suggestions while visiting the University of Reading under an EPSRC Visiting Fellowship.

REFERENCES

[1] M. J. Baines, *Least-squares and approximate equidistribution in multidimensions*, Numer. Methods Partial Differential Equations, 15 (1999), pp. 605–615.

[2] M. J. Baines and M. E. Hubbard, *Multidimensional upwinding with grid adaptation*, in Numerical Methods for Wave Propagation, E. F. Toro and J. F. Clarke, eds., Kluwer Academic Publishers, Dordrecht, The Netherlands, 1998, pp. 33–54.

[3] M. J. Baines and S. J. Leary, *Fluctuation and signals for scalar hyperbolic equations on adjustable meshes*, Comm. Numer. Methods Engrg., 15 (1999), pp. 877–886.

[4] M. J. Baines, S. J. Leary, and M. E. Hubbard, *A finite volume method for steady hyperbolic equations*, in Finite Volumes for Complex Applications II, R. Vilsmeier, F. Benkhaldoun, and D. Hanel, eds., Hermes, Paris, 1999, pp. 787–794.

[5] M. E. Hubbard, *Multidimensional Upwinding and Grid Adaptation for Conservation Laws*, Ph.D. Thesis, University of Reading, UK, 1996.

[6] M. E. Hubbard and M. J. Baines, *Conservative multidimensional upwinding for the steady two-dimensional shallow water equations*, J. Comput. Phys., 138 (1997), pp. 419–448.

[7] S. J. Leary, *Least Squares Methods with Adjustable Nodes for Steady Hyperbolic PDEs*, Ph.D. Thesis, University of Reading, UK, 1999.

[8] L. M. Mesaros and P. L. Roe, *Multidimensional fluctuation splitting schemes based on decomposition methods*, in Proceedings of the 12th AIAA CFD Conference, San Diego, 1995.

[9] K. Miller and M. J. Baines, *Least squares moving finite elements*, IMA J. Numer. Anal., 21 (2001), pp. 621–642.

[10] P. L. Roe, *Fluctuation and signals: A framework for numerical evolution problems*, in Proceedings of IMA Conference on Numerical Methods for Fluid Dynamics, Reading, UK, 1982, Academic Press, London, 1982, pp. 219–257.

[11] P. L. Roe, *Compounded of many simples*, in Barriers and Challenges in Computational Fluid Dynamics, V. Ventakrishnan, M. D. Salas, and S. R. Chakravarthy, eds., Kluwer Academic Publishers, Dordrecht, The Netherlands, 1998, pp. 241–258.

[12] P. L. Roe, *Fluctuation Splitting on Optimal Grids.* AIAA paper 97-2034, 1997.

[13] R. Struijs, P. L. Roe, and H. Deconinck, *Fluctuation splitting schemes for the 2-D Euler equations*, in Computational Fluid Dynamics 1991-01, VKI Lecture Series, 1991.

[14] F. Taghaddosi, W. G. Habashi, G. Guevremont, and D. Alt-Ali-Yahia, *An Adaptive Least Squares Method for the Compressible Euler Equations*, AIAA paper 97-2097, 1997.

[15] J. Y. Trepanier, M. Paraschivoiu, M. Reggio, and R. Camarero, *A conservative shock fitting method on unstructured grids*, J. Comput. Phys., 126 (1996), pp. 421–433.

[16] J. Van Rosendale, *Floating shock fitting via Lagrangian adaptive meshes*, in Proceedings of the 12th AIAA CFD Conference, San Diego, 1995.

[17] H. Yee, R. F. Warming, and A. Harten, *Implicit total variation diminishing (TVD) schemes for steady state calculations*, J. Comput. Phys., 57 (1985), pp. 327–366.

SIAM J. SCI. COMPUT.
Vol. 23, No. 5, pp. 1503–1525

LEAST SQUARES RESIDUALS AND MINIMAL RESIDUAL METHODS[*]

J. LIESEN[†], M. ROZLOŽNÍK[‡], AND Z. STRAKOŠ[§]

Abstract. We study Krylov subspace methods for solving unsymmetric linear algebraic systems that minimize the norm of the residual at each step (minimal residual (MR) methods). MR methods are often formulated in terms of a sequence of least squares (LS) problems of increasing dimension. We present several basic identities and bounds for the LS residual. These results are interesting in the general context of solving LS problems. When applied to MR methods, they show that the size of the MR residual is strongly related to the conditioning of different bases of the same Krylov subspace. Using different bases is useful in theory because relating convergence to the characteristics of different bases offers new insight into the behavior of MR methods.

Different bases also lead to different implementations which are mathematically equivalent but can differ numerically. Our theoretical results are used for a finite precision analysis of implementations of the GMRES method [Y. Saad and M. H. Schultz, *SIAM J. Sci. Statist. Comput.*, 7 (1986), pp. 856–869]. We explain that the choice of the basis is fundamental for the numerical stability of the implementation. As demonstrated in the case of Simpler GMRES [H. F. Walker and L. Zhou, *Numer. Linear Algebra Appl.*, 1 (1994), pp. 571–581], the best orthogonalization technique used for computing the basis does not compensate for the loss of accuracy due to an inappropriate choice of the basis. In particular, we prove that Simpler GMRES is inherently less numerically stable than the Classical GMRES implementation due to Saad and Schultz [*SIAM J. Sci. Statist. Comput.*, 7 (1986), pp. 856–869].

Key words. linear systems, least squares problems, Krylov subspace methods, minimal residual methods, GMRES, convergence, rounding errors

AMS subject classifications. 65F10, 65F20, 65G50, 15A42

PII. S1064827500377988

1. Introduction.

Consider a linear algebraic system $Ax = b$, where $A \in \mathcal{R}^{N,N}$ is a nonsingular matrix (generally unsymmetric) and $b \in \mathcal{R}^N$. Krylov subspace methods for solving such systems start with an initial approximation x_0, compute the initial residual $r_0 = b - Ax_0$, and then determine a sequence of approximate solutions $x_1, \ldots, x_n, \ldots$ such that x_n belongs to the linear manifold determined by x_0 and the nth Krylov subspace

$$x_n \in x_0 + \mathcal{K}_n(A, r_0), \quad \mathcal{K}_n(A, r_0) \equiv \operatorname{span}\{r_0, Ar_0, \ldots, A^{n-1}r_0\}\,. \tag{1.1}$$

The nth residual then belongs to the manifold given by r_0 and the shifted Krylov subspace (also called the Krylov residual subspace)

$$r_n \equiv b - Ax_n \in r_0 + A\mathcal{K}_n(A, r_0)\,. \tag{1.2}$$

[*] Received by the editors September 12, 2000; accepted for publication (in revised form) July 27, 2001; published electronically January 4, 2002.

http://www.siam.org/journals/sisc/23-5/37798.html

[†] Center for Simulation of Advanced Rockets, University of Illinois at Urbana-Champaign, Urbana, IL 61801 (liesen@uiuc.edu). This author's work was partly supported by the SFB 343, Fakultät für Mathematik, Universität Bielefeld, Germany.

[‡] Institute of Computer Science, Academy of Sciences of the Czech Republic, Prague (miro@cs.cas.cz). This author's work was supported by the GA CR under grant 101/00/1035. Part of the work was performed while this author visited ETH Zürich from 1998–2000 and Emory University in May, 1999.

[§] Department of Mathematics and Computer Science, Emory University, Atlanta, GA 30322 and Institute of Computer Science, Academy of Sciences of the Czech Republic, Prague (strakos@cs.cas.cz). This author's work was supported by the GA AS CR under grant A1030103.

The choice of x_n is based on some particular additional condition. In this paper we focus on the *minimal residual (MR) principle*

$$\|r_n\| = \min_{u \in x_0 + \mathcal{K}_n(A, r_0)} \|b - Au\|, \tag{1.3}$$

which can be equivalently formulated as the *orthogonal projection principle*

$$r_n \perp A\mathcal{K}_n(A, r_0). \tag{1.4}$$

Since A is assumed to be nonsingular, both (1.3) and (1.4) determine the unique sequence of approximate solutions $x_1, \dots, x_n$; see [26]. Mathematically (in exact arithmetic), there are several different methods and many of their algorithmic variants for generating this sequence. Computationally (in finite precision arithmetic), however, different algorithms for computing the same mathematical sequence may produce different results.

We will call the Krylov subspace methods (1.1) generating mathematically the approximate solutions $x_1, \dots, x_n$ uniquely determined by the MR principle (1.3) (or by the equivalent orthogonal projection principle (1.4)) *MR Krylov subspace methods* (MR methods).

The MR principle (1.3) represents a least squares (LS) problem, and thus MR methods are often described as a sequence of LS least problems of increasing dimension [26]. In this paper we use general results about LS residuals to analyze the properties of different implementations of MR methods in exact as well as finite precision arithmetic. Our approach is as follows.

In section 2 we present several basic identities and bounds for the norm of the residual $r = c - By$ of the overdetermined LS problem $Bu \approx c$. Specifically, our results relate $\|r\|$ to the singular values of the matrix $[c\gamma, B]$, where $\gamma > 0$ is a scaling parameter, and occasionally some other data. Results of this type have been presented in the literature before (see, e.g., [29]), and they are of importance in studying LS problems in general. While our main focus is on MR methods, only a part of our general LS results are used later in the paper. We believe, however, that the presented LS results which are not directly applied here might be found useful in the context of MR methods in the future.

In section 3 we apply results from section 2 to MR methods for the problem $Ax = b$. In particular, we relate the norm of the MR residual to the conditioning of different bases of $\mathcal{K}_n(A, r_0)$. We derive several necessary and sufficient conditions for fast convergence as well as for stagnation of MR methods. Our results are significantly stronger and more complete than the corresponding results published previously [16, 17]. We point out that our results should not be interpreted as bounds for measuring convergence. As demonstrated in the further sections, results relating residual norm to the conditioning of different bases lead to a new understanding of MR methods.

Section 4 describes the main examples of the MR methods, in particular various forms of the GMRES method [26]. We then apply our theoretical results about the MR residual to finite precision analysis of the important implementations in section 5. Our results explain why the choice of the basis is fundamental for the numerical stability of the implementation. As demonstrated on the example of Simpler GMRES [34], which constructs in exact arithmetic an orthonormal basis of $A\mathcal{K}_n(A, r_0)$, the best orthogonalization technique (Householder reflections) in computing the basis does not compensate for the loss of accuracy due to the inappropriate choice of the basis.

Simpler GMRES is proved inherently less stable than the Classical GMRES implementation [26], which constructs in exact arithmetic an orthonormal basis of $\mathcal{K}_n(A, r_0)$. Our findings are illustrated by numerical experiments.

We denote by $\sigma_i(\cdot)$ the ith largest singular value and by $\sigma_{min}(\cdot)$ the smallest singular value of a given matrix. By $\kappa(\cdot)$ we denote the ratio of the largest to the smallest singular value (condition number). We use $\|\cdot\|$ to denote the 2-norm, e_i to denote the ith vector of the standard Euclidean basis, and I to denote the identity matrix.

2. Basic relations for the LS residual.

As the MR methods can be expressed as sequences of LS problems, it will prove useful to collect some basic relations for the LS residual. We will recall some known results, prove several new results, and put the ones known previously in a new context. Most of the results of this section will be used in our analysis of MR methods later in the paper. We believe that they are also of interest in the LS context in general.

Consider an overdetermined linear approximation problem

$$Bu \approx c, \quad B \in \mathcal{R}^{N,n},\ c \in \mathcal{R}^N,\ n < N,\ \text{rank}(B) = n. \tag{2.1}$$

We denote by y the LS solution of (2.1) and by $r = c - By$ the corresponding LS residual,

$$\|r\| \ = \ \|c - By\| \ = \ \min_u \|c - Bu\| \,. \tag{2.2}$$

We introduce a real scaling parameter $\gamma > 0$ and consider a scaled version of (2.1),

$$Bz \approx c\gamma, \quad B \in \mathcal{R}^{N,n},\ c \in \mathcal{R}^N,\ n < N,\ \text{rank}(B) = n. \tag{2.3}$$

Note that if the right-hand side c is replaced in (2.1) and (2.2) by the scaled vector $c\gamma$, the LS solution and the LS residual scale trivially to $z = y\gamma$ and $r\gamma$. We start with general identities relating r to the matrix $[c\gamma, B]$.

THEOREM 2.1. *Suppose that $[c, B] \in \mathcal{R}^{N,n+1}$ has full column rank, and $r \neq 0$ is the residual of the LS problem* (2.1)–(2.2). *Let $\gamma > 0$ be a real parameter. Then*

$$e_1^T\,[c\gamma, B]^\dagger = \frac{r^T}{\gamma\|r\|^2} \qquad \textit{and} \qquad \gamma\|r\| \ = \frac{1}{\{e_1^T([c\gamma, B]^T[c\gamma, B])^{-1}e_1\}^{\frac{1}{2}}}, \tag{2.4}$$

where $X^\dagger$ denotes the Moore–Penrose generalized inverse of a matrix X.

Proof. For any matrix X the Moore–Penrose pseudoinverse $X^\dagger$ satisfies $XX^\dagger X = X$ (see, e.g., [5]), which using the symmetry of $XX^\dagger$ gives $X^T = X^T XX^\dagger$. Substituting $X = [c\gamma, B]$, we receive the following simple identities:

$$\begin{aligned} \gamma r^T &= [1, -\gamma y^T]\,[c\gamma, B]^T = [1, -\gamma y^T]\,[c\gamma, B]^T\,[c\gamma, B][c\gamma, B]^\dagger \\ &= \gamma r^T\,[c\gamma, B][c\gamma, B]^\dagger\,. \end{aligned}$$

Since r is orthogonal to the columns of B, $\gamma r^T\,[c\gamma, B] \ = \ \gamma^2(r^Tc)\,e_1^T \ = \ \gamma^2\|r\|^2\,e_1^T$, which proves the first part of the theorem. The second part follows from the identity $\|e_1^T\,[c\gamma, B]^\dagger\|^2 = e_1^T([c\gamma, B]^T[c\gamma, B])^{-1}e_1$, which can be verified by a straightforward calculation. □

The first equality in (2.4) was essentially proven (though neither the statement nor the proof were formulated explicitly in the form presented here) in [28, relations

(2.5), (2.6), (3.7), and (3.8)]. Later it was presented (with $\gamma = 1$) in [16, Lemma 7.1] (see also other references therein).

It is important to notice that for an arbitrary nonsingular matrix $M \in \mathcal{R}^{n,n}$,

$$\|r\| \;=\; \|c - By\| \;=\; \|c - (BM)(M^{-1}y)\| \;=\; \min_u \|c - (BM)u\|\,.$$

As a consequence of this simple observation, (2.4) will hold when B is replaced by BM. A particularly useful choice is $M = R^{-1}$, where R is the upper triangular factor of a QR-factorization of B.

COROLLARY 2.2. *Using the assumptions and the notation of Theorem* 2.1, *and a QR-factorization $B = QR$ of the matrix B,*

$$e_1^T\,[c\gamma, Q]^\dagger = \frac{r^T}{\gamma\|r\|^2} \qquad \textit{and} \qquad \gamma\|r\| \;=\; \frac{1}{\{e_1^T([c\gamma,Q]^T[c\gamma,Q])^{-1}e_1\}^{\frac{1}{2}}}\,. \tag{2.5}$$

It may look a bit surprising that the first rows of the matrices $[c\gamma, B]^\dagger$ and $[c\gamma, Q]^\dagger$ are identical. A second look reveals that this fact is simple and natural.

Consider a full column rank matrix $X = [c\gamma, B] \in \mathcal{R}^{N,n+1}$. Then the rows of $X^\dagger$ are linear combinations of the rows of X^T (the transposed columns of X), and $X^\dagger X = I$. The last relation can be interpreted geometrically as an orthogonal relation between the rows of $X^\dagger$ and the columns of X. Denote by $s = e_1^T X^\dagger$ the first row of $X^\dagger$. Then s is orthogonal to all but the first column of X; i.e., it is orthogonal to the columns of the matrix B. Because s represents a linear combination of c^T and the transposed columns of B, it must be equal to a scalar multiple of the transposed residual $r^T = (c - By)^T$ for the LS problem (2.1)–(2.2). The identity $(\zeta r^T)\,(c\gamma) = 1$ then immediately gives $\zeta = \gamma^{-1}\|r\|^{-2}$.

The orthogonality idea clearly applies with no change when B is replaced by any matrix BM, where $M \in \mathcal{R}^{n,n}$ is nonsingular. The geometrical interpretation of the generalized inverse is simple but revealing.

The following theorem relates the norm of the LS residual (2.2) to the singular values of the matrices B, $[c\gamma, B]$, and $[c\gamma, Q]$. This theorem plays a substantial role in our further analysis.

THEOREM 2.3. *Suppose that $[c, B] \in \mathcal{R}^{N,n+1}$ has full column rank, and $r \neq 0$ is the residual of the LS problem* (2.1)–(2.2). *Let $B = QR$ be a QR-factorization of the matrix B and $\gamma > 0$ be a real parameter. Then*

$$\|r\| \;=\; \frac{\sigma_{min}([c\gamma,B])}{\gamma}\prod_{j=1}^{n}\frac{\sigma_j([c\gamma,B])}{\sigma_j(B)} \tag{2.6}$$

$$\;=\; \frac{1}{\gamma}\,\sigma_{min}([c\gamma,Q])\,\sigma_1([c\gamma,Q]). \tag{2.7}$$

Furthermore,

$$\kappa([c\gamma,Q]) \;=\; \frac{\alpha + \left(\alpha^2 - 4\gamma^2\|r\|^2\right)^{1/2}}{2\gamma\,\|r\|}\,, \quad \|r\| \;=\; \frac{\alpha}{\gamma}\,\frac{\kappa([c\gamma,\,Q])}{\kappa([c\gamma,\,Q])^2+1}, \tag{2.8}$$

where $\alpha \equiv 1 + \gamma^2\|c\|^2$.

Proof. Using the orthogonality of the columns of the matrix Q, the right-hand side c and the residual r are related by the identity

$$c = Qh + r, \quad h \equiv Q^Tc, \quad \|c\|^2 = \|h\|^2 + \|r\|^2\,.$$

Now consider an orthogonal matrix $U \in \mathcal{R}^{n,n}, U^T U = I$, such that $Uh = \|h\| e_1$. Then

$$[c\gamma, Q]^T [c\gamma, Q] = \begin{bmatrix} 1 & 0 \\ 0 & U^T \end{bmatrix} \begin{bmatrix} \gamma^2\|c\|^2 & \gamma\|h\|e_1^T \\ \gamma\|h\|e_1 & I \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & U \end{bmatrix}, \tag{2.9}$$

$$[c\gamma, B]^T [c\gamma, B] = \begin{bmatrix} 1 & 0 \\ 0 & R^T \end{bmatrix} [c\gamma, Q]^T [c\gamma, Q] \begin{bmatrix} 1 & 0 \\ 0 & R \end{bmatrix}. \tag{2.10}$$

Identity (2.9) shows that all but two of the eigenvalues of $[c\gamma, Q]^T[c\gamma, Q]$ are equal to one. The two remaining eigenvalues are easily determined as the eigenvalues of the left principal two-by-two block,

$$\sigma_1^2([c\gamma, Q]) = \frac{\alpha + (\alpha^2 - 4\gamma^2\|r\|^2)^{1/2}}{2}, \tag{2.11}$$

$$\sigma_{min}^2([c\gamma, Q]) = \frac{\alpha - (\alpha^2 - 4\gamma^2\|r\|^2)^{1/2}}{2}, \tag{2.12}$$

where $\alpha \equiv 1 + \gamma^2\|c\|^2$. (Notice that $\alpha^2 - 4\gamma^2\|r\|^2 \geq (1 - \gamma^2\|c\|)^2 \geq 0$.) Using

$$\kappa([c\gamma, Q]) = \sigma_1([c\gamma, Q])/\sigma_{min}([c\gamma, Q]),$$

(2.8) is obtained by a simple algebraic manipulation.

Evaluating the determinants on both sides of (2.9) yields

$$\det([c\gamma, Q]^T[c\gamma, Q]) = \sigma_1^2([c\gamma, Q])\, \sigma_{min}^2([c\gamma, Q]) = \gamma^2\|r\|^2,$$

which shows (2.7). Similarly, transformation (2.10) yields

$$\begin{aligned} \det([c\gamma, B]^T[c\gamma, B]) &= \prod_{j=1}^{n+1} \sigma_j^2([c\gamma, B]) \\ &= \det([c\gamma, Q]^T[c\gamma, Q]) \det(R^T R) = \gamma^2\|r\|^2 \prod_{j=1}^{n} \sigma_j^2(B), \end{aligned}$$

which proves (2.6). □

The relations (2.8) generalize results presented in [19, section 5.5.2]. The identity (2.6) (with $\gamma = 1$) was first shown by Van Huffel and Vandewalle [29, Theorem 6.9], and it also appeared (with a different proof) in [20].

Van Huffel and Vandewalle [29, Theorem 6.10] gave the following lower and upper bounds for $\|r\|$ (with $\gamma = 1$) in terms of $\sigma_{min}([c\gamma, B])$ (see also [20]). Let

$$\delta(\gamma) \equiv \sigma_{min}([c\gamma, B])/\sigma_{min}(B). \tag{2.13}$$

Then

$$\frac{\sigma_{min}([c\gamma, B])}{\gamma} \leq \|r\| \leq \frac{\sigma_{min}([c\gamma, B])}{\gamma} \left\{1 - \delta(\gamma)^2 + \frac{\gamma^2\|c\|^2}{\sigma_{min}^2(B)}\right\}^{\frac{1}{2}}. \tag{2.14}$$

Bounds for $\|r\|$ in terms of the minimal singular values of B and $[c\gamma, B]$, and as little additional information as possible, were studied in detail in [20]. In particular, when B has full column rank and

$$c \not\perp \{\text{left singular vector subspace of } B \text{ corresponding to } \sigma_{min}(B)\}, \tag{2.15}$$

then the following bounds were given in [20]:

$$\begin{aligned} \sigma_{min}([c\gamma, B])\,\{\gamma^{-2} + \|y\|^2\}^{\frac{1}{2}} \le \|r\| \\ \le \sigma_{min}([c\gamma, B])\left\{\gamma^{-2} + \frac{\|y\|^2}{1 - \delta(\gamma)^2}\right\}^{\frac{1}{2}}. \end{aligned} \tag{2.16}$$

Though (2.14) can be derived from (2.16) (and not vice versa; see [20]), the upper bound in (2.16) is not always tighter than the upper bound in (2.14). When $\delta(\gamma) \approx 1$ and $\|r\| \approx \|c\|$, it is possible for the upper bound in (2.14) to be smaller than that in (2.16). However, in this case the upper bound in (2.14) becomes trivial. For details, see [20].

For $\delta(\gamma) = 1$ the upper bound in (2.16) does not exist. It was shown in [22] that if (2.15) holds, then $\delta(\gamma) < 1$ for all $\gamma > 0$. As explained in [22], the role of the assumption (2.15) is truly fundamental. If it does not hold, both theory and computation in errors-in-variables modeling are enormously complicated by the possible case $\delta(\gamma) = 1$. Fortunately, nearly all practical problems will satisfy (2.15). Nevertheless, it is instructive to consider possible cases where (2.15) does not hold, so that $\delta(\gamma) = 1$ is possible.

The lower bound in (2.14) shows that we can make $\sigma_{min}([c\gamma, B])$ arbitrarily small by taking γ small and thus ensure $\delta(\gamma) < 1$ in (2.13). How small must γ be to ensure this? The next theorem answers a variant of this question. Given $\sigma_{min}(B)$ and $\|c\|$, it shows that there is a γ_0 such that $\gamma < \gamma_0$ ensures $\delta(\gamma) < 1$, but $\gamma = \gamma_0$ does not.

THEOREM 2.4. *Suppose that* $[c, B] \in \mathcal{R}^{N,n+1}$ *has full column rank, y is the solution, and $r \neq 0$ is the residual of the LS problem* (2.1)–(2.2). *Let $\gamma > 0$ be a real parameter, and $\delta(\gamma) \equiv \sigma_{min}([c\gamma, B])/\sigma_{min}(B)$. Define $\gamma_0 \equiv \sigma_{min}(B)/\|c\|$. Then*

$$\delta(\gamma) < 1 \;\; \textit{for all} \;\; \gamma < \gamma_0. \tag{2.17}$$

Moreover,

$$y = 0 \;\; (r = c) \quad \textit{if and only if} \quad \delta(\gamma_0) = 1. \tag{2.18}$$

Proof. Note that when $\gamma < \gamma_0$, then $\|c\gamma\| < \sigma_{min}(B)$. Therefore $\sigma_{min}([c\gamma, B]) < \sigma_{min}(B)$, i.e., $\delta(\gamma) < 1$.

Now assume that the LS problem (2.1)–(2.2) has the trivial solution $y = 0$ $(r = c)$. Then $B^T c = 0$, which yields

$$[c\gamma, B]^T [c\gamma, B] = \begin{bmatrix} \|c\|^2\gamma^2 & 0 \\ 0 & B^T B \end{bmatrix}.$$

Thus, $\sigma_{min}([c\gamma, B]) = \min\{\|c\|\gamma,\ \sigma_{min}(B)\}$, $\delta(\gamma) = \min\{\|c\|\gamma/\sigma_{min}(B),\ 1\}$, and $\delta(\gamma_0) = 1$. Conversely, (2.14) gives with $\gamma = \gamma_0$,

$$\delta(\gamma_0)\,\|c\| \;\le\; \|r\| \;\le\; \delta(\gamma_0)\,\|c\|\,\{2 - \delta(\gamma_0)^2\}^{1/2}, \tag{2.19}$$

which for $\delta(\gamma_0) = 1$ reduces to $\|c\| \le \|r\| \le \|c\|$, i.e., $\|r\| = \|c\|$, which completes the proof. □

We see that $\gamma_0 \equiv \sigma_{min}(B)/\|c\|$ represents an important number. For $\gamma < \gamma_0$ the value of $\delta(\gamma)$ is always strictly less than one, and $\delta(\gamma_0) = 1$ if and only if the LS problem (2.1)–(2.2) has the trivial solution $y = 0$. Moreover, (2.19) shows that $\|r\|$ is significantly smaller than $\|c\|$ if and only if $\delta(\gamma_0)$ is significantly smaller than one. As an application, we will show in section 3 how these results characterize stagnation or near stagnation of MR methods.

One consequence of Theorem 2.4 can be stated as follows: Consider a rectangular matrix (here B) having full column rank and an additional column (here $c\gamma$). If the norm of the additional column is smaller than the smallest singular value of the matrix (here if $\gamma < \gamma_0$), then appending the column necessarily decreases the smallest singular value. If the norm of the appended column is equal to the smallest singular value of the matrix (here if $\gamma = \gamma_0$), then appending the column to the matrix does not change the smallest singular value if and only if the appended column is orthogonal to all the columns (all the left singular vectors) of the original matrix. This is a somewhat specialized result because of the norm of the added column fixed to $\sigma_{min}(B)$. Note that the condition is linear. The general necessary and sufficient condition under which adding a column (with a norm larger or equal to $\sigma_{min}(B)$) to a matrix B does not alter the smallest singular value was given in [22]. The added column must be orthogonal to the left singular vector subspace of B corresponding to $\sigma_{min}(B)$, and the left-hand side of the (deflated) secular equation [22, relation (3.4)] must be nonnegative at $\sigma_{min}(B)$. Theorem 2.4 can also be derived from this. The second part of the condition from [22] is nonlinear.

Theorem 2.4 and the consequence stated above must be understood in their proper context. It was pointed out in [22] that nearly all practical problems will satisfy (2.15), that any problem $Bu \approx c$ can be reduced to a core problem satisfying (2.15), and that for many formulations it makes sense only to consider problems satisfying (2.15). Also, if the problem satisfies (2.15), then $\delta(\gamma) < 1$ for all $\gamma > 0$, and in this case (2.17) and (2.18) are irrelevant. On the other hand, (2.19) seems to be a generally useful result. Thus $\gamma_0 \equiv \sigma_{min}(B)/\|c\|$ is a significant quantity, as can be seen from the interesting but rarely practical properties (2.17) and (2.18), and the interesting and compact bounds (2.19). Note also that in many practical problems of interest, $\gamma_0 \equiv \sigma_{min}(B)/\|c\|$ will be a *very* small number. In particular, this suggests that for a general LS problem the above "column addition" result will be of minor practical use. It is, however, important theoretically because it offers a new insight into the stagnation or near stagnation of the MR methods.

Finally, for completeness, consider a QR-factorization $B = QR$. Replacing B by $BR^{-1} = Q$ and y by Ry (notice that $\|Ry\| = \|By\|$) gives the analogies of (2.14) and (2.16),

$$\frac{\sigma_{min}([c\gamma, Q])}{\gamma} \le \|r\| \le \frac{\sigma_{min}([c\gamma, Q])}{\gamma} \{1 - \sigma^2_{min}([c\gamma, Q]) + \gamma^2\|c\|^2\}^{\frac{1}{2}}, \tag{2.20}$$

$$\begin{aligned} \sigma_{min}([c\gamma, Q])\,\{\gamma^{-2} + \|By\|^2\}^{\frac{1}{2}} &\le \|r\| \\ &\le \sigma_{min}([c\gamma, Q]) \left\{\gamma^{-2} + \frac{\|By\|^2}{1 - \sigma_{min}([c\gamma, Q])^2}\right\}^{\frac{1}{2}}. \end{aligned} \tag{2.21}$$

Theorem 2.4 can be reformulated in a similar way. It is interesting to note that the bounds (2.20) do not give additional information. Indeed, since $\sigma_1([c\gamma, Q]) \ge 1$, the lower bound in (2.20) follows immediately from (2.7). And since $\{1 - \sigma^2_{min}([c\gamma, Q]) + \gamma^2\|c\|^2\}^{1/2} = \sigma_1([c\gamma, Q])$, the upper bound is a weak reformulation of (2.7) only.

In the following section we apply results of this section to the MR Krylov subspace methods.

3. Characteristics of the basis and the size of the MR residual.

Let $\rho_0 \equiv \|r_0\|$, $v_1 \equiv r_0/\rho_0$, $w_1 \equiv Av_1/\|Av_1\|$. Consider two sequences of orthonormal vectors, $v_1, v_2, \dots$ and $w_1, w_2, \dots$, such that for each iterative step n,

$$\mathcal{K}_n(A, r_0) \;=\; \text{span}\{v_1, \dots, v_n\}, \;\; V_n \equiv [v_1, \dots, v_n], \;\; V_n^T V_n = I, \tag{3.1}$$

$$A\mathcal{K}_n(A, r_0) \;=\; \text{span}\{w_1, \dots, w_n\}, \;\; W_n \equiv [w_1, \dots, w_n], \;\; W_n^T W_n = I. \tag{3.2}$$

Then the MR principle (1.3) can be formulated as

$$\|r_n\| \;=\; \min_{u \in \mathcal{R}^n} \|r_0 - AV_n u\| \tag{3.3}$$

$$\;=\; \min_{u \in \mathcal{R}^n} \|r_0 - W_n u\|\,. \tag{3.4}$$

The MR residual at step n is therefore the LS residual for the LS problems $AV_n u \approx v_1\rho_0$ and $W_n u \approx v_1\rho_0$.

The application of the results presented in section 2 to (3.3) and (3.4) is straightforward: For the nth step of an MR method we consider $c \equiv r_0 = v_1\rho_0$, $B \equiv AV_n$, $Q \equiv W_n$, and $r \equiv r_n$. The scaling parameter $\gamma > 0$ offers some flexibility. While it seems natural to use $\gamma \equiv \|r_0\|^{-1} = \rho_0^{-1}$, other values of γ also prove useful; cf. [21] and our discussion below.

With $\gamma \equiv \|r_0\|^{-1} = \rho_0^{-1}$, Theorem 2.3 and relations (2.7) and (2.8) give the following identities for the relative residual norm $\|r_n\|/\rho_0$:

$$\|r_n\|/\rho_0 \;=\; \sigma_{min}([v_1,\, W_n])\, \sigma_1([v_1,\, W_n]) \tag{3.5}$$

$$\;=\; \frac{2\,\kappa([v_1,\, W_n])}{\kappa([v_1,\, W_n])^2 + 1}. \tag{3.6}$$

Identities (3.5) and (3.6) show that the conditioning of the basis $[v_1,\, W_n]$ of the Krylov subspace $\mathcal{K}_{n+1}(A, r_0)$ is fully determined (except for an unimportant multiplicative factor) by the convergence of the MR methods, and vice versa. In other words,

$$\|r_n\| = \rho_0 \quad \text{if and only if} \quad \kappa([v_1,\, W_n]) = 1, \tag{3.7}$$

and the relative residual norm $\|r_n\|/\rho_0$ is small if and only if $[v_1,\, W_n]$ is ill-conditioned.

The previous statement can also be mathematically expressed by inequalities. Dividing both the numerator and the denominator in (3.6) by $\kappa([v_1, W_n])$ gives in a simple way the bounds

$$\kappa([v_1, W_n])^{-1} \;\le\; \|r_n\|/\rho_0 \;\le\; 2\,\kappa([v_1, W_n])^{-1}. \tag{3.8}$$

The upper bound in (3.8) was published by Walker and Zhou [34, Lemma 3.1]. It is interesting to note that, because of (2.11),

$$1 \;\le\; \sigma_1([v_1, W_n]) \;\le\; \sqrt{2}, \tag{3.9}$$

which shows that the size of $\kappa([v_1, W_n])$ is in fact determined by the smallest singular value $\sigma_{min}([v_1, W_n])$.

Relations between the size of the residuals of the MR methods and the condition number of matrices $[v_1, W_n]$ and $[r_0, W_n]$ were studied in [19, section 5.5.2]. We will generalize the result [19, relation (5.48)] and develop an elegant tool for quantification of the influence of the scaling parameter γ.

THEOREM 3.1. *Let r_0, r_n, and W_n be as in (3.4), $\rho_0 \equiv \|r_0\|, v_1 \equiv r_0/\rho_0$, and $\gamma > 0$. Then*

$$\kappa([r_0\gamma, W_n]) \;\geq\; \kappa([v_1, W_n]) \,+\, \frac{\gamma\,(\rho_0 - \gamma^{-1})^2}{2\|r_n\|}. \tag{3.10}$$

Proof. Using (2.8) with the particular choices $c \equiv r_0$, $Q \equiv W_n$, $\gamma > 0$, and $c \equiv r_0$, $Q \equiv W_n$, $\gamma \equiv \rho_0^{-1} = \|r_0\|^{-1}$ gives

$$\begin{aligned}
&\kappa([r_0\gamma, W_n]) - \kappa([v_1, W_n]) \\
&\quad = \frac{1 + \gamma^2\rho_0^2 + [(1+\gamma^2\rho_0^2)^2 - 4\gamma^2\|r_n\|^2]^{1/2}}{2\gamma\|r_n\|} - \frac{2 + [4 - 4\rho_0^{-2}\|r_n\|^2]^{1/2}}{2\rho_0^{-1}\|r_n\|} \\
&\quad = \frac{\gamma^{-1} + \gamma\rho_0^2 + [(\gamma^{-1}+\gamma\rho_0^2)^2 - 4\|r_n\|^2]^{1/2} - 2\rho_0 - [4\rho_0^2 - 4\|r_n\|^2]^{1/2}}{2\|r_n\|} \\
&\quad = \frac{\gamma\,(\rho_0 - \gamma^{-1})^2}{2\|r_n\|} + \frac{[(\gamma^{-1}+\gamma\rho_0^2)^2 - 4\|r_n\|^2]^{1/2} - [4\rho_0^2 - 4\|r_n\|^2]^{1/2}}{2\|r_n\|} \\
&\quad \geq \frac{\gamma\,(\rho_0 - \gamma^{-1})^2}{2\|r_n\|}. \qquad \square
\end{aligned}$$

Clearly, $\kappa([r_0\gamma, W_n])$ is minimal for $\gamma = \rho_0^{-1}$, and the minimum is equal to $\kappa([v_1, W_n])$ (see also [8]). If $\gamma \neq \rho_0^{-1}$, then with the residual norm $\|r_n\|$ decreasing towards zero the condition number $\kappa([r_0\gamma, W_n])$ grows much faster than $\kappa([v_1, W_n])$. The results considering the matrix $[r_0\gamma, W_n]$ will be particularly useful for our discussion of MR implementations based on the orthogonal projection principle (1.4) in section 5.

With $c \equiv r_0$, $r \equiv r_n$, $y \equiv y_n$, and $B \equiv AV_n$, (2.16) gives the following bounds for the residual norm in terms of $\sigma_{min}([r_0\gamma, AV_n])$:

$$\begin{aligned}
\sigma_{min}([r_0\gamma, AV_n])\,\{\gamma^{-2} + \|y_n\|^2\}^{\frac{1}{2}} \;&\leq\; \|r_n\| \\
&\leq\; \sigma_{min}([r_0\gamma, AV_n])\left\{\gamma^{-2} + \frac{\|y_n\|^2}{1 - \delta_n(\gamma)^2}\right\}^{\frac{1}{2}},
\end{aligned} \tag{3.11}$$

where $\delta_n(\gamma) \equiv \sigma_{min}([r_0\gamma, AV_n])/\sigma_{min}(AV_n)$. As mentioned in section 2, the upper bound in (3.11) becomes intriguing for $\delta_n(\gamma) \approx 1$, and for $\delta_n(\gamma) = 1$ it is not defined.

The convergence of the MR methods and the situation $\delta_n(\gamma) = 1$ or $\delta_n(\gamma) \approx 1$ are related by Theorem 2.4. Define $\gamma_0^{(n)} \equiv \sigma_{min}(AV_n)/\rho_0$. Then $\delta_n(\gamma) < 1$ for all $\gamma < \gamma_0^{(n)}$ and

$$\begin{aligned}
&\|r_n\| = \rho_0 \;\;\Leftrightarrow \\
&\delta_n(\gamma_0^{(n)}) \equiv \frac{\sigma_{min}([v_1\sigma_{min}(AV_n), AV_n])}{\sigma_{min}(AV_n)} = \sigma_{min}([v_1, AV_n/\sigma_{min}(AV_n)]) = 1.
\end{aligned} \tag{3.12}$$

Moreover, (2.19) gives

$$\delta_n(\gamma_0^{(n)}) \;\leq\; \|r_n\|/\rho_0 \;\leq\; \sqrt{2}\,\delta_n(\gamma_0^{(n)}), \tag{3.13}$$

which shows that the rate of convergence of the MR methods is determined by the size of $\delta_n(\gamma_0^{(n)})$. Summarizing, the MR methods stagnate in steps 1 through n if and only if $\delta_n(\gamma_0^{(n)}) = 1$, and they nearly stagnate in steps 1 through n if and only if $\delta_n(\gamma_0^{(n)}) \approx 1$. However, this specific link between convergence of the MR methods and the value of $\delta_n(\gamma)$ can be made for $\gamma = \gamma_0^{(n)}$ only. In particular, when $\delta_n(\gamma_1) = 1$ for some $\gamma_1 > \gamma_0^{(n)}$, the MR methods do not necessarily stagnate or nearly stagnate. They may exhibit very fast convergence while $\delta_n(\gamma_1) \approx 1$ and very slow convergence while $\delta_n(\gamma_1) \ll 1$. (For more details, see [21].)

For $\gamma = \gamma_0^{(n)}$ there is an interesting relationship between the smallest singular values of the matrices $[v_1, AV_n/\sigma_{min}(AV_n)]$ and $[v_1, W_n]$: (3.5), (3.9), and (3.13) yield

$$\sigma_{min}([v_1, AV_n/\sigma_{min}(AV_n)]) \leq \sqrt{2}\,\sigma_{min}([v_1, W_n]) \leq 2\,\sigma_{min}([v_1, AV_n/\sigma_{min}(AV_n)]),$$

which shows that these smallest singular values are very close to each other.

Using the matrix $[r_0\gamma, AV_n]$ instead of $[v_1, W_n]$ may seem unwise because it necessarily brings into play the potentially ill-conditioned matrix AV_n (in comparison to W_n having orthonormal columns). However, as shown in [22, 21], bounds using the matrix $[r_0\gamma, AV_n]$ are very useful for the analysis of the modified Gram–Schmidt implementation of Classical GMRES. Notice that the bounds (3.11) are not based on singular values only. Using $\|y_n\|$, the norm of the MR approximate solution, makes (3.11) amazingly tight [22]. The parameter γ offers flexibility required for the analysis of the GMRES method [21].

It is also possible to consider other bases of the Krylov subspaces or Krylov residual subspaces which lead to other matrices, identities, and bounds. For example, Ipsen [16, 17] used the matrix $K_{n+1} = [r_0, Ar_0, \ldots, A^n r_0]$, got the identity

$$\|r_n\| = 1/\|e_1^T\, K_{n+1}^\dagger\| \tag{3.14}$$

(cf. Theorem 2.1), and developed the bound $\|r_n\|/\rho_0 \geq 1/(\|K_{n+1}\|\,\|K_{n+1}^\dagger\|)$. However, any bound based directly on the matrix K_{n+1} necessarily suffers from the potential ill-conditioning of the matrix $[Ar_0, \ldots, A^n r_0]$. Consider the QR-decomposition $[Ar_0, \ldots, A^n r_0] = W_n R_n$. In light of the results presented above (see, in particular, (2.5), (3.5), and (3.6)), the upper triangular factor R_n containing *all* the potential ill-conditioning of the matrix $[Ar_0, \ldots, A^n r_0]$ is mathematically in no relation whatsoever to the residual r_n and to the convergence of any MR method measured by the residual norm. Except for some (rather special) examples, bounds based on the matrix K_{n+1} are therefore necessarily much weaker than the bounds based on the matrices $[r_0\gamma, W_n]$ and $[r_0\gamma, AV_n]$.

In the following we use our theoretical results to obtain new insight into the numerical behavior of MR methods.

4. Implementations of the MR methods. Numerous residual norm minimizing Krylov subspace methods have been proposed in the last decades [18, 30, 35, 1, 11, 26]. Resulting from different approaches, they generate (under different assumptions) approximate solutions satisfying (1.3) and (1.4). Though they are, under some particular assumptions, mathematically equivalent, they differ in various algorithmic aspects, and, consequently, in their numerical behavior.

We will concentrate on two main approaches which explicitly compute the basis vectors $v_1, v_2, \ldots, v_n$ (respectively, $v_1, w_1, \ldots, w_{n-1}$) defined in (3.1) and (3.2). In the first approach, the approximate solution x_n is expressed as

$$x_n \;=\; x_0 \;+\; V_n\, y_n$$

for some y_n, and the residual norm is bounded in terms of $\sigma_{min}([v_1\rho_0\gamma, AV_n])$ via (3.11). In the second approach the approximate solution is expressed as

$$x_n \;=\; x_0 \;+\; [v_1, W_{n-1}]\, t_n$$

for some t_n, and for the residual norm we have the identities (3.5)–(3.6). At first sight the second approach seems more attractive because it gives a cleaner relation between the residual norm (which is minimized at every step) and the conditioning of the computed basis. Its implementation is also simpler. On the other hand, the fact that the approximate solution is in this approach determined via the basis vectors $v_1, w_1, \ldots, w_{n-1}$ which *are not mutually orthogonal* raises some suspicions about potential numerical problems. In this section we recall implementations of both approaches resulting in different variants of the GMRES algorithm. In section 5 we will discuss their numerical properties.

A variety of MR methods that do not explicitly compute the vectors $v_1, v_2, \ldots, v_n$ or $v_1, w_1, \ldots, w_{n-1}$ have been proposed. For example, the method by Khabaza [18] uses the vectors $r_0, Ar_0, \ldots, A^{n-1}r_0$; Orthomin [30], Orthodir [35], Generalized Conjugate Gradient (GCG) [1, 2] and Generalized Conjugate Residual (GCR) [10, 11] compute an A^TA-orthogonal basis of $\mathcal{K}_n(A, r_0)$. These methods played an important role in the development of the field. In comparison to the approaches discussed in this paper they are, however, less numerically stable. Therefore we will not consider them below.

4.1. Minimal residual principle: Classical GMRES. Consider an initial approximation x_0 and the initial residual $r_0 = b - Ax_0$, $\rho_0 \equiv \|r_0\|$. In their classical paper [26], Saad and Schultz used the orthonormal basis (3.1) (Arnoldi basis). As noted in [33], this basis can be mathematically expressed as the Q-factor of a recursive columnwise QR-factorization

$$[r_0, AV_n] = V_{n+1}\,[e_1\rho_0, H_{n+1,n}], \qquad V_{n+1} \equiv [v_1, \ldots, v_{n+1}], \qquad V_{n+1}^T V_{n+1} = I. \tag{4.1}$$

Here $H_{n+1,n}$ is an $(n+1)$-by-n upper Hessenberg matrix with elements $h_{i,j}$, $h_{j+1,j} \neq 0, j = 1, 2, \ldots, n-1$. If at any stage $h_{n+1,n} = 0$, the algorithm would stop with $[r_0, AV_n] = V_n\,[e_1\rho_0, H_{n,n}]$. Using the substitution

$$x_n \;=\; x_0 \;+\; V_n\, y_n \tag{4.2}$$

and (4.1), the MR principle (1.3) gives the LS problem for the vector of coefficients y_n:

$$\|r_n\| \equiv \|b - Ax_n\| = \min_{y\in\mathcal{R}^n} \|r_0 - AV_n\, y\| \;=\; \min_{y\in\mathcal{R}^n} \|V_{n+1}\,(e_1\rho_0 - H_{n+1,n}\, y)\| \tag{4.3}$$

$$= \min_{y\in\mathcal{R}^n} \|e_1\rho_0 - H_{n+1,n}\, y\|. \tag{4.4}$$

To solve (4.3) we apply orthogonal rotations $J_1, J_2, \ldots, J_n$ sequentially to $H_{n+1,n}$ to bring it to the upper triangular form T_n:

$$J_n \cdots J_2 J_1\, H_{n+1,n} \;=\; \begin{bmatrix} T_n \\ 0 \end{bmatrix}.$$

The vectors y_n and $\|r_n\|$ then satisfy

$$\begin{bmatrix} T_n\, y_n \\ \|r_n\| \end{bmatrix} \;=\; J_1^T J_2^T \cdots J_n^T\, e_1 \rho_0. \tag{4.5}$$

The value of the (nonincreasing) residual norm is available without determining y_n, and it can be easily updated at each iteration, while y_{n+1} and x_{n+1} will usually differ in every element from y_n and x_n, respectively. We refer to this algorithm as Classical GMRES.

Several variants for computing the basis vectors $v_1, \ldots, v_n$ were proposed. Saad and Schultz [26] considered the modified Gram–Schmidt process. Walker [32, 33] presented Classical GMRES based on Householder transformations. Iterated classical and iterated modified Gram–Schmidt versions were studied in [9].

A variety of parallel implementations [6, 3, 12, 23, 7, 27] use various techniques to increase the parallel efficiency of the basically sequential basis-generating process. Parallel aspects are out of the scope of this paper.

4.2. Orthogonal projection principle: Simpler GMRES. We now consider an implementation of an MR method derived from the orthogonal projection principle (1.4). The approach proposed by Walker and Zhou [34], called Simpler GMRES, uses the orthonormal basis (3.2).

This basis is computed by a recursive columnwise QR-factorization of the matrix $[Ar_0\gamma, AW_{n-1}]$. Based on Theorem 3.1 we set $\gamma = \rho_0^{-1}$, and we will use this value of the scaling parameter γ throughout the rest of this paper. Then

$$A[v_1, W_{n-1}] = [Av_1, AW_{n-1}] = W_n S_n, \;\; W_n \equiv [w_1, \ldots, w_n], \;\; W_n^T W_n = I, \tag{4.6}$$

where S_n is an n-by-n upper triangular matrix with elements $s_{i,j}$, $s_{j,j} \neq 0$. If at any stage $s_{n,n} = 0$, the algorithm would stop with $[Av_1, AW_{n-1}] = W_{n-1}[S_{n-1}, \hat{s}_n]$. Using the substitution

$$x_n \;=\; x_0 \,+\, [v_1, W_{n-1}]\, t_n\,, \tag{4.7}$$

the vector $t_n \in \mathcal{R}^n$ solves the LS problem

$$\|r_n\| \equiv \|b - Ax_n\| = \min_{t \in \mathcal{R}^n} \|r_0 - A[v_1, W_{n-1}]\, t\| \tag{4.8}$$

$$= \min_{t \in \mathcal{R}^n} \|r_0 - W_n S_n\, t\|\,. \tag{4.9}$$

Solving the LS problem (4.8)–(4.9) in a numerically stable way represents a more subtle task then solving (4.3)–(4.4). The main difference is in handling the right-hand side vector r_0. In (4.3)–(4.4), r_0 is expressed in terms of the vectors $v_1, v_2, \ldots, v_n$ simply as $r_0 = v_1 \rho_0$. In finite precision arithmetic, until the linear independence of the vectors $v_1, v_2, \ldots, v_n$ is lost, this expression is maximally accurate. On the other hand, application of the orthogonal projection principle (1.4) directly to (4.8)–(4.9) gives the upper triangular system

$$S_n\, t_n \;=\; W_n^T r_0. \tag{4.10}$$

As demonstrated in [25], computing the vector of coefficients t_n from (4.10) leads to numerical difficulties. Numerically more stable implementations are described next.

First consider the implementation of Simpler GMRES using the modified Gram–Schmidt process for generating the basis vectors $w_1, \ldots, w_n$. A properly implemented algorithm for solving the LS problem (4.8)–(4.9) applies the orthogonalization process also to the right-hand side r_0 (see [4, pp. 64–65]). Then, using the recursive columnwise modified Gram–Schmidt QR-factorization of the extended matrix $[Av_1, AW_{n-1}, r_0]$,

$$[Av_1, AW_{n-1}, r_0] \;=\; W_n\,[S_n, z_n] \,+\, \left[0, \frac{r_n}{\|r_n\|}\right] \left[\begin{array}{c} 0 \\ \|r_n\| \end{array}\right], \tag{4.11}$$

the vector t_n solves the upper triangular system

$$S_n\, t_n \;=\; z_n. \tag{4.12}$$

The jth component of $z_n \equiv (\zeta_1, \ldots, \zeta_n)^T$ is determined by

$$\zeta_j \;=\; w_j^T\,(I - w_{j-1}w_{j-1}^T)\cdots(I - w_1 w_1^T)\,r_0 \;=\; w_j^T\, r_{j-1}\,, \tag{4.13}$$

where we use

$$r_j \;=\; (I - w_j w_j^T)\cdots(I - w_1 w_1^T)\,r_0 \;=\; r_{j-1} \,-\, (w_j^T\, r_{j-1})\, w_{j-1}\,. \tag{4.14}$$

A complete algorithm of the modified Gram–Schmidt implementation of Simpler GMRES is given in the appendix.

Now we consider the implementation of Simpler GMRES based on Householder reflections. It transforms the matrix $[Av_1, AW_{n-1}]$ into upper triangular form,

$$P_n \cdots P_2 P_1\,[Av_1, AW_{n-1}] \;=\; \left[\begin{array}{c} S_n \\ 0 \end{array}\right], \tag{4.15}$$

where $P_j,\;\, j \,=\, 1, \ldots, n$, are elementary Householder matrices. (For details, see [9, p. 312].) Then the transformed right-hand side is determined as

$$z_n \;=\; [P_n \cdots P_1\, r_0]_{\{1:n\}},$$

where $[\cdot]_{\{1:n\}}$ denotes the first n elements of a vector. The vector of coefficients t_n is determined from (4.12). A complete algorithm of the Householder implementation of Simpler GMRES is given in the appendix.

Related to Simpler GMRES are stabilized Orthodir [31] and the recent A^TA-variant of GMRES [25]. Both compute an A^TA-orthogonal basis of $\mathcal{K}_n(A, r_0)$, and thus each step of these methods requires about twice as much storage and also slightly more arithmetic operations than Simpler GMRES. They are also numerically less stable than Simpler GMRES. On the other hand, they allow easier parallel implementations because they feature step by step updates of both the approximate solution and the residual vectors.

5. Numerical stability. In this section we analyze and compare the numerical stability of Classical and Simpler GMRES. As mentioned in section 4, different orthogonalization techniques for computing the columns of V_n or W_n can be applied. Here we focus on implementations based on Householder transformations [32, 33] and on the modified Gram–Schmidt process [26].

For distinction, we denote quantities computed in finite precision arithmetic (with the machine precision ε) by bars. We assume the standard model of floating point arithmetic (see, e.g., [15, equation (2.4)]). In our bounds we present only those terms which are linear in ε and do not account for the terms proportional to higher powers of ε. By $p_k(n,m,N)$, $k=1,2,\ldots$, we denote low degree polynomials in the number of iteration steps n, the maximum number of nonzeros per row in the system matrix m, and the dimension of the system N. They are introduced in a number of places in the text; some of them depend only on one or two variables. In all cases, $p_k(n,m,N) \leq c_k N^{5/2}$, where $c_k > 0$ is a constant independent of n, m, and N. This bound is, in general, very pessimistic; it accounts for the worst possible case. For details, see [9, 14, 24].

5.1. Classical GMRES. In the Classical GMRES implementation the computed approximate solution $\bar{x}_n$ satisfies

$$\begin{aligned} \bar{x}_n &= x_0 + \bar{V}_n \bar{y}_n + g_n, \\ \|g_n\| &\leq \varepsilon \|x_0\| + p_1(n)\,\varepsilon \|\bar{V}_n\| \|\bar{y}_n\|. \end{aligned} \tag{5.1}$$

It was shown in [9, 14] that the computed matrix $\bar{V}_n = [\bar{v}_1, \bar{v}_2, \ldots, \bar{v}_n]$ satisfies the recurrence

$$\begin{aligned} [\bar{r}_0, A\bar{V}_n] &= \hat{V}_{n+1}\,[\bar{\rho}_0 e_1, \bar{H}_{n+1,n}] + [f_n, F_n]\,, \\ \|f_n\| &\leq p_2(m,N)\,\varepsilon \|A\| \|x_0\| + p_3(N)\,\varepsilon \|b\|, \\ \|F_n\| &\leq p_4(n,m,N)\,\varepsilon \|A\| \|\bar{V}_n\|, \end{aligned} \tag{5.2}$$

where the matrix $\hat{V}_{n+1}$ has exactly orthogonal columns ($\hat{V}_{n+1}^T \hat{V}_{n+1} = I_{n+1}$). The vector $\bar{y}_n$ is a computed solution of the finite precision analogue of the transformed LS problem (4.4), and $\bar{r}_0$ satisfies

$$\|\bar{r}_0 - (b - Ax_0)\| \leq p_5(m,N)\,\varepsilon \|A\| \|x_0\| + p_6(N)\,\varepsilon \|b\|\,. \tag{5.3}$$

For details, we refer to [9] and also to [24, pp. 25–26].

Our goal is not to give a complete rounding error analysis of GMRES. (For the Householder implementation of Classical GMRES this was published in [9], and the modified Gram–Schmidt implementation of Classical GMRES has been analyzed in [14, 24, 21].) We wish to explain that there is a potential weakness of Simpler GMRES which may negatively affect its computational behavior in comparison with Classical GMRES. For this purpose we can simplify our description of the GMRES convergence. This allows us to avoid tedious details which would make reading of this section difficult. We will describe the convergence of Classical GMRES by the norm of the LS residual associated with the matrix $A\bar{V}_n$ and the computed initial residual $\bar{r}_0$:

$$\|\hat{r}_n\| \equiv \|\bar{r}_0 - A\bar{V}_n \hat{y}_n\| = \min_y \|\bar{r}_0 - A\bar{V}_n y\|. \tag{5.4}$$

The analysis in [14, section 3] as well as numerical experiments confirm that for Classical GMRES $\|\hat{r}_n\|$ is close to the norm of the actually computed GMRES residual $\|b - A\bar{x}_n\|$.

It follows immediately from (2.14) that the residual norm (5.4) can be bounded in terms of the minimal singular values of matrices $[\bar{r}_0, A\bar{V}_n]$ and $A\bar{V}_n$ as

$$\sigma_{min}([\bar{r}_0, A\bar{V}_n]) \leq \|\hat{r}_n\| \leq \sigma_{min}([\bar{r}_0, A\bar{V}_n]) \left\{ 1 - \frac{\sigma_{min}^2([\bar{r}_0, A\bar{V}_n])}{\sigma_{min}^2(A\bar{V}_n)} + \frac{\|\bar{r}_0\|^2}{\sigma_{min}^2(A\bar{V}_n)} \right\}^{1/2}. \tag{5.5}$$

We see that convergence of the residual $\hat{r}_n$ is closely related to ill-conditioning of the matrix $[\bar{r}_0, A\bar{V}_n]$; i.e., decreasing $\|\hat{r}_n\|$ leads to ill-conditioning of $[\bar{r}_0, A\bar{V}_n]$. Moreover, it follows from (5.2) and from classical perturbation theory (see, e.g., [13, p. 449]), that the minimum singular values of the matrices $[\bar{r}_0, A\bar{V}_n]$ and $[\bar{\rho}_0 e_1, \bar{H}_{n+1,n}]$ are close to each other:

$$\left|\sigma_{min}([\bar{\rho}_0 e_1, \bar{H}_{n+1,n}]) - \sigma_{min}([\bar{r}_0, A\bar{V}_n])\right| \le \|[f_n, F_n]\| \,. \tag{5.6}$$

Consequently, decreasing $\|\hat{r}_n\|$ leads to ill-conditioning of the matrix $[\bar{\rho}_0 e_1, \bar{H}_{n+1,n}]$. The vector $\bar{y}_n$ from (5.1) is a computed solution of the LS problem

$$\min_y \|e_1 \bar{\rho}_0 - \bar{H}_{n+1,n} y\|. \tag{5.7}$$

Using (5.2), the extremal singular values of $\bar{H}_{n+1,n}$ can be bounded by

$$\|\bar{H}_{n+1,n}\| \le \|A\bar{V}_n\| + \|F_n\| \le \|A\|\|\bar{V}_n\| + \|F_n\|, \tag{5.8}$$

$$\sigma_{min}(\bar{H}_{n+1,n}) \ge \sigma_{min}(A\bar{V}_n) - \|F_n\| \ge \sigma_{min}(A)\,\sigma_{min}(\bar{V}_n) - \|F_n\|. \tag{5.9}$$

When $\|\hat{r}_n\|$ (and $\|b - A\bar{x}_n\|$) decreases, $\sigma_{min}([\bar{r}_0, A\bar{V}_n])$ and $\sigma_{min}([\bar{\rho}_0 e_1, \bar{H}_{n+1,n}])$ also decrease. However, while the columns of $\bar{V}_n$ (the Arnoldi vectors) keep their linear independence (while $\sigma_{min}(\bar{V}_n) \approx 1$), *the condition number of the computed upper Hessenberg matrix* $\bar{H}_{n+1,n}$ *is essentially bounded by the condition number of* A. Consequently, until the linear independence of the Arnoldi vectors begins to deteriorate, the solution $\bar{y}_n$ of the transformed LS problem and the GMRES solution $\bar{x}_n$ are affected by rounding errors in a minimal possible way. This distinguishes Classical GMRES from the other MR methods, in particular from Simpler GMRES. Finite precision analysis of the QR-factorization of the matrix $\bar{H}_{n+1,n}$ via Givens rotations and of forming the GMRES solution can be found in [9] or [24, equations (4.6)–(4.12)].

It is important to note that not the orthogonality but the linear independence of the columns of $\bar{V}_n$ (measured by its extremal singular values) plays a decisive role in the relations (5.8) and (5.9). If we use Householder reflections in the Arnoldi process, the loss of orthogonality among the computed columns of $\bar{V}_n$ and the extremal singular values of $\bar{V}_n$ are bounded independent of the system parameters

$$1 - p_7(n, N)\,\varepsilon \le \sigma_n(\bar{V}_n) \le \|\bar{V}_n\| \le 1 + p_7(n, N)\,\varepsilon. \tag{5.10}$$

Moreover, it was shown in [9] that the Householder implementation of Classical GMRES is backward stable. Assuming that a conjecture similar to (5.10) holds, the same result can also be shown for the iterated classical or modified Gram–Schmidt implementations; see [9].

In practical computations, cheaper orthogonalization techniques like the modified Gram–Schmidt algorithm are used. It is well known that the orthogonality among the columns of $\bar{V}_n$ computed via the modified Gram–Schmidt process will gradually deteriorate. However, from [14, equation (1.7) and Corollary 2.4] it follows that

$$\|\hat{V}_n - \bar{V}_n\| \le p_8(n, m, N)\,\varepsilon\kappa([\bar{v}_1, A\bar{V}_{n-1}]), \tag{5.11}$$

and the minimal singular value and the norm of $\bar{V}_n$ are bounded by

$$1 - \frac{p_9(n, m, N)\,\varepsilon\kappa(A)}{\|\hat{r}_{n-1}\|/\bar{\rho}_0} \le \sigma_n(\bar{V}_n) \le \|\bar{V}_n\| \le 1 + \frac{p_9(n, m, N)\,\varepsilon\kappa(A)}{\|\hat{r}_{n-1}\|/\bar{\rho}_0}. \tag{5.12}$$

The columns of $\bar{V}_n$ will thus begin to lose their linear independence *only after* the relative residual norm is reduced close to the level $\varepsilon\kappa(A)$. Up to that point the modified Gram–Schmidt implementation of Classical GMRES behaves about as well as the Householder implementation.

It was shown in [20, 21] that there is a tight relation between the normwise backward error $\|b - Ax_n\|/(\|A\|\|x_n\| + \|b\|)$ associated with the approximate solution x_n and the condition number of the matrix $[r_0, AV_n]$. A finite precision analogy of this statement will prove normwise backward stability of the modified Gram–Schmidt implementation of Classical GMRES. A formal proof will be given elsewhere.

The results in [20, 21] are based on (2.16). We could have also used (2.16) instead of (2.14) in our derivation here, which would lead to tighter estimates. However, using (2.14) makes our derivation much simpler, and the results are fully sufficient for our purpose.

5.2. Simpler GMRES. In Simpler GMRES the approximate solution $\bar{x}_n$ computed in finite precision arithmetic satisfies

$$\begin{aligned}\bar{x}_n &= x_0 + [\bar{v}_1, \bar{W}_{n-1}]\,\bar{t}_n + g_n,\\ \|g_n\| &\le \varepsilon\|x_0\| + p_1(n)\,\varepsilon\|[\bar{v}_1, \bar{W}_{n-1}]\|\,\|\bar{t}_n\|.\end{aligned} \tag{5.13}$$

Analogously to (5.2), for every iteration step n there exists a matrix $\hat{W}_n$ with exactly orthonormal columns ($\hat{W}_n^T\hat{W}_n = I$) such that

$$\begin{aligned}A[\bar{v}_1, \bar{W}_{n-1}] &= \hat{W}_n\bar{S}_n + F_n,\\ \|F_n\| &\le p_4(n,m,N)\,\varepsilon\|A\|\|[\bar{v}_1, \bar{W}_{n-1}]\|.\end{aligned} \tag{5.14}$$

The vector of coefficients $\bar{t}_n$ is computed by solving the upper triangular system with the matrix $\bar{S}_n$. From (5.14) the extremal singular values of the matrix $\bar{S}_n$ are bounded by

$$\|\bar{S}_n\| \le \|A[\bar{v}_1, \bar{W}_{n-1}]\| + \|F_n\| \le \|A\|\|[\bar{v}_1, \bar{W}_{n-1}]\| + \|F_n\|, \tag{5.15}$$

$$\begin{aligned}\sigma_{min}(\bar{S}_n) &\ge \sigma_{min}(A[\bar{v}_1, \bar{W}_{n-1}]) - \|F_n\|\\ &\ge \sigma_{min}(A)\,\sigma_{min}([\bar{v}_1, \bar{W}_{n-1}]) - \|F_n\|.\end{aligned} \tag{5.16}$$

The minimal singular value of the matrix $[\bar{v}_1, \bar{W}_{n-1}]$ can further be related to the minimal singular value of the matrix $[\bar{r}_0/\|\bar{r}_0\|, \hat{W}_{n-1}]$, where $\hat{W}_{n-1}$ comes from the recurrence (5.14),

$$\sigma_n([\bar{v}_1, \bar{W}_{n-1}]) \ge \sigma_n([\bar{r}_0/\|\bar{r}_0\|, \hat{W}_{n-1}]) - \|[\bar{v}_1 - \bar{r}_0/\|\bar{r}_0\|, \bar{W}_{n-1} - \hat{W}_{n-1}]\|. \tag{5.17}$$

For the condition number $\kappa([\bar{r}_0/\|\bar{r}_0\|, \hat{W}_{n-1}])$ it follows from (3.6) that

$$\kappa([\bar{r}_0/\|\bar{r}_0\|,\, \hat{W}_{n-1}]) = \frac{\|\bar{r}_0\| + \left(\|\bar{r}_0\|^2 - \|\hat{r}_{n-1}\|^2\right)^{1/2}}{\|\hat{r}_{n-1}\|}, \tag{5.18}$$

where $\hat{r}_{n-1} \equiv (I - \hat{W}_{n-1}\hat{W}_{n-1}^T)\,\bar{r}_0$ is the LS residual associated with the matrix $\hat{W}_{n-1}$, $\|\hat{r}_{n-1}\| = \min_y \|\bar{r}_0 - \hat{W}_{n-1}y\|$. The identity (5.18) proves that *convergence of the residual norm $\|\hat{r}_{n-1}\|$ and ill-conditioning of the matrix $[\bar{r}_0/\|\bar{r}_0\|, \hat{W}_{n-1}]$ are closely related.*

Summarizing, small $\|\bar{W}_{n-1} - \hat{W}_{n-1}\|$ means $\kappa([\bar{v}_1, \bar{W}_{n-1}]) \approx \kappa([\bar{r}_0/\|\bar{r}_0\|, \hat{W}_{n-1}])$. (It can be shown that $\|\bar{v}_1 - \bar{r}_0/\|\bar{r}_0\|\| \le (N+4)\varepsilon$; see [9].) Using (5.15) and (5.16), we conclude that *decreasing* $\|\hat{r}_{n-1}\|$ *may lead to ill-conditioning of the upper triangular matrix* $\bar{S}_n$*, and thus to a potentially large error in computing the vector* $\bar{t}_n$*, independent of the (well-) conditioning of the matrix* A. This important fact may negatively affect the numerical accuracy of the approximate solution $\bar{x}_n$ in Simpler GMRES in comparison to Classical GMRES.

Until $\bar{S}_n$ becomes pathologically ill-conditioned, $\|\hat{r}_n\|$ is (similarly to subsection 5.1) close to $\|b - A\bar{x}_n\|$. After that the behavior of $\|\hat{r}_n\|$ and $\|b - A\bar{x}_n\|$ may be significantly different.

We have seen that the relation between the condition number of the matrix $\bar{S}_n$ and the condition number of the matrix $[\bar{r}_0/\|\bar{r}_0\|, \hat{W}_{n-1}]$ (the decrease of $\|\hat{r}_n\|$) is strongly affected by the size of the term $\|\bar{W}_{n-1} - \hat{W}_{n-1}\|$. In the Householder implementation the computed matrix $\bar{W}_{n-1}$ is, up to a small multiple of the machine precision, close to the matrix $\hat{W}_{n-1}$ with exactly orthogonal columns,

$$\|\bar{W}_{n-1} - \hat{W}_{n-1}\| \leq p_7(n, N)\, \varepsilon. \tag{5.19}$$

It follows from (5.19) that the condition number $\kappa([\bar{v}_1, \bar{W}_{n-1}])$ is, up to terms proportional to the machine precision, equal to $\kappa([\bar{r}_0/\|\bar{r}_0\|, \hat{W}_{n-1}])$. In practice one frequently observes that after $\|b - A\bar{x}_n\|/\|\bar{r}_0\|$ reaches some particular point the norm of the computed vector $\bar{t}_n$ starts to increase dramatically (the computed results become irrelevant due to rounding errors), and the residual norm $\|b - A\bar{x}_n\|$ diverges.

For the modified Gram–Schmidt implementation we have

$$\|\bar{W}_{n-1} - \hat{W}_{n-1}\| \leq p_8(n, m, N)\, \varepsilon \kappa(A[\bar{v}_1, \bar{W}_{n-1}]). \tag{5.20}$$

Because $\kappa(A[\bar{v}_1, \hat{W}_{n-1}])$ is potentially much worse than $\kappa([\bar{v}_1, A\hat{V}_{n-1}])$, the linear independence of the columns of $\bar{W}_n$ often begins to deteriorate much sooner than the linear independence of the columns of $\bar{V}_n$ in Classical GMRES. Until that point the modified Gram–Schmidt and Householder implementations of Simpler GMRES behave similarly. In subsequent iterations, surprisingly, the behavior of the modified Gram–Schmidt implementation of Simpler GMRES may be better than the behavior of the Householder implementation. For the Householder implementation of Simpler GMRES the true residual $b - A\bar{x}_n$ often diverges. This has been linked to the tight relation between $\kappa([\bar{r}_0/\|\bar{r}_0\|, \hat{W}_{n-1}])$ and $\kappa([\bar{v}_1, \bar{W}_{n-1}])$, and, consequently, to the relation between the decrease of $\|\hat{r}_n\|$ and the simultaneous increase of $\kappa(\bar{S}_n)$. For the modified Gram–Schmidt implementation, after reaching a certain point no such relations hold. The norm of $\bar{t}_n$ does not diverge, and the norm of the true residual remains (and often slightly oscillates) on or below the level corresponding to the turning point for the Householder implementation.

5.3. Numerical experiments. The different behavior of Classical and Simpler GMRES implementations is demonstrated by numerical examples with the matrix FS1836 from the Harwell–Boeing collection, $N = 183$, $\kappa(A) = 1.5 \times 10^{11}$, $\|A\| = 1.2 \times 10^9$. Experiments were performed using MATLAB 5.2, $\varepsilon = 1.1 \times 10^{-16}$. Householder and modified Gram–Schmidt orthogonalizations have been considered for both Classical and Simpler GMRES. In all experiments we used $x = (1, \ldots, 1)^T$, $b = Ax$, and $x_0 = 0$ ($\|\bar{r}_0\| = \|b\|$).

Figures 1 and 2 illustrate characteristics of the transformed LS problem (5.7) for the Householder and the modified Gram–Schmidt implementations of Classical GMRES. In both figures horizontal dotted lines represent $\|A\|$ and the minimal singular value $\sigma_{min}(A)$. The dashed lines show $\|\bar{H}_{n+1,n}\|$, the norm of the computed

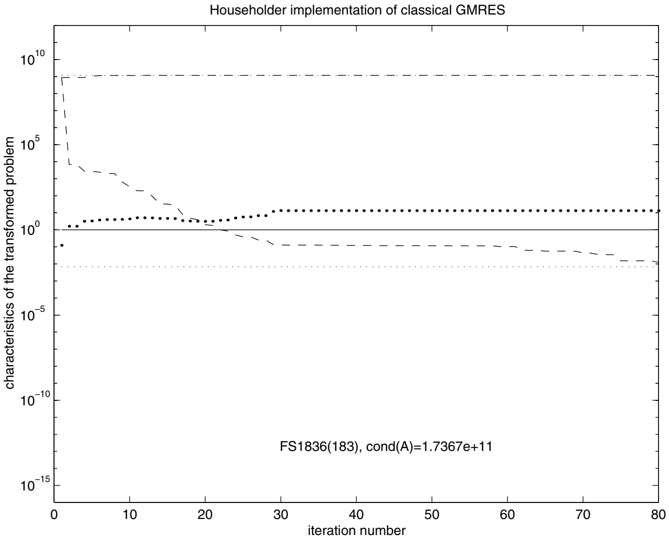


FIG. 1. *Householder implementation of Classical GMRES: $\|A\|$ and $\sigma_{min}(A)$ (dotted lines), $\|\bar{H}_{n+1,n}\|$ and $\sigma_{min}(\bar{H}_{n+1,n})$ (dashed lines), $\sigma_{min}(\bar{V}_n)$ (solid line), and $\|\bar{y}_n\|$ (dots).*

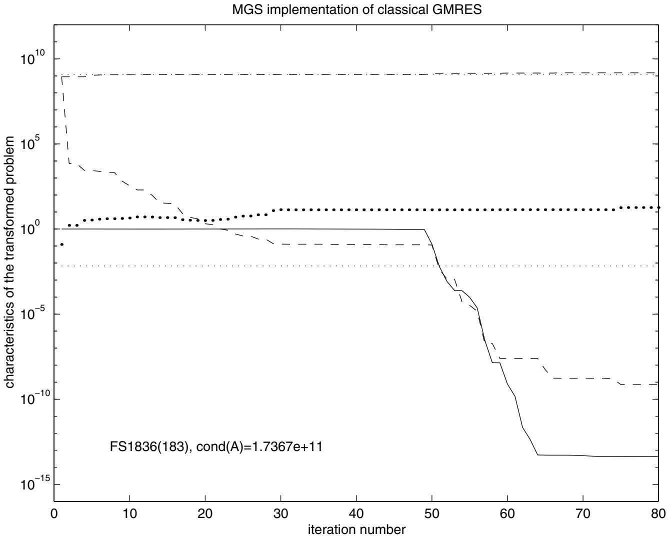


FIG. 2. *Modified Gram–Schmidt implementation of Classical GMRES: $\|A\|$ and $\sigma_{min}(A)$ (dotted lines), $\|\bar{H}_{n+1,n}\|$ and $\sigma_{min}(\bar{H}_{n+1,n})$ (dashed lines), $\sigma_{min}(\bar{V}_n)$ (solid line), and $\|\bar{y}_n\|$ (dots).*

upper Hessenberg matrix (it almost coincides with $\|A\|$), and the minimal singular value $\sigma_{min}(\bar{H}_{n+1,n})$. The solid line stands for $\sigma_{min}(\bar{V}_n)$, the minimal singular value of the matrix of computed Arnoldi vectors, and the dots depict $\|\bar{y}_n\|$, the norm of the computed solution vector of (5.7). We see that until the linear independence of the columns of $\bar{V}_n$ in the modified Gram–Schmidt implementation begins to deteriorate, Figures 1 and 2 are almost identical. There is no substantial growth in $\|\bar{y}_n\|$ even after the linear independence of the computed Arnoldi vectors is completely lost (cf. Figure 2).

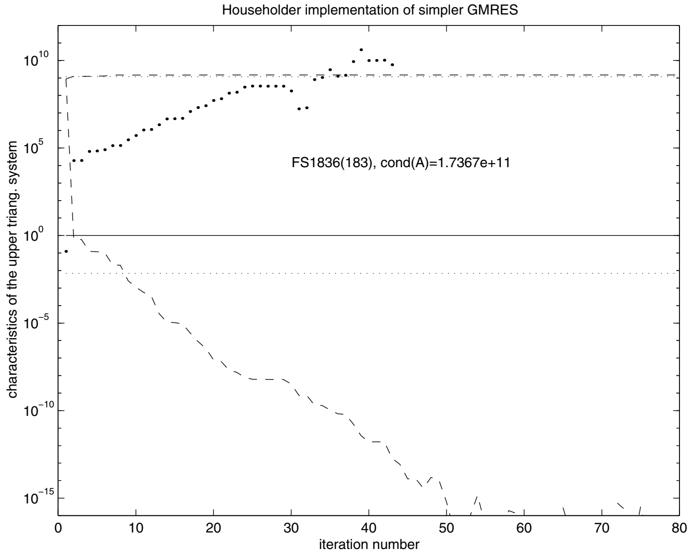


FIG. 3. *Householder implementation of Simpler GMRES: $\|A\|$ and $\sigma_{min}(A)$ (dotted lines), $\|\bar{S}_n\|$ and $\sigma_{min}(\bar{S}_n)$ (dashed lines), $\sigma_{min}(\bar{W}_n)$ (solid line), and $\|\bar{t}_n\|$ (dots).*

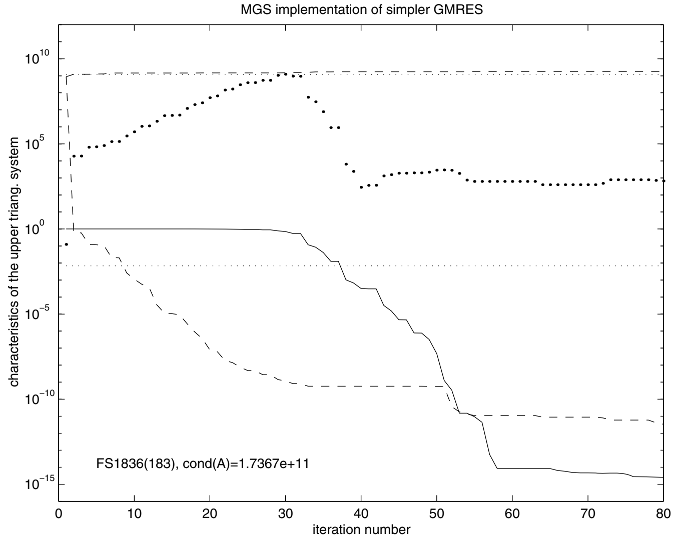


FIG. 4. *Modified Gram–Schmidt implementation of Simpler GMRES: $\|A\|$ and $\sigma_{min}(A)$ (dotted lines), $\|\bar{S}_n\|$ and $\sigma_{min}(\bar{S}_n)$ (dashed lines), $\sigma_{min}(\bar{W}_n)$ (solid line), and $\|\bar{t}_n\|$ (dots).*

Similar quantities are illustrated in Figures 3 and 4 for the Householder and the modified Gram–Schmidt implementations of Simpler GMRES. The dashed lines here represent $\|\bar{S}_n\|$, the norm of the computed upper triangular matrix, and its minimal singular value $\sigma_{min}(\bar{S}_n)$. The dots denote $\|\bar{t}_n\|$, the norm of the computed solution of the upper triangular system with the matrix $\bar{S}_n$.

We see that the condition number of the matrix $\bar{H}_{n+1,n}$ is in Figure 1 (the House-

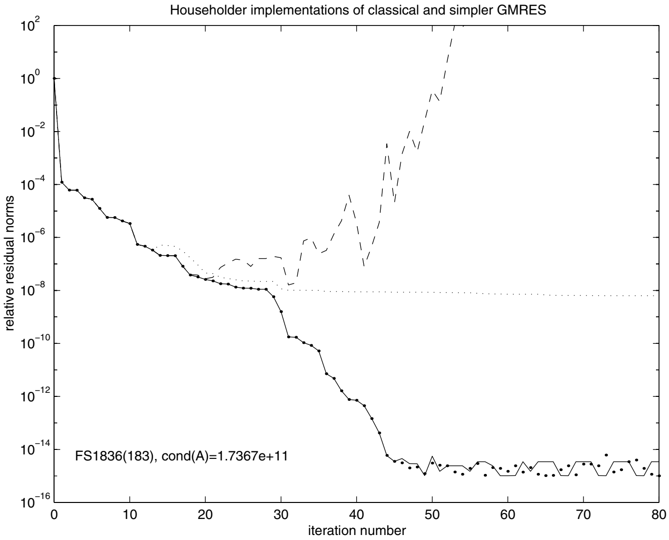


Fig. 5. *Householder implementation of Classical and Simpler GMRES: Normalized true residual norm $\|b - A\bar{x}_n\|/\|b\|$ (solid line—Classical GMRES, dashed line—Simpler GMRES), and $\|\hat{r}_n\|/\|b\|$ (dots—Classical GMRES, dotted line—Simpler GMRES).*

holder implementation of Classical GMRES) approximately bounded by the condition number of A, and for Figure 2 (the modified Gram–Schmidt implementation of Classical GMRES) the same is true until $\sigma_{min}(\bar{V}_n)$ begins to deteriorate. In contrast, in both implementations of Simpler GMRES, the minimal singular value of $\bar{S}_n$ decreases very soon far below $\sigma_{min}(A)$. Consequently, the accuracy of the computed vector $\bar{t}_n$ deteriorates, and for the Householder implementation $\|\bar{t}_n\|$ diverges. Also note the difference between $\sigma_{min}(\bar{V}_n)$ and $\sigma_{min}(\bar{W}_n)$ in Figures 2 and 4.

In Figure 5 we compare the convergence characteristics for the Householder implementations of both Classical GMRES ($\|b - A\bar{x}_n\|/\|b\|$ is represented by the solid line, $\|\hat{r}_n\|/\|b\|$ by dots) and Simpler GMRES ($\|b - A\bar{x}_n\|/\|b\|$ is represented by the dashed line, $\|\hat{r}_n\|/\|b\|$ by the dotted line). Figure 5 illustrates our theoretical considerations and shows that the true residual norm of the Householder implementation of Simpler GMRES may after some initial reduction diverge. Figure 6 uses similar notation for the illustration of the modified Gram–Schmidt implementations. The residual norm of Simpler GMRES again exhibits worse behavior than the residual norm corresponding to Classical GMRES.

6. Conclusions.

MR methods can be formulated and implemented using different bases and different orthogonalization processes. Using general theoretical results about the LS residual, this paper shows that the choice of the basis is fundamental for getting revealing theoretical results about convergence of MR methods. It is also important for getting a numerically stable implementation. The choice of the computed basis may strongly affect the numerical behavior of the implementation. It is explained that using the best orthogonalization technique in building the basis does not compensate for the possible loss of accuracy in a given method which is related to the choice of the basis. In particular, it is shown that the classical implementation of GMRES, which is based on the Arnoldi process starting from the normalized initial residual (as proposed by Saad and Schultz), has numerical advantages over Simpler

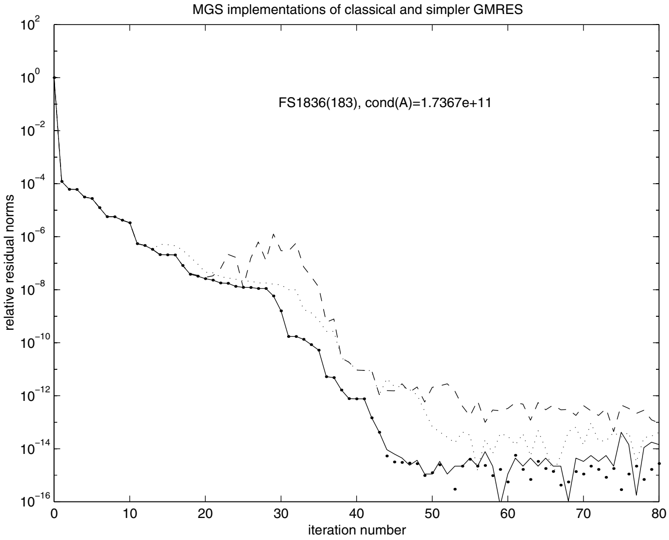


FIG. 6. *Modified Gram–Schmidt implementation of Classical and Simpler GMRES: Normalized true residual norm $\|b - A\bar{x}_n\|/\|b\|$ (solid line—Classical GMRES, dashed line—Simpler GMRES), and $\|\hat{r}_n\|/\|b\|$ (dots—Classical GMRES, dotted line—Simpler GMRES).*

GMRES, which is based on the shifted Arnoldi process.

7. Appendix.

Here we present the implementations of Simpler GMRES used throughout the paper.

Modified Gram–Schmidt implementation of Simpler GMRES:

```
$x_0$, $r_0 = b - Ax_0$, $v_1 = r_0/\|r_0\|$, $w_0 = v_1$
$n = 1, 2, \ldots$
    $w_n = Aw_{n-1}$
    $j = 1, 2, \ldots, n-1$
        $w_n \leftarrow w_n - \rho_{j,n} w_j$,  $\rho_{j,n} = (w_n, w_j)$
    $w_n \leftarrow w_n/\rho_{n,n}$,  $\rho_{n,n} = \|w_n\|$
    $S_n = \begin{pmatrix} S_{n-1} & \rho_{1,n} \\ & \vdots \\ 0 & \rho_{n,n} \end{pmatrix}$,  $S_1 = (\rho_{1,1})$
    $r_n = r_{n-1} - \zeta_n w_n$,  $\zeta_n = (r_{n-1}, w_n)$
    Solve $S_n\, t_n = (\zeta_1, \ldots, \zeta_n)^T$
    $x_n = x_0 + [v_1, w_1, \ldots, w_{n-1}]\, t_n$
```

Householder implementation of Simpler GMRES:

```
$x_0,\ r_0 = b - Ax_0,\ v_1 = r_0/\|r_0\|,\ (\zeta_1, \ldots, \zeta_N)^T = r_0,\ w_0 = v_1$

$n = 1, 2, \ldots$

    Compute $P_n$ such that $P_n\, Aw_{n-1} = (\rho_{1,n}, \ldots, \rho_{n,n}, 0, \ldots, 0)^T$

    $S_n = \begin{pmatrix} S_{n-1} & \rho_{1,n} \\ & \vdots \\ 0 & \rho_{n,n} \end{pmatrix}, \quad S_1 = (\rho_{1,1})$

    $(\zeta_1, \ldots, \zeta_N)^T \leftarrow P_n\, (\zeta_1, \ldots, \zeta_N)$

    Solve    $S_n\, t_n = (\zeta_1, \ldots, \zeta_n)^T$

    $r_n = r_{n-1} - \zeta_n w_n$

    $w_n = P_1 \ldots P_n\, e_n$

    $x_n = x_0 + [v_1, w_1, \ldots, w_{n-1}]\, t_n$
```

Acknowledgments. The authors are indebted to Chris Paige for his comments which improved the formulation and interpretation of several results. They also wish to thank the referees for their comments which improved presentation of the results.

REFERENCES

[1] O. Axelsson, *Conjugate gradient type methods for unsymmetric and inconsistent systems of linear equations*, Linear Algebra Appl., 29 (1980), pp. 1–16.

[2] O. Axelsson, *A generalized conjugate gradient, least square method*, Numer. Math., 51 (1987), pp. 209–227.

[3] Z. Bai, D. Hu, and L. Reichel, *A Newton basis GMRES implementation*, IMA J. Numer. Anal., 14 (1994), pp. 563–581.

[4] A. Björck, *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, PA, 1996.

[5] S. L. Campbell and C. D. Meyer, Jr., *Generalized Inverses of Linear Transformations*, Pitman, Boston, MA, 1979.

[6] E. de Sturler, *A parallel variant of GMRES(m)*, in Proceedings of the 13th World Congress on Computation and Applied Mathematics, Dublin, Ireland, 1991, pp. 682–683.

[7] E. de Sturler and H. van der Vorst, *Reducing the effect of global communication in GMRES(m) and CG on parallel distributed memory computers*, Appl. Numer. Math., 18 (1995), pp. 441–459.

[8] J. Demmel, *The condition number of equivalence transformations that block diagonalize matrix pencils*, SIAM J. Numer. Anal., 20 (1983), pp. 599–610.

[9] J. Drkošová, A. Greenbaum, M. Rozložník, and Z. Strakoš, *Numerical stability of GMRES*, BIT, 35 (1995), pp. 309–330.

[10] S. C. Eisenstat, H. C. Elman, and M. H. Schultz, *Variational iterative methods for nonsymmetric systems of linear equations*, SIAM J. Numer. Anal., 20 (1983), pp. 345–357.

[11] H. Elman, *Iterative Methods for Large Sparse Nonsymmetric Systems of Linear Equations*, Ph.D. thesis, Yale University, New Haven, CT, 1982.

[12] J. Erhel, *A parallel GMRES version for general sparse matrices*, Electron. Trans. Numer. Anal., 3 (1995), pp. 160–176.

[13] G. H. Golub and C. F. van Loan, *Matrix Computations*, 3rd ed., Johns Hopkins University Press, Baltimore, MD, 1996.

[14] A. Greenbaum, M. Rozložník, and Z. Strakoš, *Numerical behaviour of the modified Gram-Schmidt GMRES implementation*, BIT, 37 (1997), pp. 706–719.

[15] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia, PA, 1996.

[16] I. Ipsen, *A Different Approach to Bounding the Minimal Residual Norm in Krylov Methods*, Technical report, CRSC, Department of Mathematics, North Carolina State University, Raleigh, NC, 1998.

[17] I. C. F. Ipsen, *Expressions and bounds for the GMRES residual*, BIT, 40 (2000), pp. 524–535.

[18] I. M. Khabaza, *An iterative least-square method suitable for solving large sparse matrices*, Comput. J., 6 (1963/1964), pp. 202–206.

[19] J. Liesen, *Construction and Analysis of Polynomial Iterative Methods for Non-Hermitian Systems of Linear Equations*, Ph.D. thesis, Fakultät für Mathematik, Universität Bielefeld, Bielefeld, Germany, 1998; also available online from http://archiv.ub.uni-bielefeld.de/disshabi/mathe.htm.

[20] C. Paige and Z. Strakoš, *Bounds for the least squares distance via scaled total least squares problems*, Numer. Math., to appear.

[21] C. Paige and Z. Strakoš, *Residual and backward error bounds in minimum residual Krylov subspace methods*, SIAM J. Sci. Comput., to appear.

[22] C. Paige and Z. Strakoš, *Scaled total least squares fundamentals*, Numer. Math., to appear.

[23] B. Philippe and R. Sidje, *Parallel algorithms for the Arnoldi procedure*, in Iterative Methods in Linear Algebra, II, IMACS Ser. Comput. Appl. Math. 3, IMACS, New Brunswick, NJ, 1995, pp. 156–165.

[24] M. Rozložník, *Numerical Stability of the GMRES Method*, Ph.D. thesis, Institute of Computer Science, Academy of Sciences of the Czech Republic, Prague, 1997; also available online from http://www.cs.cas.cz/~miro.

[25] M. Rozložník and Z. Strakoš, *Variants of the residual minimizing Krylov space methods*, in Proceedings of the 11th Summer School on Software and Algorithms of Numerical Mathematics, I. Marek, ed., 1995, pp. 208–225.

[26] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[27] R. B. Sidje, *Alternatives for parallel Krylov subspace basis computation*, Numer. Linear Algebra Appl., 4 (1997), pp. 305–331.

[28] G. W. Stewart, *Collinearity and least squares regression*, Statist. Sci., 2 (1987), pp. 68–100.

[29] S. Van Huffel and J. Vandewalle, *The Total Least Squares Problem: Computational Aspects and Analysis*, SIAM, Philadelphia, PA, 1991.

[30] P. Vinsome, *Orthomin, an iterative method for solving sparse sets of simultaneous linear equations*, in Proceedings of the Fourth Symposium on Reservoir Simulation, Society of Petroleum Engineers of AIME, 1976, pp. 149–159.

[31] C. Vuik and H. A. van der Vorst, *A comparison of some GMRES-like methods*, Linear Algebra Appl., 160 (1992), pp. 131–162.

[32] H. F. Walker, *Implementation of the GMRES method using Householder transformations*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 152–163.

[33] H. F. Walker, *Implementations of the GMRES method*, Comput. Phys. Comm., 53 (1989), pp. 311–320.

[34] H. F. Walker and L. Zhou, *A simpler GMRES*, Numer. Linear Algebra Appl., 1 (1994), pp. 571–581.

[35] D. M. Young and K. C. Jea, *Generalized conjugate-gradient acceleration of nonsymmetrizable iterative methods*, Linear Algebra Appl., 34 (1980), pp. 159–194.

SIAM J. SCI. COMPUT.
Vol. 23, No. 5, pp. 1526–1548

A SEMI-LAGRANGIAN SEMI-IMPLICIT NUMERICAL METHOD FOR MODELS OF THE URINE CONCENTRATING MECHANISM*

ANITA T. LAYTON† AND HAROLD E. LAYTON‡

Abstract. Mathematical models of the urine concentrating mechanism consist of large systems of coupled differential equations. The numerical methods that have usually been used to solve the steady-state formulation of these equations involve implicit Newton-type solvers that are limited by numerical instability attributed to transient flow reversal. Dynamic numerical methods, which solve the dynamic formulation of the equations by means of a direction-sensitive time integration until a steady state is reached, are stable in the presence of transient flow reversal. However, when an explicit, Eulerian-based dynamic method is used, prohibitively small time steps may be required owing to the CFL condition and the stiffness of the problem. In this report, we describe a semi-Lagrangian semi-implicit (SLSI) method for solving the system of hyperbolic partial differential equations that arises in the dynamic formulation. The semi-Lagrangian scheme advances the solution in time by integrating backward along flow trajectories, thus allowing large time steps while maintaining stability. The semi-implicit approach controls stiffness by averaging transtubular transport terms in time along flow trajectories. For sufficiently refined spatial grids, the SLSI method computes stable and accurate solutions with substantially reduced computation costs.

Key words. mathematical models, differential equations, mathematical biology, kidney, renal medulla

AMS subject classifications. 65-04, 65M12, 65M25; 92-04, 92C35; 35-04, 35L45

PII. S1064827500381781

1. Introduction.

Mammals can produce urine that has a much higher osmolality than that of blood plasma.[1] This capability, which allows mammals to maintain a steady plasma osmolality during periods of water deprivation, is provided by the urine concentrating mechanism. Many model studies have sought to elucidate this mechanism, which is localized in the renal medulla and which depends on a countercurrent configuration of fluid flows in thousands of nearly parallel tubules [5, 6, 9, 12, 20, 23].

Models of the urine concentrating mechanism have usually been formulated as steady-state boundary-value problems involving differential equations expressing solute and water conservation. The solutions describe solute concentrations and fluid flow rates in interacting tubules having diameters of ~ 20 μm and lengths of several millimeters. Models are typically formulated in one space dimension corresponding to the axis parallel to intratubular flow.

The steady-state model problem consists of a nonlinear system of coupled, stiff ordinary differential equations (ODEs), which are usually solved by adaptations of Newton's method [16, 18, 20, 23]. However, this formulation has to contend with the difficulty that intratubular flow may be transiently reversed relative to the assumed steady-state direction. This transient reversal of flow direction, which arises when

*Received by the editors November 30, 2000; accepted for publication (in revised form) July 23, 2001; published electronically January 4, 2002. This work was supported in part by the National Institutes of Health (National Institute of Diabetes and Digestive and Kidney Diseases, grant DK-42091).

http://www.siam.org/journals/sisc/23-5/38178.html

†Department of Computer Science, University of Toronto, Toronto, ON, Canada M5S 3G4 (tam@cs.toronto.edu). Present address: Department of Mathematics, University of North Carolina, Chapel Hill, NC 27599-0001.

‡Department of Mathematics, Duke University, Durham, NC 27708-0320 (layton@math.duke.edu).

[1] Osmolality is approximated by a weighted sum of the solute concentrations [20].

transmural water fluxes are sufficiently large, may introduce numerical instability. In steady state, intratubular fluid flow in a tubule typically assumes a uniform direction, so flow reversal is a dynamic effect. However, iterates of a Newton-type method may pass through states that exhibit flow reversal, unless initial conditions that are sufficiently close to the steady-state solution are specified [5].

An alternative method for obtaining a solution to the steady-state equations is to formulate the problem in terms of its dynamic equations and then compute a steady-state solution to the dynamic equations. The dynamic equations form a nonlinear system of hyperbolic partial differential equations (PDEs), which represent solute conservation, and ODEs, which represent water conservation [5, 6]. A direction-sensitive time integration scheme, such as upwind differencing or the ENO (essentially nonoscillatory) scheme [2], has been used to advance the solution in time until a steady state is reached [5, 6]. These methods eliminate numerical instability arising from transient reversal of intratubular flow direction. However, owing to stiffness of the problem, which arises from transtubular transport terms, these explicit methods may require prohibitively small time steps, thus resulting in high computation cost.

The purpose of this paper is to describe a stable and efficient numerical method, based on the semi-Lagrangian semi-implicit (SLSI) scheme [13], for approximating solutions to the dynamic equations. The Lagrangian nature of this method avoids numerical instability arising from flow reversal, and its implicit nature controls stiffness and maintains stability even with large time steps.

In section 4, we present numerical results from the SLSI scheme that demonstrate second-order spatial convergence of the solution in a simple, one-solute model of the urine concentrating mechanism and in a two-solute model of the outer renal medulla, with parameters primarily from the rat. The one-solute model is an approximation to a model for which an explicit solution can be derived. In section 5, we compare the stability and efficiency of the SLSI method with the ENO method in models of the outer and inner renal medullas, and we show that the SLSI method is stable with a Courant–Friedrichs–Lewy (CFL) number much larger than 1, whereas the ENO method is limited by the CFL condition (i.e., the CFL number must be less than 1 to maintain numerical stability). Furthermore, compared with the ENO method, the SLSI method generates a solution with comparable accuracy in substantially less time, provided that the spatial grid is sufficiently refined.

2. Model equations.

In this section we introduce the equations for a simple dynamic model of the renal medulla. The model includes a single loop of Henle and a collecting duct. The loop of Henle has a descending limb and an ascending limb. The limbs of the loop of Henle and the collecting duct interact in a common tubular compartment, the central core. The central core represents all structures within the medulla but external to the loop of Henle and collecting duct, e.g., the interstitial spaces, interstitial cells, and vasculature. The central core formulation was introduced by Stephenson [15]. This model configuration is shown in Figure 2.1.

The model equations are based on conservation of solute and water in the renal medulla. A derivation of the equations can be found in [5]. The descending limb, ascending limb, collecting duct, and central core are represented by rigid tubules (indexed by $i = 1, 2, 3$, and 4, respectively) that extend in space from $x = 0$ to $x = L$. NaCl and urea are the two principal solutes of the renal medulla. For simplicity, we represent only the Cl^- and urea concentrations, denoted by $k = 1$ and 2, respectively.

The water flow rate at time t in a given tubule i is denoted by $F_{iV}(x, t)$, and the transmural (i.e., transtubular) water line flux (i.e., the water transport rate per

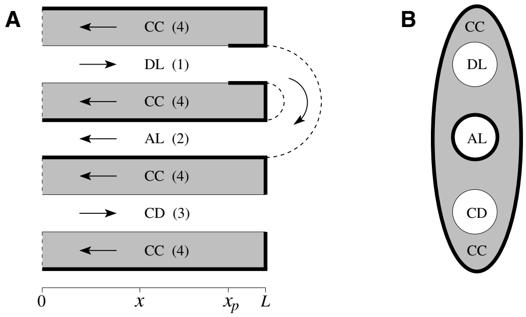


FIG. 2.1. *Model configuration. Panel* A: *Tubules along spatial axis. DL, descending limb* ($i = 1$)*; AL, ascending limb* ($i = 2$)*; CD, collecting duct* ($i = 3$)*; CC, central core* ($i = 4$)*; arrows, steady-state flow directions; thick lines, water-impermeable boundaries; prebend segment begins at* x_p (*see section* 4). *Panel* B: *Cross-section showing connectivity between CC and other tubules.*

unit tubular length) is given by $J_{iV}(x,t)$, taken positive for transport into the tubule. With this notation, water conservation, for $i = 1, 2, 3$, or 4, is represented by

$$\frac{\partial}{\partial x} F_{iV}(x,t) = J_{iV}(x,t). \tag{2.1}$$

Solute conservation is represented by

(2.2)
$$\frac{\partial}{\partial t} C_{ik}(x,t) = \frac{1}{\sqrt{A_i(x)}} \left(-F_{iV}(x,t) \frac{\partial}{\partial x} C_{ik}(x,t) + J_{ik}(x,t) - C_{ik}(x,t) J_{iV}(x,t) \right),$$

where $i = 1, 2, 3$, or 4 and $k = 1$ or 2. For a given tubule of type i, $C_{ik}(x,t)$ is the concentration of solute k, $A_i(x)$ is the cross-sectional area of the tubule, and $J_{ik}(x,t)$ is the transmural line flux of solute k, taken positive into the tubule. The three terms inside the outer parentheses on the right arise from axial intratubular solute convection, transmural solute transport, and transmural water transport, respectively. The first and third terms equal the spatial derivative of the solute flow, $F_{iV}(x,t)C_{ik}(x,t)$.

The transmural water flux across a descending limb, ascending limb, or collecting duct ($i = 1, 2$, or 3) is given by

$$J_{iV}(x,t) = 2\pi r_i(x) d_i(x) \sum_{k=1}^{2} \phi_k \sigma_{ik}(x) \left(C_{ik}(x,t) - C_{4k}(x,t) \right), \tag{2.3}$$

where $r_i(x)$ is the radius of the tubule, $d_i(x,y)$ is the product of the partial molar volume of water (0.018136 cm^3/mmole at 37°C) and the transmural osmotic water permeability $P_{f,i}(x)$, ϕ_k is the osmotic coefficient of solute k, and $\sigma_{ik}(x)$ is the reflection coefficient of solute k. For the central core, the equation for transmural water flux arises from the fluxes in (2.3) and is given by

$$J_{4V}(x,t) = -\left(J_{1V}(x,t) + J_{2V}(x,t) + J_{3V}(x,t) \right). \tag{2.4}$$

The transmural solute flux in a descending limb, ascending limb, or collecting duct ($i = 1, 2$, or 3) is given by

$$J_{ik}(x,t) = (1-\sigma_{ik}(x))J_{iV}(x,t)\left(C_{4k}(x,t)+C_{ik}(x,t)\right)/2$$
$$+\,2\pi r_i(x)\left(P_{ik}(x)\left(C_{4k}(x,t)-C_{ik}(x,t)\right)-\frac{V_{\max,ik}(x)C_{ik}(x,t)}{K_{M,ik}(x)+C_{ik}(x,t)}\right). \tag{2.5}$$

The first term on the right is the linearized solvent drag. The first term inside the pair of large parentheses is the transmural diffusion characterized by permeability $P_{ik}(x)$; the second term inside these parentheses represents active transport, characterized by maximum transport rate per unit tubular area $V_{\max,ik}(x)$ and Michaelis–Menten kinetics with Michaelis constant $K_{M,ik}(x)$. For the central core, the equation for solute flux arises from (2.5) and is given by

$$J_{4k}(x,t) = -\left(J_{1k}(x,t)+J_{2k}(x,t)+J_{3k}(x,t)\right). \tag{2.6}$$

To complete the system, boundary and initial conditions must be specified. At the entrances of the descending limb and collecting duct, $F_{1V}(0,t)$ and $F_{3V}(0,t)$, as well as $C_{1k}(0,t)$ and $C_{3k}(0,t)$ for each solute k, must be specified for $t \geq 0$. At the loop bend, the descending limb is continuous with the ascending limb; thus, $F_{2V}(L,t) = -F_{1V}(L,t)$, where the flow is taken positive in the increasing x direction; and for each solute k, $C_{2k}(L,t) = C_{1k}(L,t)$. The central core is assumed to be closed at $x = L$, which implies that there is no convective solute or fluid flow at $x = L$; thus, $F_{4V}(L,t) = 0$. Natural initial conditions are $C_{1k}(x,0) = C_{2k}(x,0) = C_{4k}(x,0) = C_{1k}(0,0)$, $C_{3k}(x,0) = C_{3k}(0,0)$, $F_{1V}(x,0) = -F_{2V}(x,0) = F_{1V}(0,0)$, $F_{3V}(x,0) = F_{3V}(0,0)$, and $F_{4V}(x,0) = 0$.

3. Discretization. In this section, we describe an algorithm to solve the model equations of section 2, (2.1)–(2.6), by a combination of the SLSI time discretization method and the trapezoidal rule. As previously noted, the goal is to develop an efficient numerical method that, like the Newton-type methods, controls the stiffness of the problem, and, like the explicit direction-sensitive integration schemes, maintains numerical stability in the presence of flow reversal.

The semi-Lagrangian treatment of advection, which is distinguished from the Lagrangian advection treatment by its backward, rather than forward, integration of flow trajectories, has generated considerable interest as a means for the efficient integration of equations arising in atmospheric models. Discretization schemes on the semi-Lagrangian treatment of advection offer the promise of larger time steps, with no loss in accuracy, in comparison with Eulerian-based advection schemes, in which the time-step size is limited by a more severe stability condition [11, 13].

In an Eulerian-based advection scheme, the observer is considered to be fixed at a position x as the world evolves. This scheme retains the regularity of the mesh (because the observer stays fixed), but small time steps may be required to satisfy the CFL condition and thus maintain numerical stability. In a Lagrangian-based scheme, the observer watches the world evolve while traveling along the trajectory of a fluid particle. This scheme is less restricted by stability requirements and therefore allows larger time steps than the Eulerian scheme. However, because the fluid particles move with time, an initially regularly spaced set of fluid particles generally becomes irregularly spaced as the system evolves.

A semi-Lagrangian advection scheme attempts to combine the advantages of both schemes—the grid regularity of the Eulerian scheme and the enhanced stability of the Lagrangian scheme—by approximating the Lagrangian derivative along trajectories defined by $dx/dt = u(x,t)$, where $u(x,t)$ denotes the flow velocity. In the model described in section 2, the flow velocity in a tubule of type i is given by $F_{iV}(x,t)/\sqrt{A_i(x)}$;

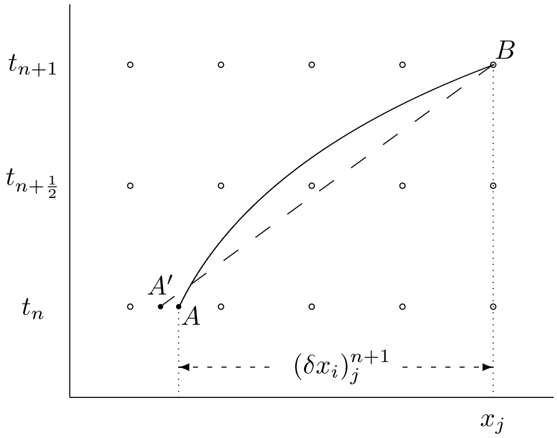


FIG. 3.1. *The semi-Lagrangian scheme approximates the Lagrangian derivative along the trajectory of a fluid particle. The curve AB denotes the actual trajectory, whereas the dashed line $A'B$ represents an approximate trajectory.*

therefore, the flow trajectory is defined by

$$\frac{d}{dt}x_i(t) = \frac{F_{iV}(x,t)}{\sqrt{A_i(x)}}. \tag{3.1}$$

Let Δt and Δx denote the time step and spatial grid interval, respectively, such that $\Delta t > 0$ and $\Delta x \equiv L/N$, where N is the number of spatial grid subintervals. Let $t_n \equiv n\Delta t$ be the nth time-level for $n = 0, 1, \ldots$, and let $x_j \equiv j\Delta x$ be the jth spatial grid point for $j = 0, 1, \ldots, N$. We adopt the two-time-level scheme[2] [13] that approximates a function associated with tubule i on a flow trajectory originating at the upstream departure point $(x_j - (\delta x_i)_j^{n+1})$ at t_n and terminating at the jth grid point x_j at t_{n+1}, where $\delta(x_i)_j^{n+1}$ denotes the displacement of the fluid particle in the time interval $[t_n, t_{n+1}]$. The trajectory is represented by the curve AB in Figure 3.1. For an arbitrary function $\psi(x,t)$, denote $\psi(x_j, t_n)$ by $(\psi)_j^n$. For a function ψ_i associated with tubule i, let $\widetilde{\psi}_i(x_j, t_n)$ (alternatively, $(\widetilde{\psi}_i)_j^n$) denote the function value at the departure point, i.e., $\psi_i(x_j - (\delta x_i)_j^{n+1}, t_n)$. Then the Lagrangian derivative of the solute concentration C_{ik} in (2.2) is approximated by

$$\left(\widetilde{\frac{d}{dt}C_{ik}}\right)_j^{n+\frac{1}{2}} = \left(\widetilde{\frac{\partial}{\partial t}C_{ik}}\right)_j^{n+\frac{1}{2}} + \left(\frac{\widetilde{F}_{iV}}{\sqrt{\widetilde{A}_i}}\widetilde{\frac{\partial}{\partial x}C_{ik}}\right)_j^{n+\frac{1}{2}} \approx \frac{(C_{ik})_j^{n+1} - (\widetilde{C}_{ik})_j^n}{\Delta t}. \tag{3.2}$$

The semi-Lagrangian approach provides accurate approximations for advection with virtually no time-step restriction. However, the transmural water and solute transport terms in (2.2) may render the equations stiff if the transmural permeabilities are sufficiently large. If these terms were treated explicitly, they would severely restrict the time step even with semi-Lagrangian advection approximations. Therefore, to obtain maximum benefit from the semi-Lagrangian approach, one needs to combine the semi-Lagrangian approximations with semi-implicit approximations. To this end,

[2] The alternative three-time-level semi-Lagrangian scheme approximates a trajectory originating at a spatial upstream departure point at time t_{n-1} and terminating at a spatial grid point at t_{n+1}.

the terms arising from water and solute transport in (2.2) are averaged in time along particle trajectories by

$$(\widetilde{C}_{ik}\widetilde{J}_{iV})_j^{n+\frac{1}{2}} \approx \frac{(C_{ik})_j^{n+1}(J_{iV})_j^{n+1} + (\widetilde{C}_{ik})_j^n(\widetilde{J}_{iV})_j^n}{2}, \qquad (\widetilde{J}_{ik})_j^{n+\frac{1}{2}} \approx \frac{(J_{ik})_j^{n+1} + (\widetilde{J}_{ik})_j^n}{2}. \tag{3.3}$$

3.1. Trajectory calculation. Before presenting the discretized form of (2.2), we describe how the upstream departure points and function values are computed. An approximate trajectory, represented by the dashed line $A'B$ in Figure 3.1, is estimated by a backward integration of (3.1). To obtain the trajectory velocities, we first compute intratubular water flows by numerically integrating (2.1) using the trapezoidal rule. For the descending limb and collecting duct ($i = 1$ or 3),

$$(F_{iV})_j^n = (F_{iV})_0^n + \frac{\Delta x}{2}\sum_{l=0}^{j-1}\left((J_{iV})_l^n + (J_{iV})_{l+1}^n\right). \tag{3.4}$$

For the ascending limb and central core ($i = 2$ or 4),

$$(F_{iV})_j^n = (F_{iV})_N^n - \frac{\Delta x}{2}\sum_{l=j}^{N-1}\left((J_{iV})_l^n + (J_{iV})_{l+1}^n\right), \tag{3.5}$$

where $(F_{2V})_N^n = -(F_{1V})_N^n$ (since $F_{2V}(L,t) = -F_{1V}(L,t)$) and $(F_{4V})_N^n = 0$ (since $F_{4V}(L,t) = 0$).

Using C_{ik} as an example, we now describe the computation of departure points and upstream function values. Flow trajectories are estimated by integrating (3.1) backward from the arrival point x_j at time t_{n+1} to the departure point $x_j - (\delta x_i)_j^{n+1}$ at time t_n by means of a second-order Runge–Kutta method:

$$(\delta x_i)_j^{n+1} \approx \frac{\Delta t}{2}\left(\frac{(F_{iV})_j^{n+1}}{\sqrt{(A_i)_j}} + \frac{F_{iV}(x_j - \Delta t(F_{iV})_j^{n+1}/\sqrt{(A_i)_j}, t_n)}{\sqrt{A_i(x_j - \Delta t(F_{iV})_j^{n+1}/\sqrt{(A_i)_j})}}\right). \tag{3.6}$$

The evaluation of the right side of (3.6) requires temporal extrapolation and spatial interpolation of F_{iV}, as will be explained shortly. The grid point values $(F_{iV})_j^n$ are given by (3.4) and (3.5). However, since the values of $(F_{iV})_j^{n+1}$, needed in the computation of $(\delta x_i)_j^{n+1}$ by (3.6), are unknown except possibly at the boundaries, they are extrapolated in time from known values at previous time steps:

$$(F_{iV})_j^{n+1} \approx 2(F_{iV})_j^n - (F_{iV})_j^{n-1}. \tag{3.7}$$

Another difficulty in estimating the trajectory arises from the second water flow term. Since the point $(x_j - \Delta t(F_{iV})_j^{n+1}/\sqrt{(A_i)_j}\ )$ will generally not coincide with a grid point, the value of $F_{iV}(x_j - \Delta t(F_{iV})_j^{n+1}/\sqrt{(A_i)_j}, t_n)$ may not be known. In that case, linear spatial interpolation based on the grid point values $(F_{iV})_j^n$ is used to approximate $F_{iV}(x_j - \Delta t(F_{iV})_j^{n+1}/\sqrt{(A_i)_j}, t_n)$ between grid points.

Once $(\delta x_i)_j^{n+1}$ is estimated, then if $x_j - (\delta x_i)_j^{n+1} \in [0, L]$, an approximation to C_{ik} at the departure points $(x_j - (\delta x_i)_j^{n+1}, t_n)$ can be computed from grid point values by cubic Lagrange spatial interpolation. In general, linear interpolation is sufficient for the computation of the flow trajectories [14, 17], and for the approximation of upstream function values it is economical to use cubic interpolation, which has fourth-order truncation error and little damping [25].

However, if an upstream departure point falls outside of the spatial domain (i.e., if $x_j - (\delta x_i)_j^{n+1} \notin [0, L]$), then the upstream function value must be estimated differently. The procedure depends on the type of the tubule and the position of the departure point. In the ascending limb, consider the case where $x_j - (\delta x_2)_j^{n+1} > L$, a likely case with a sufficiently large Δt since F_{2V} is negative with normal physiological parameters. Since the descending limb is continuous with the ascending limb at $x = L$, the upstream concentration in the ascending limb is given by $C_{2k}(x_j - (\delta x_2)_j^{n+1}, t_n) = C_{1k}(2L - x_j + (\delta x_2)_j^{n+1}, t_n)$. Similarly, for the descending limb, $C_{1k}^n(x_j - (\delta x_1)_j^{n+1}, t_n) = C_{2k}(2L - x_j + (\delta x_1)_j^{n+1}, t_n)$ if $x_j - (\delta x_1)_j^{n+1} > L$.

For the collecting duct and the central core, function values at departure points with values greater than x_N are estimated with cubic Lagrange extrapolation using function values at x_{N-3}, x_{N-2}, x_{N-1}, and x_N at time-level t_n. (Since normally F_{1V} and F_{3V} are positive at steady state, the associated departure points fall at values greater than x_N only in transient states.) Cubic Lagrange extrapolation is also used to approximate upstream function values associated with the ascending limb and central core if $x_j - (\delta x_i)_j^{n+1} < 0$; function values at x_0, x_1, x_2, and x_3 at $t = t_n$ are used.

If a departure point falls at a value less than x_0 in a descending limb or collecting duct, then the Lagrangian derivative approximation (3.2) and the semi-implicit approximation (3.3) are cast in terms of their respective boundary values at x_0. The upstream function value $(\widetilde{C}_{ik})_j^n$ is linearly extrapolated along the flow trajectory, using downstream and boundary function values, to give

$$\widetilde{C}_{ik}(x_j, t_n) = (1 - (\alpha_i^*)_j^{n+1}) C_{ik}(0, (\tau_i)_j^{n+1}) + (\alpha_i^*)_j^{n+1} C_{ik}(x_j, t_{n+1}), \tag{3.8}$$

where

$$(\alpha_i^*)_j^{n+1} \equiv \frac{x_j - (\delta x_i)_j^{n+1}}{x_j} \quad \text{and} \quad (\tau_i)_j^{n+1} \equiv t_n - \frac{(\alpha_i^*)_j^{n+1} \Delta t}{1 - (\alpha_i^*)_j^{n+1}}. \tag{3.9}$$

Figure 3.2 illustrates the relationship among $(\alpha_i)_j^{*n+1}$, $(\tau_i)_j^{n+1}$, and $C_{ik}(0, (\tau_i)_j^{n+1})$.

For an arbitrary function ψ_i associated with a tubule i, redefine $(\widetilde{\psi}_i)_j^n$ to be $(1 - (\alpha_i^*)_j^{n+1})\psi_i(0, (\tau_i)_j^{n+1})$ if $x_j - (\delta x_i)_j^{n+1} < 0$ and $i = 1$ or 3. Also, define the boundary correction factor $(\alpha_i)_j^{n+1}$ by

$$(\alpha_i)_j^{n+1} = \begin{cases} (\alpha_i^*)_j^{n+1}, & x_j - (\delta x_i)_j^{n+1} < 0 \text{ and } i = 1 \text{ or } 3, \\ 0 & \text{otherwise.} \end{cases} \tag{3.10}$$

With this notation, the Lagrangian derivative approximation (3.2) and the semi-implicit approximation (3.3) for C_{ik} can be rewritten to take into account upstream points that fall outside the spatial domain:

$$\left(\widetilde{\frac{d}{dt} C_{ik}} \right)_j^{n+\frac{1}{2}} \approx \frac{(1 - (\alpha_i)_j^{n+1})(C_{ik})_j^{n+1} - (\widetilde{C}_{ik})_j^n}{\Delta t}, \tag{3.11}$$

$$(\widetilde{C}_{ik})_j^{n+\frac{1}{2}} \approx \frac{1}{2} \left((1 + (\alpha_i)_j^{n+1})(C_{ik})_j^{n+1} + (\widetilde{C}_{ik})_j^n \right). \tag{3.12}$$

Equations (3.11) and (3.12) reduce to (3.2) and (3.3) for C_{ik}, respectively, when $0 \le x_j - (\delta x_i)_j^{n+1} \le L$. Approximations for $(\widetilde{J}_{iV})_j^n$ and $(\widetilde{J}_{ik})_j^n$ can be treated similarly.

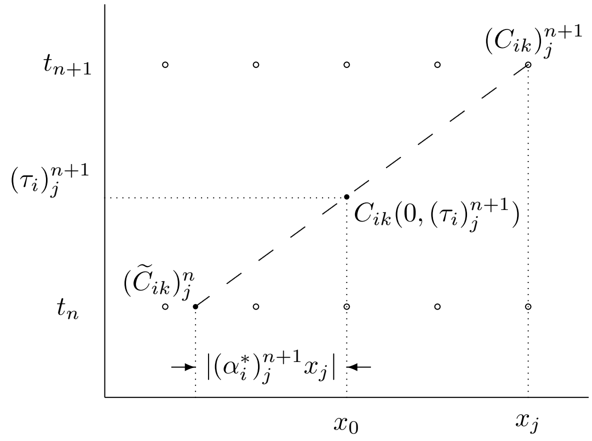

t_{n+1}
$(C_{ik})_j^{n+1}$
$(\tau_i)_j^{n+1}$
$C_{ik}(0,(\tau_i)_j^{n+1})$
$(\widetilde{C}_{ik})_j^n$
t_n
$|(\alpha_i^*)_j^{n+1}x_j|$
x_0
x_j

FIG. 3.2. *A schematic diagram that illustrates relation* (3.8).

3.2. Numerical procedures. When discretized in time using the two-level SLSI scheme and evaluated at the grid points $\{x_j\}$, (2.2) takes the following form:

$$\frac{(1-(\alpha_i)_j^{n+1})(C_{ik})_j^{n+1}-(\widetilde{C}_{ik})_j^n}{\Delta t} = \frac{1}{2}\left(\frac{(1+(\alpha_i)_j^{n+1})(J_{ik})_j^{n+1}}{\sqrt{(A_i)_j}} + \frac{(\widetilde{J}_{ik})_j^n}{\sqrt{(\widetilde{A}_i)_j}} - \frac{(1+(\alpha_i)_j^{n+1})(C_{ik})_j^{n+1}(J_{iV})_j^{n+1}}{\sqrt{(A_i)_j}} - \frac{(\widetilde{C}_{ik})_j^n(\widetilde{J}_{iV})_j^n}{\sqrt{(\widetilde{A}_i)_j}}\right). \tag{3.13}$$

To obtain (3.13), the Lagrangian derivative dC_{ik}/dt was approximated by (3.11), and the transmural flux term J_{ik} and the water transport term $C_{ik}J_{iV}$ were averaged in time using approximations analogous to (3.12). Equations (2.3)–(2.6) and (3.13) form a nonlinear, coupled system, which we refer to as the *nonlinear kernel* and which we solve by means of Gauss–Seidel iteration. Specifically, at the mth iteration, we solve the following equations, obtained by rearranging (3.13), so that the unknown terms with $(C_{ik})_j^{n+1}$ are on the left and the other terms are on the right:

$$\left((1-\ (\alpha_i)_j^{n+1}) + \frac{\Delta t}{2\sqrt{(A_i)_j}}(J_{iV})_j^{n+1,[m-1]}\right)(C_{ik})_j^{n+1,[m]}$$
$$= (\widetilde{C}_{ik})_j^n + \frac{\Delta t}{2}\left((1+(\alpha_i)_j^{n+1})\frac{(J_{ik})_j^{n+1,[m-1]}}{\sqrt{(A_i)_j}} + \frac{(\widetilde{J}_{ik})_j^n}{\sqrt{(\widetilde{A}_i)_j}} - \frac{(\widetilde{C}_{ik})_j^n(\widetilde{J}_{iV})_j^n}{\sqrt{(\widetilde{A}_i)_j}}\right). \tag{3.14}$$

A summary of the integration algorithm and the iterative process is given at the end of this subsection. The Gauss–Seidel iteration is repeated until the iterates have sufficiently converged, that is, until the max-norm of successive iterations is less than a given tolerance TOL. Since our goal is to compute a steady-state solution, computation costs are reduced by adopting a larger TOL at the onset of the simulation when the system is far from its steady state and then refining the accuracy by reducing TOL when the solution is approaching a steady state. How far a given solution at time-level t_n differs from the steady-state solution is measured by computing the difference $\max_{i,k} |C_{ik}^n - C_{ik}^{n-1}|$, denoted by ΔC^n, from the previous solution at t_{n-1}. Thus, we choose the Gauss–Seidel tolerance dynamically as $\mathrm{TOL} = \min(\mathrm{TOL}_1, \max(0.1 \times \Delta C^n, \mathrm{TOL}_2))$, where $\mathrm{TOL}_1 > \mathrm{TOL}_2$.

In our implementation, a linear extrapolation, using values from the previous two time-levels, $C_{ik}^{n+1,[0]} = 2C_{ik}^n - C_{ik}^{n-1}$, is used as an initial guess for the Gauss–Seidel iterates. With this choice of initial guess, and with $\mathrm{TOL}_1 = 10^{-3}$ and $\mathrm{TOL}_2 = 10^{-9}$, one or two Gauss–Seidel iterations suffice to attain a level of accuracy in the steady-state solution that is not improved by increasing the number of iterations. Because the convergence is rapid, we have not used a more rapidly converging but more costly nonlinear solver, e.g., Newton's method, for the nonlinear kernel.

Now we summarize the steps that advance the model by one time step. Suppose that the boundary conditions and the initial conditions $C_{ik}(x_j, 0)$ and $F_{iV}(x_j, 0)$, specified in section 2, are given. Then, until a steady state is reached, repeat steps 1 to 4:

1. Estimate the departure points for each tube using (3.6).
2. Approximate upstream function values using cubic Lagrange approximations.
3. Solve the nonlinear kernel using Gauss–Seidel iterations. Compute an initial guess $C_{ik}^{n+1,[0]}$, using linear extrapolation, and the dynamic Gauss–Seidel tolerance TOL; then compute the iterates (indexed by m) until the convergence criterion is met:
   (a) For each solute k,
      i. compute $(C_{ik})_j^{n+1,[m]}$, $(C_{2k})_j^{n+1,[m]}$, and $(C_{3k})_j^{n+1,[m]}$ using (3.14);
      ii. update water fluxes $(J_{iV})_j^{n+1,[m]}$ for $i = 1, 2, 3$, and 4 using (2.3) and (2.4).
   (b) Update solute fluxes $(J_{ik})_j^{n+1,[m]}$ for $i = 1, 2, 3$, and 4 using (2.5) and (2.6).
   (c) Compute $(C_{4k})_j^{n+1,[m]}$ using (3.14).
4. Update $(F_{iV})_j^{n+1}$ using (3.4) and (3.5).

4. Convergence properties.

In this section, we present numerical results that demonstrate the second-order spatial convergence properties of the SLSI method. First, we describe a simple steady-state model that has an explicit solution in terms of solute transport. Then we formulate the parameters of the model equations developed in section 2 to approximate those of the simple model, and we compare results from the two models. In the second study, we use more realistic model parameters that are based mostly on the outer medulla of the rat. Calculations in sections 4 and 5 were performed using Fortran programs in double precision on a Dell Dimension XPS T550 system with an Intel Pentium III 550 MHz processor and with 256 MB of RAM.

4.1. A simple steady-state model with explicit solution.

We now restrict our attention to a simple, steady-state, one-solute model. This model, which is based on the structure and geometry described in section 2 (see Figure 2.1), represents a descending limb, ascending limb, collecting duct, and central core. The model descending limb is structurally and functionally divided into two segments. The second segment, which we call the *prebend* segment, is a terminal portion of the descending limb that may have the diameter and transport properties of the ascending limb. The model prebend segment begins at $x_p \in [0, L]$.

We let $C_i(x)$ be the steady-state solute concentration in a tubule i. We assume that the descending limb, except for its prebend segment, and the collecting duct are infinitely permeable to water. The prebend segment and the ascending limb are assumed to be impermeable to water. With these assumptions, which are suggested by permeabilities that have been measured in the outer medulla of the rat kidney, we have $C_3(x) = C_4(x) \equiv C(x)$ and $C_1(x) = C(x)$ for $x \in [0, x_p]$; $J_{2V}(x) = 0$ and $J_{1V}(x) = 0$ for $x \in (x_p, L]$. We define $F_{ik}(x) \equiv F_{iV}(x)C_i(x)$ to be the rate of solute advection ("solute flow") in a tubule of type i, with $k = 1$ in this one-solute model.

With these stipulations, $C(x)$, the steady-state concentration at level x in the initial descending limb ($x \le x_p$), the collecting duct, and the central core is given by

$$\frac{C(x)}{C(0)} = \exp\left(\int_0^x \frac{-H^+_{x_p}(y)J_{1k}(y) - J_{2k}(y)}{H^-_{x_p}(y)F_{1k}(y) + F_{3k}(L) + \int_y^L (J_{1k}(s) + J_{2k}(s))\, ds}\, dy\right), \tag{4.1}$$

where $H^+_{x_p}(x)$ and $H^-_{x_p}(x)$ denote Heaviside functions such that $H^+_{x_p}(x) = 1$ if $x > x_p$, $H^+_{x_p}(x) = 0$ if $x \le x_p$, and $H^-_{x_p}(x) = 1 - H^+_{x_p}(x)$. $H^-_{x_p}(x)F_{1k}(x)$ represents the solute flow in the proximal, water-permeable portion of the descending limb; $H^+_{x_p}(x)J_{1k}(x)$ represents the transmural solute flux across the prebend segment. Equation (4.1) gives the steady-state concentration profile as a function only of the boundary concentration $C(0)$, the solute flow along the descending limb, the solute flow out of the collecting duct, and the transmural fluxes of solute across the descending and ascending limbs. A derivation for (4.1) can be found in the appendix. If one further assumes that $J_{1k}(x) = 0$ for $x \le x_p$, then $F_{1k}(x) = F_{1k}(0)$ for $x \le x_p$, and the integral can be evaluated analytically. If one specifies that $F_{3k}(L) = \epsilon F_{1k}(0)$, one obtains

$$\frac{C(x)}{C(0)} = \begin{cases} f(1, x, 0), & 0 \le x \le x_p, \\ f(1, x_p, 0)f(0, x, x_p), & x_p < x \le L, \end{cases} \tag{4.2}$$

where

$$f(\alpha, x, y) \equiv \frac{(\alpha + \epsilon)F_{1k}(0) + \int_x^L (J_{1k}(s) + J_{2k}(s))\, ds}{(\alpha + \epsilon)F_{1k}(0) + \int_y^L (J_{1k}(s) + J_{2k}(s))\, ds}. \tag{4.3}$$

For this model, we assume that $L = 2$ mm, $x_p = 0.9\ L$, and that each tubule has a fixed diameter of 20 μm. We assume that all tubules are impermeable to solute; thus $P_{ik}(x) = 0$ for each i. For the collecting duct, we further assume that $J_{3k}(x) = 0$. The boundary concentrations and water flows are specified for both the descending limb and collecting duct: $C_1(0) = C_3(0) = 200$ mM, $F_{1V}(0) = 10$ nl/min, and $F_{3V}(0) = 5$ nl/min (thus, $\epsilon = 0.5$). For the ascending limb and the prebend segment, we assume a constant rate of solute transport such that half of the solute entering the descending limb is transported out of the ascending limb and the prebend segment: $J_{2k}(x) = -83.\bar{3}$ nmole/(cm^2·s) and, for $x \in (x_p, L]$, $J_{1k}(x) = -83.\bar{3}$ nmole/(cm^2·s).

We approximated this simple steady-state model with the dynamic equations (2.1)–(2.6), which were integrated in time using the SLSI method described in section 3 until a steady state was attained. The derivation of (4.2) assumes infinite water permeability for the descending limb and collecting duct. However, in our approximation to this model, water permeabilities must be finite and were taken to be 5,000 μm/s for both the descending limb and collecting duct. As water permeability increases, the numerical results should approach (4.2). Nonetheless, a value of

5,000 μm/s should be sufficiently large so that (4.2) can be used as the reference solution in this convergence study. The osmotic coefficient ϕ was set to 1.84 for NaCl [22], and all reflection coefficients σ_i were set to 1.

The two portions of the descending limb have markedly different transport properties. To maintain good spatial convergence near the point where tubular properties change, we represent the transition by means of a twice-differentiable polynomial. Thus, suppose that at $x = \alpha_0$ the value of a transport property $q(x)$ (e.g., a solute permeability $P_{ik}(x)$ or radius $r_i(x)$) changes abruptly from q_1 to q_2. Then q is represented by a piecewise cubic polynomial

$$q(x) = \begin{cases} q_1, & x < \alpha_1, \\ -2(x-\alpha_1)^3 \Delta q/b^3 + 3(x-\alpha_1)^2 \Delta q/b^2 + q_1, & \alpha_1 \le x \le \alpha_2, \\ q_2, & x > \alpha_2, \end{cases} \tag{4.4}$$

where $b > 0$ specifies the sharpness of the transition, $\alpha_1 \equiv \alpha_0 - b/2$, $\alpha_2 \equiv \alpha_0 + b/2$, and $\Delta q \equiv q_2 - q_1$. Because the trapezoidal rule (the spatial integration scheme used in this method) assumes a continuous second derivative within each integration subinterval, $q(x)$ is designed to satisfy this assumption. In this model, α_1 and α_2 are grid points, and α_0 is identified with the beginning of the prebend segment; thus the changes in water permeability and active transport rate of the descending limb are near $\alpha_0 = x_p$. In the spatial convergence experiments presented in this section, we set $b = 0.05L$. (Experiments have shown that tubular diameters may also change, but for simplicity, tubular diameters are assumed to be uniform for these calculations.)

Results of the numerical experiments were normalized and are presented in dimensionless form. Dimensionless variables were obtained by dividing dimensional variables by dimensional reference values. Specifically, the spatial variable x was normalized by dividing by the dimensional length of the renal medulla, L; C_{ik} by the solute concentration in fluid entering the descending limb, C_o (if there is more than one solute in the system, C_o is chosen to be the chloride concentration); F_{iV} by the axial water flow rate entering the descending limb, F_{Vo}; A_i by the cross-sectional area of a descending limb at $x = 0$, A_o; t by $A_o L/F_{V_o}$; d_i by $F_{Vo}/(C_o L)$; P_{ik} by $F_{Vo}/(2L\sqrt{\pi A_o})$; $V_{\max,ik}$ by $C_o F_{Vo}/(2L\sqrt{\pi A_o})$; and $K_{M,ik}$ by C_o.

4.2. Convergence studies and comparison with the explicit model. In these convergence studies, we will first examine quantities that have been commonly reported in numerical studies of the urine concentrating mechanism. These quantities are the collecting duct outflow variables, the central core concentration at the medullary tip ($x = L$), and the water and solute balance properties of the medulla [4, 5, 6, 19, 21, 24]. The collecting duct outflow variables are of particular interest because the urine osmolality, urine solute concentration, and urine flow are key scientific results, and because the collecting duct fluid and solute outflow, being a small fraction of typical tubular flow, can be sensitive to changes in parameters or to the degree of spatial grid refinement. We will also present global measures of convergence in standard norms for key variables. Unless otherwise specified, a dimensionless time step Δt of 0.0002 was used in the numerical calculations.

The convergence of numerical results was assessed by comparing numerical solutions computed on successively refined spatial grids. Empirical orders of convergence were obtained by means of a diagnostic calculation based on Aitken extrapolation [1], which we now describe. Let ψ be a scalar dependent variable of the model at steady state, and let ψ_m be a numerical approximation to ψ computed on a spatial grid of size $N = N_0 2^m$, where N_0 is a positive integer and $m = 0, 1,$ or 2. An empirical order

TABLE 4.1

Spatial convergence results for the steady-state central core concentration $(C_4)_N^{ss}$ and collecting duct water flow rate $(F_{3V})_N^{ss}$ at $x = L$. p, C_{4_∞}, and F_{3V_∞} are computed by means of (4.5) *using the three steady-state values on and above the line containing p.*

N	$(C_4)_N^{ss}$	p	C_{4_∞}	$(F_{3V})_N^{ss}$	p	F_{3V_∞}
40	1.82051	–	–	2.74363E-1	–	–
80	1.79050	–	–	2.79281E-1	–	–
160	1.78395	2.20	1.78214	2.80495E-1	2.03	2.80892E-1
320	1.78332	3.40	1.78257	2.80677E-1	2.74	2.80709E-1
640	1.78322	2.65	1.78320	2.80719E-1	2.12	2.80731E-1

TABLE 4.2

Convergence study for increasing water permeabilities (P_f). Steady-state solute concentrations $(C_4)_N^{ss}$ of the central core at $x = L$ are shown for spatial grid refinements $N =$ 80, 160, and 320.

N	P_f (10^4 μm/s)				
	1.25	2.5	5.0	10.0	20.0
80	1.86317	1.86189	**1.86126**	1.86094	1.86077
160	1.86029	1.85594	1.85051	**1.84823**	1.84567
320	1.86065	1.85658	1.85286	1.84965	**1.84705**

of convergence p of three strictly monotonic values ψ_0, ψ_1, and ψ_2 may be obtained by solving simultaneously the equations

$$\psi_\infty - \psi_m = K(\Delta x/2^m)^p, \qquad m = 0, 1, 2. \tag{4.5}$$

The empirical order of convergence is $p = \log_2(\Delta_1/\Delta_2)$, where $\Delta_m = \psi_m - \psi_{m-1}$. The asymptotic error constant is estimated by $K = \Delta_1(\Delta x)^{-p}(1-2^{-p})^{-1}$. The value to which the ψ_m's are apparently converging is $\psi_\infty = \psi_0 + K(\Delta x)^p$.

In Table 4.1, the spatial convergence of the steady-state solute concentration in the central core and the water flow rate in the collecting duct at $x = L$ were estimated using (4.5). The empirical orders of convergence are consistent with the expected second-order convergence. Similar results were obtained for collecting duct solute outflow $(F_{31})_N^{ss}$. Generally speaking, the orders of convergence for water and solute flow rates are more regularly second order than are orders of convergence for other variables. This is because the principal equations (2.1) and (2.2) are derived on the basis of water and solute (F_{iV} and F_{ik}) conservation; consequently, the convergence for these variables should be robustly second order. The convergence for other variables, such as solute concentration, $C_i = F_{ik}/F_{iV}$, may deviate somewhat from second order.

Using grid sizes $N =$ 160, 320, and 640, the approximations $(C_4)_N^{ss}$ for $C_4(L)$ converge to a concentration of ~1.78320, which is within 3.20% of the corresponding value in the explicit solution, 1.84211; the approximations $(F_{3V})_N^{ss}$ for $F_{3V}(L)$ converge to a flow of ~0.280731, which is within 3.43% of the explicit solution of 0.27143. The discrepancy between the explicit and the numerical solutions, after sufficient grid refinement, arises from the use of finite water permeabilities and from the nonzero transition length at the beginning of the prebend segment.

To provide evidence that the model indeed converges to the explicit solution when model water permeability values approach infinity and the prebend transition length approaches zero, we show in Table 4.2 steady-state values of $(C_4)_N^{ss}$ computed for water permeability values starting at $P_f = 1.25 \times 10^4$ μm/s and increasing by factors of 2, 4, 8, and 16 for grid sizes of $N =$ 80, 160, and 320. A smaller prebend transition

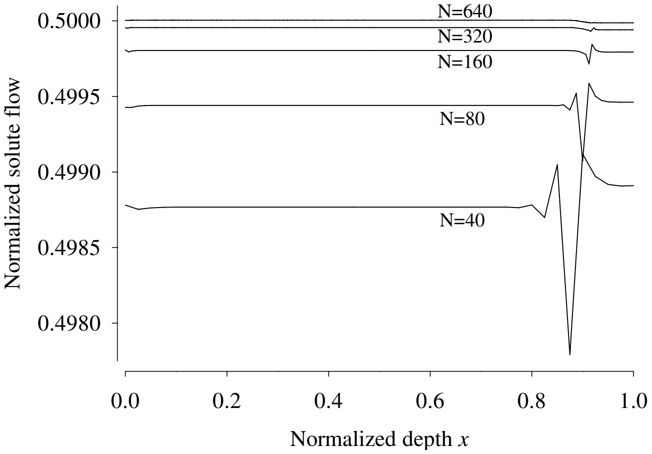


FIG. 4.1. *Net steady-state chloride flow rate F_1^{ss} through the medulla, computed with successive spatial grid refinements. Second-order convergence to the correct normalized solute flow,* 0.5, *is apparent for $N > 80$.*

length of $b = 0.02L$ was used, compared to the base-case value of $b = 0.05L$ used in the spatial convergence studies. Note that for a given N, the solution decreases toward the explicit model value of 1.84211 as water permeability increases. However, stronger evidence for convergence can be obtained by increasing the water permeability and the spatial grid resolution simultaneously. This is because as the water permeability is increased, the errors in computing water efflux from the water-permeable tubules near $x = 0$ (a boundary effect) and x_p (beginning of prebend segment) also increase unless the grid is refined. A convergence study based on the diagonal concentrations given in bold typeface in Table 4.2 indicates that the concentrations are converging to a value of 1.84693, which differs from the explicit model value by a relative difference of 0.26%. For the studies in the remainder of this section, the base-case values $P_f = 5{,}000$ μm/s and $b = 0.05L$ were used.

Numerical solutions to mathematical models of the urine concentrating mechanism have frequently been assessed on the basis of their mass conservation properties [6, 8, 21]. To assess solute and water conservation of the model, define $F_k^{ss}(x)$ and $F_V^{ss}(x)$ to be the total steady-state flow rates of solute and water through the renal medulla. Then,

$$F_k^{ss}(x) = \sum_{i=1}^{4} F_{iV}^{ss}(x) C_i^{ss}(x) \qquad \text{and} \qquad F_V^{ss}(x) = \sum_{i=1}^{4} F_{iV}^{ss}(x), \tag{4.6}$$

where $F_{iV}^{ss}(x)$ and $C_i^{ss}(x)$ are the steady-state flow and solute concentration in an individual tubule i. At steady state, both F_k^{ss} and F_V^{ss} are constant functions. Since the collecting duct is the only tubule that is open at $x = L$, F_k^{ss} and F_V^{ss} will equal the solute and water flow rates there, respectively; that is, $F_k^{ss} \equiv F_{3k}^{ss}(L)$ and $F_V^{ss} \equiv F_{3V}^{ss}(L)$. Numerical results should approximate these properties.

Since the collecting duct is assumed to be solute-impermeable in this model, $F_1^{ss} = F_{31}^{ss}(L) = F_{31}(0) = 0.5$. Indeed, this is the value that the curves in Figure 4.1 are approaching, with a convergence rate that is approximately second order. At low resolution, there is a substantial defect around $x = x_p$, where the transmural

Table 4.3

Global spatial convergence results computed for the water flow $\vec{F}_V$, chloride flow $\vec{F}_1$, and chloride concentration $\vec{C}$, using the L_1-norm, L_2-norm, and L_∞-norm. The orders of convergence are shown in columns p_1, p_2, and p_∞.

		$\vec{F}_V \equiv (F^{ss}_{1V}, F^{ss}_{2V}, F^{ss}_{3V}, F^{ss}_{4V})$				
N	r_1	p_1	r_2	p_2	r_∞	p_∞
80	3.54526E-3	–	4.85106E-3	–	3.22711E-3	–
160	9.67002E-4	1.88	1.20676E-3	2.01	7.98999E-4	2.01
320	2.07363E-4	2.22	2.29190E-4	2.40	1.77574E-4	2.17
640	5.23266E-5	1.97	5.46919E-5	2.07	4.66975E-5	1.93
		$\vec{F}_1 \equiv (F^{ss}_{11}, F^{ss}_{21}, F^{ss}_{31}, F^{ss}_{41})$				
N	r_1	p_1	r_2	p_2	r_∞	p_∞
80	1.53426E-3	–	1.65883E-3	–	2.80338E-3	–
160	5.30343E-4	1.53	5.65183E-4	1.55	9.51475E-4	1.54
320	1.86390E-4	1.51	1.95506E-4	1.53	2.48346E-4	1.94
640	5.51589E-5	1.76	5.77176E-5	1.76	6.75311E-5	1.88
		$\vec{C} \equiv (C^{ss}_1, C^{ss}_2, C^{ss}_3, C^{ss}_4)$				
N	r_1	p_1	r_2	p_2	r_∞	p_∞
80	2.78896E-3	–	5.96690E-3	–	1.68521E-2	–
160	5.55048E-4	2.32	1.22541E-3	2.28	3.67474E-3	2.20
320	4.02216E-5	3.78	1.04542E-4	3.55	5.01056E-4	2.87
640	8.97683E-6	2.16	1.68173E-5	2.64	9.16320E-5	2.45

properties of the descending limb change. Nevertheless, the curves flatten as the resolution is increased. The steady-state value of F^{ss}_V depends on the steady-state tubular concentration and thus is not known a priori. The curves for F^{ss}_V (not shown) show no derivation from horizontal lines even at a low resolution of $N = 40$, and they show an approximately second-order convergence to a steady-state value of ~0.28073. The solution to the explicit model yields a value of $F^{ss}_V = F^{ss}_{31}(L)/C^{ss}(L) = 0.27143$.

Table 4.3 shows global convergence in norm for steady-state water flow $\vec{F}_V$, chloride flow $\vec{F}_1$, and chloride concentration $\vec{C}$. The ordering of the entries of the vector $\vec{F}_V \in R^{4(N+1)}$ is indicated in the table, where each F^{ss}_{iV} represents the $N+1$ components $(F^{ss}_{iV})_j$, $j = 0, \dots, N$. The vectors $\vec{F}_1$ and $\vec{C}$ are defined similarly. For a vector $\vec{F}_V$ computed on N subintervals and denoted by $\vec{F}^N_V$, we define $P_{N/2}$: $R^{4(N+1)} \to R^{4(N/2+1)}$ to be the operator that reduces $\vec{F}^N_V$ to the vector $P_{N/2}\vec{F}^N_V$ by retaining the $N/2+1$ entries $(F^{ss}_{iV})_0$, $(F^{ss}_{iV})_2$, $(F^{ss}_{iV})_4, \dots, (F^{ss}_{iV})_N$ in F^{ss}_{iV} for $i = 1, 2, 3$, and 4. We index the components $(\vec{F}_V)_j$ of $\vec{F}^N_V \in R^{4(N+1)}$ by $j = 1, 2, \dots, M$, where $M = 4(N+1)$, and we define the q-norm of $\vec{F}_V$ by

$$\|\vec{F}^N_V\|_q \equiv \left(\sum_{j=1}^{M} |(\vec{F}_V)_j|^q \frac{1}{M} \right)^{1/q}, \qquad q = 1, 2, \text{ or } \infty. \tag{4.7}$$

If we define $r^N_q = \|P_{N/2}\vec{F}^N_V - \vec{F}^{N/2}_V\|_q / \|\vec{F}^{640}_V\|_q$, then an order of convergence, analogous to that previously obtained by the Aitken method, can be estimated by $p^N_q = \log_2(r^{N/2}_q / r^N_q)$. As in the case for the results in Table 4.1, convergence in Table 4.3 was found to be approximately second order for $q = 1$, 2, and ∞.

The ratio r^N_q has a natural interpretation as an estimate for the relative error $e^N_q \equiv \|\vec{F}^A_V - \vec{F}^N_V\|_q / \|\vec{F}^A_V\|_q$, where $\vec{F}^A_V$ is the vector, analogous to $\vec{F}^{ss}_V$, of values at the spatial grid points $\{x_j\}^N_{j=0}$ of the true (analytic) steady-state solution to the

Table 4.4

Tubular diameters and transport parameters used in the outer medulla model. DL, descending limb; AL, ascending limb; CD, collecting duct; P_f*,* P_{Cl^-}*,* P_{urea}*, permeabilities of water, Cl*$^-$*, and urea, respectively;* $V_{\max,Cl^-}$*, maximum Cl*$^-$ *active transport rate.*

Property	DL	AL	CD
Diameter (μm)	15	20	26
P_f (μm/s)	2,295	0	445
P_{Cl^-} (10^{-5} cm/s)	4.8	1	0.39
P_{urea} (10^{-5} cm/s)	13	1	3.5
$V_{\max,Cl^-}$ (nmole/(cm^2·s))	0	8	0

finite-permeability model equations. This is because $\|\vec{F}_V^A\|_q \approx \|\vec{F}_V^{640}\|_q$; $\|\vec{F}_V^{640}\|_q \gg \|P_{N/2}\vec{F}_V^N - \vec{F}_V^{N/2}\|_q$; and $\|P_{N/2}\vec{F}_V^N - \vec{F}_V^{N/2}\|_q \approx \|P_{N/2}\vec{K}^N(N^{-p} - (2N)^{-p})\|_q \approx (1 - 2^{-p})\|\vec{K}^N N^{-p}\|_q \approx (1 - 2^{-p})\|\vec{F}_V^A - \vec{F}_V^N\|_q$, by analogy with (4.5), for an asymptotic error vector $\vec{K}^N \in R^{4(N+1)}$. Thus, if $p \approx 2$, then $e_q^N \approx \frac{4}{3}r_q^N$.

4.3. An outer medulla model. The physiological setting in which we wish to apply the SLSI method is substantially more complex than the simple explicit model, and the approximation to that model, described in the previous section. Therefore, in a second set of experiments, we tested the SLSI method using parameter values based on experiments in the outer medullas of small mammals (primarily rat, but also hamster and rabbit [3, 7]). These parameter values are listed in Table 4.4. The medullary length L was set to be 3.2 mm. The transition in the late descending limb, which includes changes in tubular diameter and transport properties, is treated using the piecewise cubic function defined in (4.4). The boundary concentrations were chosen for $C_{11}(0,t)$, $C_{12}(0,t)$, $C_{31}(0,t)$, and $C_{32}(0,t)$ to be 160.0, 15.0, 63.8, and 197.5 mM, respectively; the boundary flows were chosen for $F_{1V}(0,t)$ and $F_{3V}(0,t)$ to be 10 and 1.5 nl/min, respectively. The osmotic coefficient ϕ_k for NaCl ($k = 1$) was set to be 1.84, and for urea ($k = 2$), 0.97 [20, 22]. The active transport rates for urea, $V_{\max,i2}$, were all set to zero, and the Michaelis constant $K_{M,ik}(x)$ for chloride ($k = 1$) was set to 70 mM. The reflection coefficients σ_{ik} were set to 1.

Figure 4.2 shows the normalized steady-state osmolality profiles for the four tubules, computed using the parameters shown in Table 4.4. The normalized osmolality of a given tubule i at level x ($i = 1, 2, 3$, or 4) was defined by

$$C_{iO}(x,t) = \frac{\phi_1 C_{i1}(x,t) + \phi_2 C_{i2}(x,t)}{\phi_1 C_{11}(0,t) + \phi_2 C_{12}(0,t)}. \tag{4.8}$$

Convergence results for the rate of steady-state chloride and water flow out of the collecting duct are shown in Table 4.5. Results for both variables indicate second-order convergence, as do results for other variables, although values for F_{3V} obtained using $N = 40$, 80, and 160 are not monotonic and thus cannot be used to estimate an order of convergence.

Convergence results that are approximately second order were obtained for the conservation of chloride, urea, and water. The net steady-state chloride flow is shown in Figure 4.3. The fluctuations for $x >\ \sim$0.7 arise from the changes in diameter and transport properties in the descending limb. These deviations from a horizontal line diminish rapidly as the grid is refined.

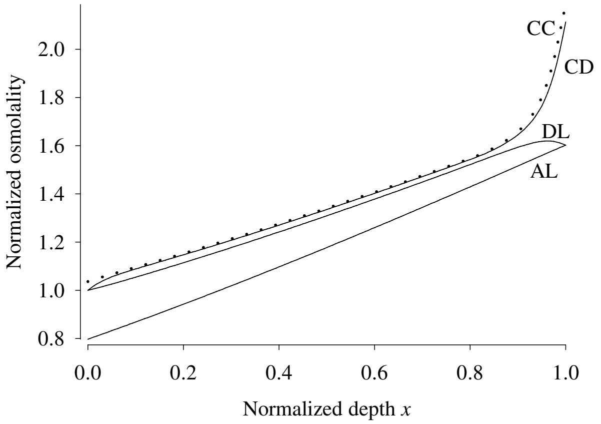


Fig. 4.2. *Steady-state osmolality profiles for the descending limb (DL), ascending limb (AL), collecting duct (CD), and central core (CC), computed on an $N = 160$ grid.*

Table 4.5
Spatial convergence results for the steady-state chloride and water flow in the collecting duct at $x = L$.

N	$(F_{31})_N^{ss}$	p	F_{31_∞}	$(F_{3V})_N^{ss}$	p	F_{3V_∞}
20	7.76885E-2	–	–	3.67955E-2	–	–
40	7.54804E-2	–	–	3.91212E-2	–	–
80	7.51773E-2	2.87	7.51291E-2	3.91878E-2	5.13	3.91897E-2
160	7.51194E-2	2.39	7.51057E-2	3.91740E-2	–	–
320	7.51064E-2	2.16	7.51027E-2	3.91679E-2	1.20	3.91633E-2
640	7.51034E-2	2.08	7.51024E-2	3.91663E-2	1.86	3.91657E-2

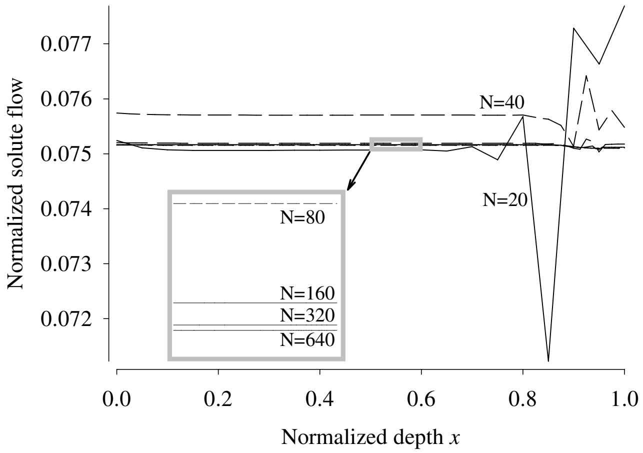


Fig. 4.3. *Net steady-state chloride flow rate through the medulla, computed with successive spatial grid refinements. For most values of x, second-order convergence can be observed.*

TABLE 4.6

Global spatial convergence results computed for the water flow $\vec{F}_V$, chloride flow $\vec{F}_1$, and chloride concentration $\vec{C}_1$, using the L_1-norm, L_2-norm, and L_∞-norm. The orders of convergence are shown in columns p_1, p_2, and p_∞.

			$\vec{F}_V \equiv (F^{ss}_{1V}, F^{ss}_{2V}, F^{ss}_{3V}, F^{ss}_{4V})$			
N	r_1	p_1	r_2	p_2	r_∞	p_∞
40	1.15087E-2	–	2.67290E-2	–	2.91863E-2	–
80	3.42156E-3	1.75	6.59025E-3	2.02	7.87467E-3	1.89
160	8.43613E-4	2.02	1.49520E-3	2.14	1.83683E-3	2.10
320	2.15335E-4	1.97	3.76399E-4	1.99	4.02782E-4	2.01
640	5.34618E-5	2.01	9.15265E-5	2.04	1.90161E-4	2.06
			$\vec{F}_1 \equiv (F^{ss}_{11}, F^{ss}_{21}, F^{ss}_{31}, F^{ss}_{41})$			
N	r_1	p_1	r_2	p_2	r_∞	p_∞
40	1.26031E-2	–	1.82649E-2	–	2.01742E-2	–
80	4.07191E-3	1.63	6.10928E-3	1.58	7.28253E-3	1.47
160	1.21901E-3	1.74	2.28311E-3	1.42	2.25705E-3	1.69
320	3.45248E-4	1.82	6.55652E-4	1.80	6.17469E-4	1.87
640	9.51078E-5	1.86	1.81873E-4	1.85	1.74880E-4	1.82
			$\vec{C}_1 \equiv (C^{ss}_{11}, C^{ss}_{21}, C^{ss}_{31}, C^{ss}_{41})$			
N	r_1	p_1	r_2	p_2	r_∞	p_∞
40	1.63866E-2	–	3.06983E-2	–	5.52759E-2	–
80	3.23587E-3	2.34	5.20568E-3	2.56	1.02573E-2	2.43
160	4.85678E-4	2.28	1.17291E-3	2.15	2.08288E-3	2.30
320	2.35243E-4	1.82	3.20876E-4	1.87	5.46608E-4	1.93
640	5.72026E-5	2.04	7.58917E-5	2.08	1.33840E-4	2.03

Table 4.6 shows global convergence in norm for water flow $\vec{F}_V$, chloride flow $\vec{F}_1$, and chloride concentration $\vec{C}_1$, which are defined as in Table 4.3, as are r_q and p_q. As in the results in Table 4.3, convergence in Table 4.6 was found to be approximately second order. Similar results (not shown) were also obtained for urea flow and urea solute concentration.

5. Stability and efficiency studies. In this section, we compare the stability conditions and efficiency of the SLSI method with an Eulerian scheme, specifically, the ENO method that is first order in time and second order in space.

The procedures by which the ENO method may be used to solve the system (2.1)–(2.6) can be found in [5, 6]. Because of its Eulerian and explicit nature, the time step Δt must be chosen to satisfy the CFL condition; that is, one must ensure that $\Delta t/\Delta x \max_i |F_{iV}/\sqrt{A_i}| < 1$ for $i = 1, 2, 3$, and 4 for the ENO method to be stable. Therefore, as the spatial grid is refined, a smaller Δt may be required to satisfy the CFL condition. Moreover, the transmural solute and water flux terms may place additional restrictions, arising from stiffness, on the size of Δt [5].

In contrast, Δt in the SLSI method is not limited by the CFL condition. Indeed, if the departure points were known exactly, then the advective terms would impose no restriction on Δt. In this problem, however, the flow rates are unknowns and the departure points are estimated numerically. It has been shown that numerical stability is ensured if the product of Δt and the flow shear, $\max_i |\partial(F_{iV}/\sqrt{A_i})/\partial x|$, is bounded by a constant [10]. The stiffness of the problem is controlled by the semi-implicit approach, which averages the stiff transtubular transport terms in time along flow trajectories.

In the studies below, we compared the ENO and SLSI methods by determining the largest Δt for which each method is stable. The efficiency for the ENO and SLSI methods was assessed by two measures. Efficiency-1 was assessed by the standard measure: the computation costs (i.e., computation time) required to attain a specified degree of accuracy. Efficiency-2 was assessed by the computation costs required to attain a steady-state solution while using the largest Δt for which a method is stable. Efficiency-2 is of interest because an approximate, but sufficiently accurate, numerical solution can be used as an initial guess for which a Newton-type solver (for the steady-state equations) will be stable. Because a Newton-type solver is much less computationally expensive than the dynamic methods (e.g., ENO and SLSI), a combination method, using first a dynamic method and then a Newton-type method, may permit an accurate steady-state solution to be found more rapidly than by means of a dynamic method alone.

We expect that the SLSI method will allow a larger Δt than the ENO method, for reasons explained above. However, the computation cost for each SLSI time step is also higher, owing to additional costs incurred in the approximation of flow trajectories, the estimation of upstream function values using spatial interpolation, and the solution of the nonlinear kernel.

5.1. An outer medulla model. In the first set of tests, we formulated the renal model based on parameter values used in section 4.3 (see Table 4.4). With these parameter values the problem is not particularly stiff, and thus Δt in an explicit Eulerian advection method is restricted mostly by the CFL condition. Using numerical experiments, we computed the maximum time steps allowed while maintaining stability for the SLSI and ENO methods for successively refined spatial grids. Then using these time steps, we determined the computation time for the model solution to approximate a steady state; we assumed that a steady state had been attained when the normalized concentrations were within 10^{-6} of their asymptotic values. Both models were run for the same simulated time of 65.97 minutes and by that time both solutions had met the steady-state criterion.

The maximum stable time steps for the ENO and the SLSI methods, the associated normalized steady-state osmolalities of the fluid leaving the collecting duct ($(C_{3O})_N^{ss}$), and the total computation times in seconds are shown in Table 5.1. For the ENO method, Δt must be chosen to be slightly smaller than the maximum value allowed by the CFL condition (i.e., $\Delta x \min_i |\sqrt{A_i}/F_{iV}|$), which is shown in the column labeled "CFL Δt" in Table 5.1. This is because $(F_{iV})_{\max}$ was determined at steady state; flow rates may exceed $(F_{iV})_{\max}$ in transient states. In contrast, a considerably larger Δt can be used in the SLSI method. For $N = 320$, a stable solution is obtained with a CFL number as large as 20.8. Moreover, the choice of Δt in the SLSI method is relatively independent of the grid resolution, whereas for the ENO method a smaller Δt must be used when the spatial grid is refined. Therefore, we expect the advantages of the SLSI method to be amplified with higher spatial resolution.

The results in Table 5.1 indicate that, in the sense of efficiency-2, the SLSI method is more efficient than the ENO method, in that an approximate steady state can be attained much more rapidly by the SLSI method than by the ENO method, and the relative efficiency of the SLSI method increases as the spatial grid is refined—from a factor of ∼2.80 at $N = 80$, to a factor of ∼9.94 at $N = 320$.

The difference between the values of $(C_{3O})_N^{ss}$ computed by the two methods is about 6% or less, a difference that can be reduced by using a smaller Δt in the SLSI method. Using $\Delta t = 0.015$, the SLSI method generates $(C_{3O})_N^{ss} = 1.93168$, 1.92840, and 1.92897 for $N = 80$, 160, and 320, respectively, with computation costs of 14.6, 29.2, and 62.0 s, respectively. The relative discrepancies between these results and

Table 5.1

Results for stability and efficiency study using model for outer medulla.

N	CFL Δt	ENO			SLSI		
		Δt	$(C_{3O})_N^{ss}$	cost (s)	Δt	$(C_{3O})_N^{ss}$	cost (s)
80	1.25E-2	1.0E-2	1.93125	8.16	7.5E-2	2.04574	2.91
160	6.25E-3	5.0E-3	1.92894	34.43	6.5E-2	2.05288	6.74
320	3.13E-3	2.5E-3	1.92874	142.18	6.5E-2	2.04039	14.30

Table 5.2

Tubular diameters and transport properties used in the inner medulla model. DL, descending limb; AL, ascending limb; CD, collecting duct; P_f, P_{Cl^-}, P_{urea}, permeabilities of water, Cl^-, and urea, respectively; $V_{\max,Cl^-}$, maximum Cl^- active transport rate.

Property	DL	AL	CD
Diameter (μm)	15	20	29
P_f (μm/s)	50	0	1,300
P_{Cl^-} (10^{-5} cm/s)	98.8	294	1
P_{urea} (10^{-5} cm/s)	47.6	170	100
$V_{\max,Cl^-}$ (nmole/(cm$^2\cdot$ s))	0	0	0

the ENO approximations shown in Table 5.1 are about 0.022%, 0.028%, and 0.012%, respectively. Thus, for sufficiently large N ($N \geq 160$ for this problem), the SLSI method generates solutions with accuracy comparable to that of the ENO method while also allowing larger time steps *and* lower computation costs. Thus for $N \geq 160$, the SLSI method is more efficient than the ENO method, in the sense of efficiency-1.

Assessed by both the efficiency-1 and efficiency-2 measures, the SLSI method is significantly more efficient than the ENO method, for N sufficiently large, because the costs for the ENO method grow as $\mathcal{O}(N^2)$, whereas the costs of the SLSI method grow as $\mathcal{O}(N)$. The ENO method is $\mathcal{O}(N^2)$ because as N is doubled, Δt must be halved to meet the CFL condition; the SLSI method is not restricted by the CFL condition.

5.2. An inner medulla model for the chinchilla. For this test, we based the model parameters on measurements in the deep inner medulla (i.e., the papillary portion) of the rat and chinchilla. Diameter and transport properties, based on anatomical and physiological investigations [7], are given in Table 5.2. The medullary length L was set to be 4.8 mm and the prebend segment in the late descending limb was assumed to begin at $x_p = 0.9L$. The descending and ascending limb transport properties were measured in chinchilla, and diameters and collecting duct transport properties were measured in rat. Transport parameters for chinchilla were chosen because they are large compared to measurements in other small mammals, thus rendering the problem stiff. Boundary conditions for this problem were specified for $C_{11}(0,t)$, $C_{12}(0,t)$, $C_{31}(0,t)$, and $C_{32}(0,t)$ to be 600.0, 150.0, 230.98, and 850.0 mM, respectively, and for $F_{1V}(0,t)$ and $F_{3V}(0,t)$ to be 1.0 and 0.15 nl/min, respectively. Model calculations were run for 65.97 minutes in simulated time, and by then the solutions had met the steady-state criterion described in section 5.1.

The maximum stable time steps for the ENO and SLSI methods, the associated normalized steady-state osmolalities of the fluid leaving the collecting duct, and the total computation times in seconds are shown in Table 5.3. The parameters for this problem were chosen so that the transmural water and solute fluxes were large near $x = 0$, rendering the problem stiff. As a result, the maximum time steps allowed

Table 5.3
Results for stability and efficiency study for a model of the inner medulla.

N	CFL Δt	ENO			SLSI		
		Δt	$(C_{3O})_N^{ss}$	cost (s)	Δt	$(C_{3O})_N^{ss}$	cost (s)
80	1.25E-2	1.0E-3	0.89539	89.90	3.0E-3	0.89376	63.19
160	6.25E-3	5.0E-4	0.95640	355.19	3.0E-3	0.96482	128.66
320	3.13E-3	2.5E-4	0.97068	1448.65	3.0E-3	0.96897	269.13

for both the ENO and SLSI methods were substantially smaller than the CFL Δt. The maximum time steps permissible for the ENO method become smaller as N is increased, even though stiffness is normally independent of the grid resolution, because the transmural water and solute fluxes increase as x approaches 0 (a boundary layer effect): a smaller Δx results in a larger computed flux at the first few grid points, and thus a smaller Δt is required. Even with the semi-implicit approach used in the SLSI method, the large water fluxes near $x = 0$ affect the fluid flow rate and thereby increase the flow shear and place a restriction on Δt. Unlike the ENO method, however, Δt for the SLSI method need not be further reduced as the spatial grid is refined. The computation costs of the ENO and SLSI methods grow as $\mathcal{O}(N^2)$ and $\mathcal{O}(N)$, respectively. For this problem, the SLSI method reduces computation times by a factor of 1.38 for $N = 80$ and by a factor of 5.38 for $N = 320$. As in the previous test, for sufficiently large N (here, $N \geq 80$), the SLSI method is more efficient than the ENO method in both the efficiency-1 sense and the efficiency-2 sense.

6. Discussion. We have presented a stable and efficient numerical method, based on an SLSI scheme, for solving the system of differential equations arising in dynamic models of the urine concentrating mechanism. The SLSI method, like the ENO method used previously [5, 6], avoids numerical instability arising from transient flow reversal, and it does so by integrating along flow trajectories. Unlike the ENO scheme, however, the SLSI method is not limited by the CFL condition. With an Eulerian scheme, a smaller time step may be required, to satisfy the CFL condition, when the spatial grid is refined. Thus, even for a one-dimensional problem, the computation cost may grow as $\mathcal{O}(N^2)$, where N is the number of spatial grid subintervals. Since the SLSI method integrates along flow trajectories and thus allows CFL numbers greater than 1, the computation cost grows only as $\mathcal{O}(N)$. Moreover, this method treats the stiff terms (the transmural water and solute fluxes) implicitly so that they do not impose a restriction on the time step. In sections 4 and 5, we demonstrated that results from the SLSI method show second-order convergence in space, and that, compared with the ENO method, the SLSI method computes solutions of comparable accuracy in substantially shorter times.

The model presented in this paper is relatively simple in that it consists of one loop of Henle, one collecting duct, and a central core. A rat kidney, however, contains ~37,500 loops of Henle, turning at various levels of the medulla [3]. To accurately represent the interactions among these loops of differing lengths, a distributed-loop model may be adopted [6]. The distributed-loop model uses two space variables: one denotes the level of the medulla, while the other represents the level of the medulla at which the associated loop of Henle turns, resulting in a system of N PDEs for each solute. When this system is solved by the ENO method, $\mathcal{O}(N^3)$ computation steps are required. Thus, even though high spatial resolution is frequently required to accurately represent the details and complexities of the mammalian kidney (e.g., rapid transitions in diameter and transtubular transport properties), one may be compelled to compromise spatial resolution to offset the long computation times.

It should be practical to develop an efficient and stable numerical method, based on the SLSI scheme, for the distributed-loop model. The solution time of an SLSI-based method for this two-dimensional model should be $\mathcal{O}(N^2)$, since the choice of time step is independent of the spatial resolution. Even if the largest stable time step is used, the approximate steady-state solution so obtained may be useful as an initial condition for a more accurate and less stable method, such as a Newton-type method or a simple explicit Eulerian advection scheme with a small time step. The resulting savings in computation costs can be invested in higher spatial resolution, so that a more complete understanding of the mammalian urine concentrating mechanism may be attained.

Appendix. Explicit solution for the simple steady-state model. This appendix provides a derivation for (4.1), following that of a related problem in [4]. From conservation of solute, one has

$$F'_{ik}(x) = J_{ik}(x) \tag{A.1}$$

for $i = 1$, 2, 3, or 4. The prime symbol denotes differentiation with respect to x. A substitution of the definition of solute flow $F_{ik}(x) = F_{iV}(x)C_{1k}(x)$ into (A.1) yields

$$F_{iV}(x)C'_{ik}(x) = -J_{iV}(x)C_{ik}(x) + J_{ik}(x). \tag{A.2}$$

Since for $x \in [0, x_p]$ the solute concentration of the descending limb equals $C(x)$, one obtains the following relation for the proximal, water-permeable portion of the descending limb:

$$H^-_{x_p}(x)F_{1V}(x)C'(x) = H^-_{x_p}(x)(-J_{1V}(x)C(x) + J_{1k}(x)), \tag{A.3}$$

where $H^+_{x_p}(x)$ and $H^-_{x_p}(x)$ are the Heaviside functions defined immediately after (4.1). By adding together (A.3) and equations of the form (A.2) for $i = 3$ and 4, one obtains

$$\begin{aligned}(H^-_{x_p}(x)&F_{1V}(x) + F_{3V}(x) + F_{4V}(x))C'(x)\\ &= -(H^-_{x_p}(x)J_{1V}(x) + J_{3V}(x) + J_{4V}(x))C(x) + (H^-_{x_p}(x)J_{1k}(x) + J_{3k}(x) + J_{4k}(x)).\end{aligned} \tag{A.4}$$

Since the ascending limb and the prebend segment are assumed to be water impermeable, $J_{2V} \equiv 0$ and $H^+_{x_p}J_{1V} \equiv 0$; thus, by water conservation, $H^-_{x_p}(x)J_{1V}(x) + J_{3V}(x) + J_{4V}(x) = 0$. By solute conservation, $H^-_{x_p}(x)J_{1k}(x) + J_{3k}(x) + J_{4k}(x) = -H^+_{x_p}(x)J_{1k}(x) - J_{2k}(x)$, where $H^+_{x_p}(x)J_{1k}(x)$ represents solute flux from the prebend segment. Recall also that $F_{iV}(x) = F_{ik}(x)/C(x)$ for $i = 3$ and 4 and for the water-permeable portion of the descending limb ($x \in [0, x_p]$). By substituting these relations into (A.4) and rearranging, one obtains

$$\frac{C'(x)}{C(x)} = \frac{-H^+_{x_p}(x)J_{1k}(x) - J_{2k}(x)}{H^-_{x_p}(x)F_{1k}(x) + F_{3k}(x) + F_{4k}(x)}. \tag{A.5}$$

The rate of solute advection along the collecting duct at level x is

$$F_{3k}(x) = F_{3k}(0) + \int_0^x J_{3k}(s)\,ds = F_{3k}(L) - \int_x^L J_{3k}(s)\,ds. \tag{A.6}$$

Since $F_{4k}(L) = 0$, the solute advection in the central core is given by

$$F_{4k}(x) = \int_x^L (J_{1k}(s) + J_{2k}(s) + J_{3k}(s))\, ds. \tag{A.7}$$

By adding (A.6) and (A.7), one obtains

$$F_{3k}(x) + F_{4k}(x) = F_{3k}(L) + \int_x^L (J_{1k}(s) + J_{2k}(s))\, ds. \tag{A.8}$$

By substituting (A.8) into (A.5), one obtains an equation that may be integrated to give (4.1).

REFERENCES

[1] K. E. Atkinson, *An Introduction to Numerical Analysis*, 2nd ed., Wiley, New York, 1989, p. 292.

[2] A. Harten and S. Osher, *Uniformly high-order accurate nonoscillatory schemes.* I, SIAM J. Numer. Anal., 24 (1987), pp. 279–309.

[3] M. A. Knepper, R. A. Danielson, G. M. Saidel, and R. S. Post, *Quantitative analysis of renal medullary anatomy in rats and rabbits*, Kidney Int., 12 (1977), pp. 313–323.

[4] H. E. Layton and J. M. Davies, *Distributed solute and water reabsorption in a central core model of the renal medulla*, Math. Biosci., 116 (1993), pp. 169–196.

[5] H. E. Layton and E. B. Pitman, *A dynamic numerical method for models of renal tubules*, Bull. Math. Biol., 56 (1994), pp. 547–565.

[6] H. E. Layton, E. B. Pitman, and M. A. Knepper, *A dynamic numerical method for models of the urine concentrating mechanism*, SIAM J. Appl. Math., 55 (1995), pp. 1390–1418.

[7] H. E. Layton, M. A. Knepper, and C.-L. Chou, *Permeability criteria for effective function of passive countercurrent multiplier*, Amer. J. Physiol., 270 (1996), pp. F9–F20.

[8] P. Lory, *Numerical solution of a kidney model by multiple shooting*, Math. Biosci., 50 (1980), pp. 117–128.

[9] L. C. Moore and D. J. Marsh, *How descending limb of Henle's loop permeability affects hypertonic urine formation*, Amer. J. Physiol., 239 (1980), pp. F57–F71.

[10] J. Pudykiewicz, R. Benoit, and A. Staniforth, *Preliminary results from a partial LRTAP model used on an existing meteorological forecast model*, Atmos.-Ocean, 32 (1985), pp. 267–303.

[11] A. Robert, *A semi-Lagrangian, semi-implicit numerical integration scheme for the primitive meteorological equations*, Atmos.-Ocean, 19 (1981), pp. 35–46.

[12] J. M. Sands and H. E. Layton, *Urine concentrating mechanism and its regulation*, in The Kidney: Physiology and Pathophysiology, 3rd ed., D. W. Seldin and G. Giebisch, eds., Lippincott Williams and Wilkins, Philadelphia, 2000, pp. 1175–1216.

[13] R. Staniforth and J. Côté, *Semi-Lagrangian integration schemes for atmospheric models: A review*, Mon. Weather Rev., 119 (1991), pp. 2206–2223.

[14] A. Staniforth and J. Pudykiewicz, *Reply to comments on addenda to "Some properties and comparative performance of the semi-Lagrangian method of Robert in the solution of the advection-diffusion equation,"* Atmos.-Ocean, 23 (1985), pp. 195–200.

[15] J. L. Stephenson, *Central core model of the renal counterflow system*, Kidney Int., 2 (1972), pp. 85–94.

[16] J. L. Stephenson, *Urinary concentration and dilution: Models*, in Handbook of Physiology: Renal Physiology, E. E. Windhager, ed., published for the American Physiological Society by Oxford University Press, New York, 1992, pp. 1349–1408.

[17] C. Temperton and A. Staniforth, *An efficient two-time-level semi-Lagrangian semi-implicit scheme*, Quart. J. Roy. Meteor. Soc., 113 (1987), pp. 1025–1039.

[18] R. P. Tewarson, H. Wang, J. L. Stephenson, and J. F. Jen, *Efficient solution of differential equations for kidney concentrating mechanism analyses*, Appl. Math. Lett., 4 (1991), pp. 69–72.

[19] H. Wang, R. P. Tewarson, J. F. Jen, and J. L. Stephenson, *A comparison of multinephron and shunt models of the renal concentrating mechanism*, Appl. Math. Lett., 6 (1993), pp. 61–65.

[20] X. Wang, A. S. Wexler, and D. J. Marsh, *The effect of solution non-ideality on membrane transport in three-dimensional models of the renal concentrating mechanism*, Bull. Math. Biol., 56 (1994), pp. 515–546.

[21] X. Wang, S. R. Thomas, and A. S. Wexler, *Outer medullary anatomy and the urine concentrating mechanism*, Amer. J. Physiol., 274 (1998), pp. F413–F424.

[22] R. C. Weast, *CRC Handbook of Chemistry and Physics*, 55th ed., CRC Press, Cleveland, OH, 1974, p. D-224.

[23] A. S. Wexler, R. E. Kalaba, and D. J. Marsh, *Three-dimensional anatomy and renal concentrating mechanism* I: *Modeling results*, Amer. J. Physiol., 260 (1991), pp. F368–F383.

[24] A. S. Wexler, R. E. Kalaba, and D. J. Marsh, *Three-dimensional anatomy and renal concentrating mechanism* II: *Sensitivity results*, Amer. J. Physiol., 260 (1991), pp. F384–F394.

[25] D. L. Williamson and R. Laprise, *Numerical approximations for global atmospheric general circulation models*, in Numerical Modelling of the Global Atmosphere in the Climate System, P. Mote and A. O'Neill, eds., Kluwer Academic Publishers, Norwell, MA, 2000, pp. 127–219.

SIAM J. SCI. COMPUT.
Vol. 23, No. 5, pp. 1549–1571

FAST EVALUATION OF RADIAL BASIS FUNCTIONS: METHODS FOR GENERALIZED MULTIQUADRICS IN $\mathbb{R}^n$*

J. B. CHERRIE†, R. K. BEATSON†, AND G. N. NEWSAM‡

Abstract. A generalized multiquadric radial basis function is a function of the form $s(x) = \sum_{i=1}^N d_i\phi(|x - t_i|)$, where $\phi(r) = (r^2 + \tau^2)^{k/2}$, $x \in \mathbb{R}^n$, and $k \in \mathbb{Z}$ is odd. The direct evaluation of an N center generalized multiquadric radial basis function at m points requires $\mathcal{O}(mN)$ flops, which is prohibitive when m and N are large. Similar considerations apparently rule out fitting an interpolating N center generalized multiquadric to N data points by either direct or iterative solution of the associated system of linear equations in realistic problems.

In this paper we will develop far field expansions, recurrence relations for efficient formation of the expansions, error estimates, and translation formulas for generalized multiquadric radial basis functions in n-variables. These pieces are combined in a hierarchical fast evaluator requiring only $\mathcal{O}((m+N)\log N|\log\epsilon|^{n+1})$ flops for evaluation of an N center generalized multiquadric at m points. This flop count is significantly less than that of the direct method. Moreover, used to compute matrix-vector products, the fast evaluator provides a basis for fast iterative fitting strategies.

Key words. radial basis functions, generalized multiquadric, fast evaluation

AMS subject classifications. 65D07, 41A15, 41A58

PII. S1064827500367609

1. Introduction.

Multiquadrics are a popular choice of radial basis function for interpolating scattered data in one or more dimensions. Many applications are described in the literature, including geodesy, image processing, and natural resource modeling (see, for example, Hardy [10]). The beautiful properties of multiquadric and other radial basis functions, such as the poisedness of suitable interpolation problems, are detailed in Cheney and Light [7, Chap. 12–16, Chap. 36]. Unfortunately, the adoption of multiquadrics for real problems with large data sets has been hindered by a perceived large computational cost. Indeed, the direct evaluation of an N center multiquadric radial basis function at m points requires $\mathcal{O}(mN)$ flops, which is prohibitive when m and N are large. Similar considerations apparently rule out fitting an interpolating N center multiquadric to N data points by either direct or iterative solution of the associated system of linear equations in realistic problems.

However, the use of hierarchical methods, fast multipole methods, and other multiresolution schemes allows fast evaluation and fitting of radial basis functions. This paper develops far field expansions for generalized multiquadric radial basis functions in n-variables of the form required by these new methods. Schemes of a hierarchical type can then be built upon these expansions that require only $\mathcal{O}((m+N)\log N|\log\epsilon|^{n+1})$ flops for evaluation of an N center generalized multiquadric to accuracy ϵ at m points. This compares very favorably with the cost of the direct method. Moreover, used to compute matrix-vector products, the fast evaluator can be combined with suitable iterative methods and preconditioning strategies to yield

*Received by the editors February 11, 2000; accepted for publication (in revised form) May 29, 2001; published electronically January 16, 2002.

http://www.siam.org/journals/sisc/23-5/36760.html

†Department of Mathematics and Statistics, University of Canterbury, Private Bag 4800, Christchurch, New Zealand (jcherrie@mathworks.co.uk, R.Beatson@math.canterbury.ac.nz). The research of the first and second authors was partially supported by PGSF subcontract DRF601.

‡Surveillance Systems Division, Defence Science and Technology Organisation, P.O. Box 1500, Salisbury, SA 5108, Australia (garry.newsam@dsto.defence.gov.au).

fast iterative algorithms for interpolatory or smoothing fits (see, e.g., [2]).

The first fast multipole method was that of Greengard and Rokhlin [9]. Since then the method has been modified and extended to apply in many different contexts [3]. For reasons of space we are forced to omit discussion of many important aspects of hierarchical and fast multipole methods from this paper. In particular, we have omitted almost all discussion of the crucial algorithmic details which enable a fast evaluation scheme for use in $\mathbb{R}^n$ to be built upon suitable far field expansions, such as the expansion for generalized multiquadrics developed in this paper.

A much fuller account of hierarchical and fast multipole methods is given in the survey paper [3]. Readers new to these methods are referred to that paper, and in particular to the tutorial section concerning hierarchical and fast multipole schemes in one dimension. Indeed, the model problem of that section is fast evaluation of an ordinary multiquadric in $\mathbb{R}^1$. However, the treatment there concentrates exclusively on algorithmic aspects and suppresses the mathematical analysis of expansions and error bounds. Previous papers concerning fast multipole and related methods for fast evaluation of radial basis functions include [5, 4, 6].

The generic fast multipole method requires results of the following nature for the basic function Φ being used:

- the existence of a rapidly converging far field expansion, centered at 0, for the shifted basic function $\Phi(x-t)$. The existence of such an expansion implies that, for all x sufficiently far from 0, the spline $s(x) = \sum_{i=1}^{N} d_i \Phi(x-t_i)$ may be approximated to the desired accuracy by a short series. When N is large it will be much faster to use the series rather than to evaluate $s(x)$ directly.
- error bounds that determine how many terms are required in each expansion to achieve a specified accuracy.
- efficient recurrence relations for computing the coefficients of the expansions.
- uniqueness results that justify indirect translation of expansions.
- formulae for efficiently converting a far field expansion to a rapidly convergent local expansion.

This paper provides appropriate results of these types for generalized multiquadric radial basis functions in $\mathbb{R}^n$. That is, for functions of the form

$$s(x) = \sum_{i=1}^{N} d_i \Phi(x - t_i; k, \tau), \tag{1.1}$$

where

$$\Phi(x) = \Phi(x; k, \tau) = \left(x^2 + \tau^2\right)^{k/2}, \tag{1.2}$$

k is an odd integer, $\tau \geq 0$, and $x \in \mathbb{R}^n$. Note that we will usually use the notation $\Phi(x)$, which hides the dependence of Φ on k and τ. The derived series and the analysis also apply when τ varies, that is, the multiquadric parameter τ changes with the center t_i.

The paper is laid out as follows. Sections 2 and 3 derive far field expansions of the form

$$\Phi(x - t; k, \tau) = \sum_{\ell=0}^{\infty} P_\ell^{(k)}(|t|^2 + \tau^2, -2\langle t, x\rangle, |x|^2)/|x|^{2\ell-k}, \tag{1.3}$$

where the $P_\ell^{(k)}$ are the polynomials

$$P_\ell(a,b,c) = P_\ell^{(k)}(a,b,c) = \sum_{j=\lfloor \frac{\ell+1}{2} \rfloor}^{\ell} \binom{k/2}{j}\binom{j}{\ell-j} b^{2j-\ell}(ac)^{\ell-j}, \qquad \ell \geq 0, \tag{1.4}$$

and $P_\ell^{(k)}$ is the zero function for ℓ negative. Section 3 also gives error bounds on approximations formed by truncating the series. Section 4 proves the uniqueness of the expansions. Section 5 discusses recurrence relations for the efficient direct calculation of the far field coefficients. It shows that the terms of the first $p+k+1$ homogeneous orders in the series for an m center cluster can be calculated in $\mathcal{O}(mn(p+k)^n)$ flops. Section 6 sets up some machinery which is used in section 7 to establish methods for indirectly translating far field expansions. Section 8 shows how to efficiently convert a far field expansion into a local polynomial approximation. The paper concludes with some numerical results showing that multiquadric radial basis functions can indeed be evaluated using this approach at a cost that grows as $\mathcal{O}(N \log N)$ in the number N of centers.

We will use lowercase ϕ for the basic function as a function of one variable, and uppercase Φ for the function of n variables, i.e., $\Phi = \phi(|\cdot|)$. It is common for the constant in the multiquadric basic function to be represented by c. However, we will use τ for this purpose, i.e., the ordinary multiquadric basic function will be $\phi(r) = \sqrt{r^2+\tau^2}$.

2. A generating function. In this section we develop some important properties of the functions

$$f_k(z) = (\sqrt{az^2+bz+c})^k, \qquad k \in \mathbb{Z} \text{ is odd.} \tag{2.1}$$

These functions will turn out to be the generating functions for the polynomials $P_\ell^{(k)}$ that occur in the far and near field expansions of the generalized multiquadric function.

To fully explore the expansions of f_k we will need to use Gauss's hypergeometric function.

LEMMA 2.1. *The* hypergeometric function *defined by*

$$F(a,b;c;z) = F(b,a;c;z) := \frac{\Gamma(c)}{\Gamma(a)\Gamma(b)} \sum_{n=0}^{\infty} \frac{\Gamma(a+n)\Gamma(b+n)}{\Gamma(c+n)} \frac{z^n}{n!},$$

for c not a negative integer and $|z|<1$, satisfies

$$F(a,b;c;z) = (1-z)^{c-a-b} F(c-a,c-b;c;z), \tag{2.2a}$$

$$\frac{d}{dz}F(a,b;c;z) = \frac{ab}{c}F(a+1,b+1;c+1;z). \tag{2.2b}$$

Furthermore, if a or b is equal to $-m$, m a nonnegative integer, then $F(a,b;c;z)$ reduces to a polynomial of degree m in z.

Proof. See [1, Chap. 15] for the proof. □

LEMMA 2.2. *Let $m,p \in \mathbb{N}_0$, and $|h|<1$. Then*

$$\sum_{n=p}^{\infty} \binom{n+m}{m} h^n = \frac{h^p}{(1-h)^{m+1}} \frac{(p+m)!}{p!\ m!} F(-m,p;p+1;h).$$

Proof.

$$\begin{aligned}\sum_{n=p}^{\infty}\binom{n+m}{m}h^n &= \frac{(p+m)!}{p!\,m!}\frac{p!}{(p+m)!}h^p\sum_{n=0}^{\infty}\frac{(n+p+m)!n!}{(n+p)!}\frac{h^n}{n!}\\ &= \frac{(p+m)!}{p!\,m!}h^pF(m+p+1,1;p+1;h)\\ &= \frac{(p+m)!}{p!\,m!}h^p(1-h)^{-(m+1)}F(-m,p;p+1;h),\end{aligned}$$

where the last equality follows from (2.2a). □

We now present the major result of this section, which gives a series expansion for f_k and a bound for the error in approximating f_k by a truncation of this series.

LEMMA 2.3. *Let $k \in \mathbb{Z}$ be odd and let $a, b, c \in \mathbb{R}$ with $a, c > 0$ and $b^2 \le 4ac$. Then for all $z \in \mathbb{C}$ such that $|z| < \sqrt{c/a}$,*

$$f_k(z) = (\sqrt{az^2+bz+c})^k = c^{k/2}\sum_{\ell=0}^{\infty}\left(\frac{z}{c}\right)^{\ell}P_\ell^{(k)}(a,b,c), \tag{2.3}$$

where the $P_\ell^{(k)}$ are the polynomials defined in (1.4). *Moreover, for all z such that $|z| < \sqrt{c/a}$ and $\nu \in \mathbb{N}$,*

$$\left|(az^2+bz+c)^{k/2} - c^{k/2}\sum_{\ell=0}^{\nu}\left(\frac{z}{c}\right)^{\ell}P_\ell^{(k)}(a,b,c)\right|$$
$$\le \begin{cases} 2^kc^{k/2}\left(\dfrac{|z|}{\sqrt{c/a}}\right)^{\nu+1}\dfrac{\sqrt{c/a}}{\sqrt{c/a}-|z|} & \textit{if } k>0,\\[2ex] \dbinom{\nu-k}{\nu+1}c^{k/2}\left(\dfrac{|z|}{\sqrt{c/a}}\right)^{\nu+1}\left(\dfrac{\sqrt{c/a}}{\sqrt{c/a}-|z|}\right)^{-k} & \\ \qquad\times F\left(k+1,\nu+1;\nu+2;\dfrac{|z|}{\sqrt{c/a}}\right) & \textit{if } k<0.\end{cases}$$

Proof. Let $\sqrt{\cdot}$ denote the principal branch of the complex square root. Then f_k is analytic whenever $q(z) = az^2+bz+c$ is away from the branch cut, that is, whenever $q(z)$ is not a nonpositive real. Completing the square,

$$q(z) = a\left\{\left(z+\frac{b}{2a}\right)^2 + \frac{4ac-b^2}{4a^2}\right\},$$

and since $b^2 \le 4ac$, it is easily seen that f_k is analytic away from

$$\left\{z = -\frac{b}{2a} + \mathbf{i}y : y \in \mathbb{R} \text{ and } |y| \ge \sqrt{\frac{4ac-b^2}{4a^2}}\right\}.$$

Hence f_k is analytic on the disc

$$D = D_\epsilon = \left\{z \in \mathbb{C} : |z| \le \rho = (1-\epsilon)\sqrt{c/a}\right\}, \qquad 0 < \epsilon < 1.$$

For all sufficiently small $|z|$, two applications of the binomial theorem and some reordering give

$$\begin{aligned}
f_k(z) &= c^{k/2}\left(1+\frac{bz+az^2}{c}\right)^{k/2}\\
&= c^{k/2}\sum_{j=0}^{\infty}\binom{k/2}{j}\left(\frac{bz+az^2}{c}\right)^{j}\\
&= c^{k/2}\sum_{j=0}^{\infty}\binom{k/2}{j}\sum_{q=0}^{j}\binom{j}{q}\frac{(bz)^{j-q}(az^2)^q}{c^j}\\
&= c^{k/2}\sum_{\ell=0}^{\infty}\sum_{j=\lfloor\frac{\ell+1}{2}\rfloor}^{\ell}\binom{k/2}{j}\binom{j}{\ell-j}\frac{(bz)^{2j-\ell}(az^2)^{\ell-j}}{c^j}\\
&= c^{k/2}\sum_{\ell=0}^{\infty}\left(\frac{z}{c}\right)^{\ell}\sum_{j=\lfloor\frac{\ell+1}{2}\rfloor}^{\ell}\binom{k/2}{j}\binom{j}{\ell-j}b^{2j-\ell}(ac)^{\ell-j}\\
&= c^{k/2}\sum_{\ell=0}^{\infty}\left(\frac{z}{c}\right)^{\ell}P_\ell^{(k)}(a,b,c).
\end{aligned}$$

Since the reordering of the double sum is valid for $|z|$ sufficiently small, for such z this is the Maclaurin series for f_k. This relation extends to all of D by the uniqueness of the Maclaurin series of f_k, proving the first part of the lemma.

We will prove the second part separately for $k > 0$ and $k < 0$. For $k > 0$ we will apply the well-known bound for the error in Taylor polynomial approximation given in Lemma 2.4 below. Fix z with $|z| < \sqrt{c/a}$ and choose ϵ with $0 < \epsilon < 1$ so small that $z \in D_\epsilon$. We apply the bound with $C = \partial D_\epsilon$. First, note that

$$q(z) = a(z-\xi_+)(z-\xi_-), \qquad \xi_\pm = \frac{-b\pm \mathbf{i}\sqrt{4ac-b^2}}{2a},$$

and that both roots of q are outside D_ϵ. Since $f_k(z) = q(z)^{k/2}$,

$$\max_{w\in C}|f_k(w)| = \left(\max_{w\in C}|q(w)|\right)^{k/2}.$$

For $w \in \partial D_\epsilon$,

$$|w-\xi_\pm| \le |w| + |\xi_\pm| = \rho + \sqrt{c/a} < 2\sqrt{c/a},$$

and thus

$$\max_{w\in\partial D_\epsilon}|q(w)| = |a|\max_{w\in\partial D_\epsilon}\{|w-\xi_+||w-\xi_-|\} \le |a|\big(2\sqrt{c/a}\big)^2 = 4c.$$

Now, applying Lemma 2.4,

$$\begin{aligned}
&\left|(az^2+bz+c)^{k/2} - c^{k/2}\sum_{\ell=0}^{\nu}\left(\frac{z}{c}\right)^{\ell}P_\ell^{(k)}(a,b,c)\right|\\
&\qquad\le \max_{w\in\partial D_\epsilon}|f_k(w)|\left(\frac{|z|}{\rho}\right)^{\nu+1}\frac{1}{1-|z|/\rho}\\
&\qquad\le (4c)^{k/2}\left(\frac{|z|}{(1-\epsilon)\sqrt{c/a}}\right)^{\nu+1}\frac{(1-\epsilon)\sqrt{c/a}}{(1-\epsilon)\sqrt{c/a}-|z|}.
\end{aligned}$$

Taking the limit as ϵ goes to zero from above gives the result for $k > 0$.

For the case $k < 0$, write the polynomial q in the form

$$\begin{aligned} q(z) = az^2 + bz + c &= c\left\{1 + \frac{b}{\sqrt{ac}}\left(\frac{z}{\sqrt{c/a}}\right) + \left(\frac{z}{\sqrt{c/a}}\right)^2\right\} \\ &= c\big(1 - 2x\xi + \xi^2\big), \end{aligned}$$

where

$$x = -\frac{1}{2}\frac{b}{\sqrt{ac}} \qquad \text{and} \qquad \xi = \frac{z}{\sqrt{c/a}}.$$

Now recall (see [11, eqn. (4.7.23)]) that $\big(1 - 2x\xi + \xi^2\big)^{-\lambda}$ is the generating function for the *Gegenbauer (or ultraspherical) polynomials* $C_\ell^{(\lambda)}(x)$, i.e.,

$$\sum_{\ell=0}^{\infty} C_\ell^{(\lambda)}(x)\xi^\ell = (1 - 2x\xi + \xi^2)^{-\lambda}.$$

Letting $\lambda = -k/2$, we see that

$$f_k(z) = c^{k/2}\sum_{\ell=0}^{\infty} C_\ell^{(\lambda)}(x)\xi^\ell,$$

and thus, equating coefficients,

$$\left(\frac{z}{c}\right)^\ell P_\ell^{(k)}(a, b, c) = C_\ell^{(\lambda)}(x)\xi^\ell, \qquad \ell \in \mathbb{N}_0. \tag{2.4}$$

For $-1 \le x \le 1$,

$$\big|C_n^{(\lambda)}(x)\big| \le \binom{n + 2\lambda - 1}{n}, \qquad \lambda > 0$$

(see [1, eqn. (22.14.2)]). By the statement of the lemma, $b^2 \le 4ac$ and $|z| < \sqrt{c/a}$. This means that $-1 \le x \le 1$ and $|\xi| < 1$ and thus

$$\begin{aligned} \left|f_k(z) - c^{k/2}\sum_{\ell=0}^{\nu}\left(\frac{z}{c}\right)^\ell P_\ell^{(k)}(a, b, c)\right| &= \left|f_k(z) - c^{k/2}\sum_{\ell=0}^{\nu} C_\ell^{(-k/2)}(x)\xi^\ell\right| \\ &\le c^{k/2}\sum_{\ell=\nu+1}^{\infty}\binom{\ell - k - 1}{\ell}|\xi|^\ell. \end{aligned} \tag{2.5}$$

By Lemma 2.2,

$$\sum_{\ell=\nu+1}^{\infty}\binom{\ell - k - 1}{\ell}|\xi|^\ell = \binom{\nu - k}{\nu + 1}\frac{|\xi|^{\nu+1}}{(1 - |\xi|)^{-k}} F\left(k + 1, \nu + 1; \nu + 2; |\xi|\right).$$

Using this in (2.5) we have

$$\left| f_k(z) - c^{k/2} \sum_{\ell=0}^{\nu} \left(\frac{z}{c}\right)^{\ell} P_\ell^{(k)}(a,b,c) \right| \le c^{k/2} \binom{\nu-k}{\nu+1} \left(\frac{|z|}{\sqrt{c/a}}\right)^{\nu+1} \left(\frac{\sqrt{c/a}}{\sqrt{c/a}-|z|}\right)^{-k} \times F\left(k+1, \nu+1; \nu+2; \frac{|z|}{\sqrt{c/a}}\right). \qquad \square$$

In the proof of Lemma 2.3 above we have made use of the following well-known bound for the error in a truncated Taylor series expansion [8, pp. 127–128].

LEMMA 2.4. *Let $C = \{w \in \mathbb{C} : |w| = \rho\}$. If f is analytic inside and on C, then for $|z| < \rho$,*

$$\left|f(z) - \big(T_\nu f\big)(z)\right| \le \max_{w\in C} |f(w)| \left(\frac{|z|}{\rho}\right)^{\nu+1} \frac{1}{1-|z|/\rho},$$

where $T_\nu f$ is the Maclaurin polynomial of f of degree ν.

In the case $k = -1$, the polynomial $F(k+1, \nu+1; \nu+2; z/\sqrt{c/a})$ that appears in the error bound of Lemma 2.3 is constant and has value 1. For all other negative values of k consider the function $F(k+1, p+1; p+2; \cdot)$, where $p \in \mathbb{N}_0$. Rephrasing Lemma 2.2 as

$$F(k+1, p; p+1; z) = \frac{p!\,(-k-1)!}{(p-k-1)!} \frac{(1-z)^{-k}}{z^p} \sum_{n=p}^{\infty} \binom{n-k-1}{-k-1} z^n,$$

we can easily see that $F(k+1, p+1; p+2; \cdot)$ is nonnegative on $[0,1)$. Using (2.2b) to differentiate F, we see that for $z \in [0,1)$

$$\frac{d}{dz} F(k+1, p; p+1; z) = \frac{(k+1)p}{p+1} F(k+2, p+1; p+2; z) \le 0,$$

since $k < -1$. Since $F(\cdot,\cdot;\cdot;0) = 1$, it follows that

$$F(k+1, \nu+1; \nu+2; |z|/\sqrt{c/a}) \le 1, \qquad k \in \mathbb{Z}_-, \quad |z| < \sqrt{c/a}. \tag{2.6}$$

As was observed in the proof of Lemma 2.3 and particularly in (2.4), for $k < 0$ the polynomials $P_\ell^{(k)}$ are closely related to the Gegenbauer polynomials $C_\ell^{(\lambda)}$ with $\lambda = -k/2$. However, many properties of the Gegenbauer polynomials are derived using their orthogonality with respect to the weight function $w(x) = (1-x^2)^{\lambda-1/2}$. This function is not integrable over the interval $[-1,1]$ when $\lambda \le -1/2$, and thus we are unable to exploit properties of the Gegenbauer polynomials derived from orthogonality when $k \ge 1$. The following lemma can be identified as a well-known recurrence for the Gegenbauer polynomials with parameter $\lambda = -k/2 > -1/2$. Our proof here is based on the characterization (2.3) and hence holds for all odd integers k.

LEMMA 2.5. *Let $k \in \mathbb{Z}$ be odd. Then the polynomials $P_\ell^{(k)}$ defined in* (1.4) *satisfy the following recurrence relation for all $a, b, c \in \mathbb{R}$ and $\ell \in \mathbb{N}$:*

$$(\ell+1) P_{\ell+1}^{(k)}(a,b,c) = \left(\frac{k}{2} - \ell\right) b P_\ell^{(k)}(a,b,c) + \big(k - (\ell-1)\big) ac P_{\ell-1}^{(k)}(a,b,c). \tag{2.7}$$

Proof. We will first prove the identity under the additional assumptions $a, c > 0$ and $b^2 \leq 4ac$. Making these assumptions and differentiating the right-hand side of (2.3) term by term gives

$$f_k'(z) = c^{(k-2)/2} \sum_{\ell=0}^{\infty} \left(\frac{z}{c}\right)^{\ell} (\ell+1) P_{\ell+1}^{(k)}(a,b,c), \tag{2.8}$$

the term-by-term differentiation being valid for $|z| < \sqrt{c/a}$.

On the other hand, differentiating the expression $f_k(z) = (\sqrt{az^2+bz+c})^k$ and then expanding gives

$$\begin{aligned} f_k'(z) &= \frac{k}{2}(az^2+bz+c)^{(k-2)/2}(2az+b) \\ &= \frac{k}{2} f_{k-2}(z)(2az+b) \\ &= \frac{k}{2} c^{(k-2)/2} \sum_{\ell=0}^{\infty} \left(\frac{z}{c}\right)^{\ell} (2az+b) P_{\ell}^{(k-2)}(a,b,c) \\ &= c^{(k-2)/2} \left\{ \sum_{\ell=0}^{\infty} \left(\frac{z}{c}\right)^{\ell} \frac{k}{2} b P_{\ell}^{(k-2)}(a,b,c) + \sum_{\ell=1}^{\infty} \left(\frac{z}{c}\right)^{\ell} kac P_{\ell-1}^{(k-2)}(a,b,c) \right\}. \end{aligned} \tag{2.9}$$

Equating (2.8) and (2.9) and then comparing coefficients gives

$$(\ell+1) P_{\ell+1}^{(k)}(a,b,c) = \frac{k}{2} b P_{\ell}^{(k-2)}(a,b,c) + kac P_{\ell-1}^{(k-2)}(a,b,c), \quad \ell \in \mathbb{N}. \tag{2.10}$$

Using the obvious recurrence on f_k and then expanding gives

$$\begin{aligned} f_k(z) &= (az^2+bz+c) f_{k-2}(z) \\ &= c^{k/2} \left\{ \sum_{\ell=2}^{\infty} \left(\frac{z}{c}\right)^{\ell} ac P_{\ell-2}^{(k-2)}(a,b,c) + \sum_{\ell=1}^{\infty} \left(\frac{z}{c}\right)^{\ell} b P_{\ell-1}^{(k-2)}(a,b,c) \right. \\ &\qquad \left. + \sum_{\ell=0}^{\infty} \left(\frac{z}{c}\right)^{\ell} P_{\ell}^{(k-2)}(a,b,c) \right\}. \end{aligned} \tag{2.11}$$

Equating (2.3) and (2.11) and then comparing coefficients gives

$$P_{\ell+1}^{(k)}(a,b,c) = P_{\ell+1}^{(k-2)}(a,b,c) + b P_{\ell}^{(k-2)}(a,b,c) + ac P_{\ell-1}^{(k-2)}(a,b,c), \quad \ell \in \mathbb{N}. \tag{2.12}$$

To obtain (2.7), multiply (2.12) by $(\ell+1)$ and equate to (2.10). Solving for $P_{\ell+1}^{(k-2)}(a,b,c)$ and making the index change $(k-2) \mapsto k$ gives (2.7).

This completes the proof when $a, c > 0$ and $b^2 \leq 4ac$. This set in $\mathbb{R}^3$ contains a nontrivial open ball, and polynomials in n variables are determined everywhere by their behavior on any nontrivial open ball in $\mathbb{R}^n$. Hence (2.7) holds for all $a, b, c \in \mathbb{R}$ since the right- and left-hand sides of (2.7) are polynomial. □

3. Multivariate expansions. Let $\Phi(x) = \left(x^2+\tau^2\right)^{k/2}$, where $\tau \geq 0$ and $k \in \mathbb{Z}$ is odd and where we have used the notational convenience $x^2 = \langle x, x\rangle = |x|^2$ for $x \in \mathbb{R}^n$. The following result gives a far field expansion for $\Phi(x-t)$ considered as a function of x, together with an error estimate for approximation with truncations of this expansion. The numerator polynomials $P_\ell^{(k)}(t^2+\tau^2,-2\langle t,x\rangle,x^2)$ that feature in the expansion are homogeneous of degree ℓ in x. Correspondingly, the ℓth term in the expansion is homogeneous of degree $k-\ell$ in x.

Lemma 3.1. *Let $k\in\mathbb{Z}$ be odd, $t\in\mathbb{R}^n$, and $\tau\geq 0$. For all $x\in\mathbb{R}^n$ with $|x|>\sqrt{t^2+\tau^2}$,*

$$\Phi(x-t)=\big((x-t)^2+\tau^2\big)^{k/2}=\sum_{\ell=0}^{\infty}P_\ell^{(k)}(t^2+\tau^2,-2\langle t,x\rangle,x^2)/|x|^{2\ell-k},$$

where the polynomials $P_\ell^{(k)}$ are defined in (1.4). *Moreover, for all x such that $|x|>\sqrt{t^2+\tau^2}$, and for all $p\in\mathbb{N}$ such that $p+k>0$,*

$$\left|\Phi(x-t)-\sum_{\ell=0}^{p+k}P_\ell^{(k)}(t^2+\tau^2,-2\langle t,x\rangle,x^2)/|x|^{2\ell-k}\right|$$
$$\leq\begin{cases}\big(2\sqrt{t^2+\tau^2}\big)^k\left(\dfrac{\sqrt{t^2+\tau^2}}{|x|}\right)^{p+1}\dfrac{|x|}{|x|-\sqrt{t^2+\tau^2}} & \text{if } k>0,\\[2ex] \dbinom{p}{p+k+1}\big(\sqrt{t^2+\tau^2}\big)^k\left(\dfrac{\sqrt{t^2+\tau^2}}{|x|}\right)^{p+1} \times\left(\dfrac{|x|}{|x|-\sqrt{t^2+\tau^2}}\right)^{-k} & \text{if } k<0.\end{cases}$$

Proof. Consider firstly the case when $\tau>0$. Let $a=t^2+\tau^2$, $b=-2\langle t,x\rangle$, and $c=x^2$. Then

$$\Phi(x-t)=\big(x^2-2\langle t,x\rangle+t^2+\tau^2\big)^{k/2}=f_k(1),$$

where f_k is the function defined in (2.1). Since $a,c>0$, $b^2\leq 4ac$, and $1=|z|<\sqrt{c/a}=|x|/\sqrt{t^2+\tau^2}$, Lemma 2.3 may be applied with $\nu=p+k$ to yield the desired results when we recall the bound on F given by (2.6).

This completes the proof when $\tau>0$. For the remaining case fix x with $|x|>|t|$. Note that $0<\widetilde{\tau}<\sqrt{|x|^2-|t|^2}$ implies $|x|>\sqrt{t^2+\widetilde{\tau}^2}$. Hence the previous case can be applied to the expansion of

$$\Phi(x-t;k,\widetilde{\tau})=\big((x-t)^2+\widetilde{\tau}^2\big)^{k/2}$$

for all sufficiently small positive $\widetilde{\tau}$. Taking the limit as $\widetilde{\tau}$ goes to zero from above, and using the continuity of all the relevant quantities as functions of $\widetilde{\tau}$, gives the result for $\tau=0$. □

Example 3.2. In the one-dimensional case it is convenient to rewrite the series in the simpler form

$$\Phi(x-t)=\operatorname{sign}(x)\sum_{\ell=0}^{\infty}P_\ell^{(k)}(t^2+\tau^2,-2t,1)/x^{\ell-k},$$

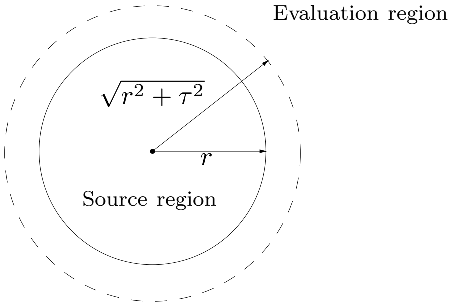


FIG. 3.1. *Region of validity of the far field expansion of a cluster.*

which becomes, in the important special case ($k=1$) of the ordinary multiquadric,

$$\begin{aligned}\sqrt{(x-t)^2+\tau^2} = \operatorname{sign}(x)\Big\{ & x - t + \frac{1}{2}\tau^2 x^{-1} + \frac{1}{2}t\tau^2 x^{-2} \\ & + \frac{1}{8}(4t^2\tau^2-\tau^4)x^{-3} + \frac{1}{8}(4t^3\tau^2 - 3t\tau^4)x^{-4} \\ & + \frac{1}{16}\tau^2(8t^4-12t^2\tau^2+\tau^4)x^{-5} + \cdots + q_\ell(t,\tau)x^{1-\ell} + \cdots\Big\}.\end{aligned}$$

Example 3.3. To display the componentwise form of the expansion in two dimensions we will temporarily adopt the notation $x=(x_1,x_2)$ and $t=(t_1,t_2)$. The far field expansion about zero of a single ordinary multiquadric basic function centered at t is then

$$\begin{aligned}&\sqrt{|x-t|^2+\tau^2} \\ &\quad = |x| - \frac{t_1x_1+t_2x_2}{|x|} + \frac{1}{2}\frac{(t_2^2+\tau^2)x_1^2 + (t_1^2+\tau^2)x_2^2 - 2t_1t_2x_1x_2}{|x|^3} \\ &\quad\quad + \frac{1}{2}\frac{(t_1x_1+t_2x_2)\left\{(t_2^2+\tau^2)x_1^2+(t_1^2+\tau^2)x_2^2-2t_1t_2x_1x_2\right\}}{|x|^5} + \cdots .\end{aligned}$$

Since the bound of Lemma 3.1 is increasing in $|t|$ we can apply it to each center in a cluster and sum, obtaining the following expansion of the generalized multiquadric radial basis function associated with a cluster of centers. The geometry of the source cluster and the evaluation region is shown in Figure 3.1.

THEOREM 3.4. *Suppose $t_i \in \mathbb{R}^n$, $|t_i| \le r$, and $d_i \in \mathbb{R}$ for each $1 \le i \le N$. Let k be odd, $\tau \ge 0$, and s be the generalized multiquadric spline*

$$s(x) = \sum_{i=1}^{N} d_i\Phi(x-t_i) = \sum_{i=1}^{N} d_i\left(\sqrt{(x-t_i)^2+\tau^2}\right)^k .$$

If $P_\ell^{(k)}$, $\ell \in \mathbb{N}_0$, are the polynomials defined by (1.4), *then the polynomials*

$$Q_\ell(x) = \sum_{i=1}^{N} d_i P_\ell^{(k)}(t_i^2+\tau^2, -2\langle t_i, x\rangle, x^2), \qquad \ell \in \mathbb{N}_0,$$

have the following property: Let $p \in \mathbb{N}_0$ and set

$$s_p(x) = \sum_{\ell=0}^{p+k} Q_\ell(x)/|x|^{2\ell-k}, \tag{3.1}$$

$x \in \mathbb{R}^n \backslash \{0\}$. *Then for all* x *with* $|x| > R = \sqrt{r^2 + \tau^2}$,

$$\left|s(x) - s_p(x)\right| \le \begin{cases} 2^k M R^k \left(\dfrac{1}{c}\right)^{p+1} \dfrac{1}{1 - 1/c} & \text{if } k > 0, \\ \dbinom{p}{p+k+1} M R^k \left(\dfrac{1}{c}\right)^{p+1} \left(\dfrac{1}{1-1/c}\right)^{-k} & \text{if } k < 0, \end{cases}$$

where $M = \sum_{i=1}^{N} |d_i|$ *and* $c = |x|/R$.

4. The uniqueness of expansions. The uniqueness of far field expansions is important for two reasons. First, redundant coefficients could mean that a small value is represented as the difference of two large values leading to numerical instability. Second, if the far field expansion of a fixed function, $s(x) = \sum_{i=1}^{N} \Phi(x - t_i)$, is unique, then it is often possible to shift the center of a truncated expansion indirectly without using any knowledge of the underlying centers and weights. The advantage of such indirect shifting over direct series formation is a flop count which depends only on the number of terms in the expansion, and not on the number of centers in the cluster. This can result in significantly faster code. Furthermore, since the uniqueness implies that the indirectly obtained series is identical with that which would have been obtained directly, the indirectly obtained series enjoys the same error bound as the directly obtained one.

We will now prove a general uniqueness lemma from which uniqueness of series expansions of the form (3.1) follows as a special case. Recall that a function g defined for all x in some subset $D \subset \mathbb{R}^n$ is said to be homogeneous of degree γ on D if

$$g(\lambda x) = \lambda^{\gamma} g(x)$$

for all $\lambda > 0$ and $x \in \mathbb{R}^n$ such that both x and $\lambda x \in D$. (Some authors use the term positively homogeneous of degree γ for this property.)

LEMMA 4.1. *Suppose* $\gamma, R \in \mathbb{R}$ *and that a function* $f : D \subset \mathbb{R}^n \to \mathbb{R}$ *can be expanded in two ways,*

$$\sum_{\ell=0}^{\infty} U_\ell(x) = f(x) = \sum_{\ell=0}^{\infty} V_\ell(x),$$

both series converging absolutely and uniformly to $f(x)$ *for all* $|x| \ge R$, *where for each* ℓ , U_ℓ *and* V_ℓ *are continuous homogeneous functions of degree* $\gamma - \ell$. *Then for each* ℓ, $U_\ell(x) = V_\ell(x)$ *for all* $|x| \ge R$.

Proof. Since the absolute series converge uniformly on $|x| = R$, there exists an $M < \infty$ such that

$$\max_{|x|=R} \{\max\{|U_\ell(x)|,\ |V_\ell(x)|\}\} \le M$$

for all $\ell \in \mathbb{N}_0$. Hence, using the homogeneity,

$$\max\{|U_\ell(x)|,\ |V_\ell(x)|\} \le M|x|^{\gamma-\ell}/R^{\gamma-\ell} \tag{4.1}$$

for all x such that $|x| \ge R$ and for all $\ell \in \mathbb{N}_0$.

Now suppose U_ℓ and V_ℓ differ for some ℓ's. Let j be the first index for which they differ. Then for all $|x| \geq R$,

$$\begin{aligned} 0 &= \left(\frac{|x|}{R}\right)^{j-\gamma} \{f(x) - f(x)\} \\ &= \left(\frac{|x|}{R}\right)^{j-\gamma} \{U_j(x) - V_j(x)\} + \sum_{\ell>j} \left(\frac{|x|}{R}\right)^{j-\gamma} \{U_\ell(x) - V_\ell(x)\}. \qquad (4.2) \end{aligned}$$

But from (4.1)

$$\begin{aligned} \left|\sum_{\ell>j} \left(\frac{|x|}{R}\right)^{j-\gamma} \{U_\ell(x) - V_\ell(x)\}\right| &\leq 2M \sum_{\ell>j} \left(\frac{|x|}{R}\right)^{j-\ell} \\ &= o(1) \qquad \text{as } |x| \to \infty. \end{aligned}$$

Hence from (4.2)

$$|U_j(x) - V_j(x)| = o(|x|^{\gamma-j}) \quad \text{as } |x| \to \infty.$$

Since $U_j - V_j$ is homogeneous of degree $\gamma - j$ on D, this implies that it is identically zero on D. □

5. Efficient formation of the far field series. In the previous sections we have developed far field expansions with the intention of using them for fast evaluation of generalized multiquadric radial basis functions. In order that these expansions be suitable for this task they must be inexpensive both to form and to evaluate. The purpose of this section is to show that the expansions can be formed in an efficient recursive manner.

Given a single center $t \in \mathbb{R}^n$ with unit weight, the corresponding truncated expansion of section 3 is

$$\Phi(x-t) = \left((x-t)^2 + \tau^2\right)^{k/2} = \sum_{\ell=0}^{\infty} P_\ell^{(k)}(t^2+\tau^2, -2\langle t, x\rangle, x^2)/|x|^{2\ell-k}. \qquad (5.1)$$

Writing $G_\ell(x) = P_\ell^{(k)}(t^2+\tau^2, -2\langle t, x\rangle, x^2)$, G_ℓ is a homogeneous polynomial of degree ℓ in x, with coefficients depending on k, τ, and t. The expansion for a single center, with corresponding weight d, then becomes

$$\sum_{\ell=0}^{p+k} dG_\ell(x)/|x|^{2\ell-k}. \qquad (5.2)$$

The expansion of a cluster is formed by summing the expansions (5.2) corresponding to each center and has the form

$$\sum_{\ell=0}^{p+k} Q_\ell(x)/|x|^{2\ell-k}, \qquad (5.3)$$

where each Q_ℓ is a homogeneous polynomial of degree ℓ. Lemma 2.5 implies that the polynomials G_ℓ satisfy the three-term recurrence

$$G_\ell(x) = \begin{cases} 1, & \ell = 0, \\ -k\langle x, t\rangle, & \ell = 1, \\ A_\ell \langle x, t\rangle G_{\ell-1}(x) + B_\ell\, x^2(t^2+\tau^2)\, G_{\ell-2}(x), & \ell \geq 2, \end{cases} \qquad (5.4)$$

where

$$A_\ell = -2\frac{k/2-\ell+1}{\ell}, \qquad B_\ell = -\frac{\ell-k-2}{\ell}.$$

The recurrence is very simple to implement, as is demonstrated by the following code fragment for the special case of two dimensions. The code fragment employs the notation of Example 3.3.

Code fragment to generate the numerator polynomial coefficients in the expansion of a generalized multiquadric in two dimensions.

Input: A center $t \in \mathbb{R}^2$, the corresponding weight d, the generalized multiquadric parameters k and τ, and the desired order of expansion p.
Output: The coefficients $G(\ell, j)$ of the homogeneous numerator polynomials in the expansion of this single center. On output $G(\ell, j)$ is the coefficient of $x_1^{\ell-j}x_2^j$ in the homogeneous polynomial dG_ℓ of (5.2).

```
PolynomialGenerator($t$, $d$, $k$, $\tau$, $p$)
    $G(0,0) = d$, $G(1,0) = -d * k * t_1$, $G(1,1) = -d * k * t_2$
    for $\ell = 2$ to $p + k$
        $a = A_\ell$, $b = B_\ell * (|t|^2 + \tau^2)$
        $\mathrm{tmp} = a * G(\ell - 1, 0)$
        $G(\ell, 0) = \mathrm{tmp} * t_1$
        $G(\ell, 1) = \mathrm{tmp} * t_2$
        for $j = 0$ to $\ell - 2$
            $\mathrm{tmp} = b * G(\ell - 2, j)$
            $G(\ell, j) = G(\ell, j) + \mathrm{tmp}$
            $G(\ell, j + 2) = \mathrm{tmp}$
            $\mathrm{tmp} = a * G(\ell - 1, j + 1)$
            $G(\ell, j + 1) = G(\ell, j + 1) + \mathrm{tmp} * t_1$
            $G(\ell, j + 2) = G(\ell, j + 2) + \mathrm{tmp} * t_2$
        end
    end
```

Recall that the $\binom{\ell+n-1}{\ell}$ monomials of exact degree ℓ, $\{x^\alpha : |\alpha| = \ell\}$, form a basis for the homogeneous polynomials of degree ℓ on $\mathbb{R}^n$. Represent the polynomials G_ℓ in terms of these monomials, i.e., let

$$G_\ell(x) = \sum_{|\alpha|=\ell} a_\alpha^\ell x^\alpha,$$

for some coefficients a_α^ℓ. Then, from the recurrence (5.4), each coefficient of G_ℓ can be calculated using at most n coefficients of $G_{\ell-1}$ and at most n coefficients of $G_{\ell-2}$. Specifically, if e_i is the multi-index with 1 in the ith position and 0 elsewhere, then the recurrence (5.4) implies that for $\ell \geq 2$,

$$a_\alpha^\ell = A_\ell \sum_{i=1}^n t_i a_{\alpha-e_i}^{\ell-1} + B_\ell(t^2 + \tau^2) \sum_{i=1}^n a_{\alpha-2e_i}^{\ell-2},$$

where a_β^j is taken to be zero if any component of β is negative. It follows that all the numerator polynomials $\{Q_\ell\}_{\ell=0}^{p+k}$ in the truncated expansion (5.3) of an m center cluster can be formed (that is, their $\binom{n+p+k}{p+k}$ coefficients calculated) in $\mathcal{O}\big(mn\binom{n+p+k}{p+k}\big)$ floating point operations. This quantity is $\mathcal{O}(mn(p+k)^n)$ when the dimension n is less than the degree $p+k$.

6. A subspace of polynomials.

In this section we will investigate a subspace of polynomials in n variables. This space will arise in section 7, and the aim of that section will be to translate a member of this subspace. It will be shown that, modulo a low-degree polynomial, this subspace is closed under translation of the underlying Cartesian coordinate system.

Throughout this section and the next, n will be fixed and any complexity estimates will be expressed as a function of polynomial degree only. Thus a typical estimate might take the form $\mathcal{O}((p+k)^n)$. In such expressions multiplicative order constants depending on n have been suppressed, and we will be interested in the estimate only when the argument $p+k$ is bigger than n.

The following standard spaces will be used:

- π_j^n, polynomials of total degree not exceeding j in n variables.
- $\mathcal{H}_j^n$, homogeneous polynomials of degree j in n variables.

Also, for given function spaces S and T, define new spaces as follows:

$$\begin{aligned} S\,T &= \{s(\cdot)\,t(\cdot)\ :\ s\in S,\ t\in T\}, && \\ S\oplus T &= \{s(\cdot)+t(\cdot)\ :\ s\in S,\ t\in T\}, && \qquad S\cap T=\{0\}, \\ s\,T &= \{s(\cdot)\,t(\cdot)\ :\ t\in T\}, && \qquad s\in S. \end{aligned}$$

The subspaces of polynomials that are the subject of this section are defined by

$$S_j^n = \left\{ q\in\pi_{2j}^n : q(\cdot) = \sum_{\ell=0}^{j} q_\ell(\cdot)|\cdot|^{2(j-\ell)}, q_\ell\in\mathcal{H}_\ell^n \right\}. \tag{6.1}$$

Apart from 0, the polynomials of S_j^n have total degree no greater than $2j$ and no less than j. It follows from Lemma 4.1 that $q\in S_n^j$ is uniquely determined by the homogeneous polynomials $\{q_\ell\}_{\ell=0}^j$ and thus by the coefficients of those polynomials with respect to some appropriate basis. Hence

$$\dim S_j^n = \sum_{\ell=0}^{j} \dim H_\ell^n. \tag{6.2}$$

THEOREM 6.1. *S_j^n is invariant under orthogonal transformation of the underlying coordinate system, i.e., if $q\in S_j^n$, then $q(Q\,\cdot)\in S_j^n$ for orthogonal Q.*

Proof. The component function $f_i(\cdot)=(Q\cdot)_i$ is nontrivial for each i since Q is invertible. Hence, since the component functions are also linear, each f_i is homogeneous of exact degree 1. Thus

$$(Q\cdot)^\alpha = (Q\cdot)_1^{\alpha_1}(Q\cdot)_2^{\alpha_s}\dots(Q\cdot)_n^{\alpha_n}$$

is homogeneous of exact degree $|\alpha|$. It follows that $q_\ell(Q\,\cdot)$ is homogeneous of degree ℓ if q_ℓ is. Finally, since Q is orthogonal,

$$|Q\,\cdot\,| = |\cdot|,$$

and the result follows. □

Before we prove translation invariance of S_j^n we will make a few simple observations regarding these spaces.

LEMMA 6.2. *The spaces S_j^n satisfy the following relations:*

(i) $S^n_{j+1} = \big(|\cdot|^2\, S^n_j\big) \oplus \mathcal{H}^n_{j+1}$,
(ii) $\mathcal{H}^n_1\, S^n_j \subset S^n_{j+1}$,
(iii) $S^n_j \subset S^n_{j+1} \oplus \mathcal{H}^n_j$.

Proof. Let $q \in S^n_{j+1}$ and let $\{q_\ell\}_{\ell=0}^{j+1}$ be the polynomials such that

$$q = \sum_{\ell=0}^{j+1} |\cdot|^{2(j+1-\ell)} q_\ell, \qquad q_\ell \in \mathcal{H}^n_\ell.$$

The observation that

$$q = |\cdot|^2 h + q_{j+1},$$

where

$$h = \left(\sum_{\ell=0}^{j} |\cdot|^{2(j-\ell)} q_\ell \right) \in S^n_j,$$

proves part (i).

Now let $p \in \mathcal{H}^n_1$. Then for each ℓ, $0 \le \ell \le j$, the product $\tilde{q}_{\ell+1} = pq_\ell \in \mathcal{H}^n_{\ell+1}$. Thus

$$pq = \sum_{\ell=0}^{j} |\cdot|^{2(j-\ell)} \tilde{q}_{\ell+1} = \sum_{k=1}^{j+1} |\cdot|^{2(j+1-k)} \tilde{q}_k \in S^n_{j+1},$$

which shows part (ii).

For part (iii),

$$\begin{aligned} q(x) = \sum_{\ell=0}^{j} q_\ell(x)|x|^{2(j-\ell)} &= q_j(x) + \sum_{\ell=0}^{j-1} q_\ell(x)|x|^2\, |x|^{2(j-1-\ell)} \\ &= q_j(x) + \sum_{\ell=0}^{j-1} \tilde{q}_{\ell+2}(x)\, |x|^{2(j-1-\ell)} \\ &= q_j(x) + \sum_{\ell=2}^{j+1} \tilde{q}_\ell(x)\, |x|^{2(j+1-\ell)} \in \mathcal{H}^n_j \oplus S^n_{j+1}, \end{aligned}$$

since the polynomials $\tilde{q}_{\ell+2}(\cdot) = q_\ell(\cdot)|\cdot|^2$ are homogeneous of degree $\ell + 2$. □

THEOREM 6.3. *S^n_j are translation invariant modulo polynomials of degree $j-1$, i.e., for any $q \in S^n_j$ and $u \in \mathbb{R}^n$, $q(\cdot - u) \in S^n_j \oplus \pi^n_{j-1}$.*

Proof. The proof is by induction on j. The result is trivially true in the case $j = 0$ since S^n_0 is the space of constants and π^n_{-1} is the singleton $\{0\}$.

Now assume the result for $k = 0, 1, 2, \ldots, j$, let $q \in S^n_{j+1}$, and let $u \in \mathbb{R}^n$. Then by Lemma 6.2(i),

$$q(x-u) = |x-u|^2\, h(x-u) + q_{j+1}(x-u), \tag{6.3}$$

where $h \in S^n_j$ and $q_{j+1} \in \mathcal{H}^n_{j+1}$. By the induction hypothesis, $h(\cdot - u) \in S^n_j \oplus \pi^n_{j-1}$. Thus

$$h(x-u) = \tilde{h}_j(x) + \tilde{h}_{j-1}(x) + \tilde{h}_<(x), \tag{6.4}$$

where $\tilde{h}_j \in S_j^n$, $\tilde{h}_{j-1} \in \mathcal{H}_{j-1}^n$, and $\tilde{h}_< \in \pi_{j-2}^n$. Since $q_{j+1} \in \mathcal{H}_{j+1}^n$,

$$q_{j+1}(x-u) = q_{j+1}(x) - \tilde{q}_<(x), \tag{6.5}$$

where $\tilde{q}_< \in \pi_j^n$. Expand (6.3) to get

$$q(x-u) = \big(|x|^2 - 2\langle x,u\rangle + |u|^2\big)\Big(\tilde{h}_j(x) + \tilde{h}_{j-1}(x) + \tilde{h}_<(x)\Big) + q_{j+1}(x) + \tilde{q}_<(x). \tag{6.6}$$

Consider each term in the expansion of this product:

$$\begin{aligned} |\cdot|^2\tilde{h}_j &\in S_{j+1}^n && \text{by Lemma 6.2(i)},\\ -2\langle\cdot,u\rangle\,\tilde{h}_j &\in S_{j+1}^n && \text{by Lemma 6.2(ii)},\\ |u|^2\tilde{h}_j &\in S_j^n \subset S_{j+1}^n \oplus \mathcal{H}_j^n && \text{by Lemma 6.2(iii)},\\ |\cdot|^2\tilde{h}_{j-1} &\in S_{j+1}^n && \text{since } |\cdot|^2\mathcal{H}_{j-1}^n \subset \mathcal{H}_{j+1}^n \subset S_{j+1}^n,\\ -2\langle\cdot,u\rangle\,\tilde{h}_{j-1} &\in \mathcal{H}_j^n,\\ |u|^2\tilde{h}_{j-1} + |\cdot - u|^2\tilde{h}_< &\in \pi_j^n. \end{aligned}$$

Thus it follows that $q(\cdot - u) \in S_{j+1}^n \oplus \pi_j^n$. The result follows by induction. □

In computations, a polynomial $p \in S_j^n$ may be known in terms of the monomial basis, but what is actually required are the polynomials $\{q_\ell\}_{\ell=0}^j$ such that

$$p(x) = \sum_{\ell=0}^{j} q_\ell(x)|x|^{2(j-\ell)}. \tag{6.7}$$

Since the polynomials $\{q_\ell\}$ are homogeneous, for a given ℓ, q_ℓ must be determined entirely by those terms of p that are homogeneous of degree $2j-\ell$. Thus the problem of determining $\{q_\ell\}$ may be broken down into homogeneous parts. Hence, without loss of generality, assume that p is a given homogeneous polynomial of degree $\ell + 2k$ such that

$$p(x) = |x|^{2k}q(x) \tag{6.8}$$

with q unknown and to be determined from p. Since

$$p(x) = |x|^{2k}q(x) = |x|^2\big(|x|^{2(k-1)}q(x)\big),$$

if q can be determined in the case where $k = 1$, the more general problem may be solved in an inductive manner.

Let $\{p_j\}_{j=0}^{\ell+2}$ and $\{q_i\}_{i=0}^{\ell}$ be homogeneous polynomials in $x_2, \dots, x_n$ such that

$$p(x) = \sum_{j=0}^{\ell+2} x_1^{\ell+2-j}p_j(\bar{x}), \quad q(x) = \sum_{i=0}^{\ell} x_1^{\ell-i}q_i(\bar{x}), \quad \text{and} \quad p(x) = |x|^2q(x),$$

where, if $x = (x_1, \dots, x_n)$, then $\bar{x} = (x_2, \dots, x_n)$. Using this same notation,

$$|x|^2 = x_1^2 + |\bar{x}|^2$$

and hence

$$\begin{aligned}\sum_{j=0}^{\ell+2} x_1^{\ell+2-j} p_j(\bar{x}) &= \left(x_1^2+|\bar{x}|^2\right)\sum_{i=0}^{\ell} x_1^{\ell-i} q_i(\bar{x})\\ &= x_1^{\ell+2}q_0(\bar{x}) + x_1^{\ell+1}q_1(\bar{x}) + \left\{\sum_{i=2}^{\ell} x_1^{\ell+2-i}\big(q_i(\bar{x})+|\bar{x}|^2 q_{i-2}(\bar{x})\big)\right\}\\ &\qquad + x_1|\bar{x}|^2 q_{\ell-1}(\bar{x}) + |\bar{x}|^2 q_\ell(\bar{x}).\end{aligned}$$

Equating coefficients, we may now write the polynomials q_i in terms of the polynomials p_j:

$$\begin{aligned} q_0(\bar{x}) &= p_0(\bar{x}),\\ q_1(\bar{x}) &= p_1(\bar{x}),\\ q_2(\bar{x}) &= p_2(\bar{x}) - |\bar{x}|^2 q_0(\bar{x}),\\ q_3(\bar{x}) &= p_3(\bar{x}) - |\bar{x}|^2 q_1(\bar{x}),\\ &\ \ \vdots\\ q_{\ell-1}(\bar{x}) &= p_{\ell-1}(\bar{x}) - |\bar{x}|^2 q_{\ell-3}(\bar{x}),\\ q_\ell(\bar{x}) &= p_\ell(\bar{x}) - |\bar{x}|^2 q_{\ell-2}(\bar{x}).\end{aligned}$$

Multiplication of a polynomial by a monomial corresponds to a relabeling of coefficients and computationally corresponds to assignment or addition. Since

$$|\bar{x}|^2 = x_2^2 + \cdots + x_n^2$$

is just the sum of $n-1$ monomials, for fixed i the product $|\cdot|^2 q_i(\cdot)$ may be calculated with $\mathcal{O}(nC_i)$ additions, where $C_i = \dim \mathcal{H}_i^{n-1}$. It is well known that

$$\dim \mathcal{H}_i^{n-1} = \binom{i+n-2}{n-2} = \frac{(i+n-2)!}{i!(n-2)!} = \frac{1}{(n-2)!}\big((i+n-2)\cdots(i+1)\big) = \mathcal{O}(i^{n-2}),$$

and hence $|\cdot|^2 q_i(\cdot)$ may be calculated in $\mathcal{O}(i^{n-2})$ operations. It now follows that all of the polynomials $\{q_i\}_{i=0}^{\ell}$ may be calculated in $\mathcal{O}(\ell^{n-1})$ operations.

Since the more general problem of (6.8) may be solved by k applications of this simpler case, $q(x) = p(x)/|x|^{2k}$ may be calculated in

$$\sum_{i=0}^{k-1} \mathcal{O}\big((\ell+2i)^{n-1}\big) = \mathcal{O}\big((\ell+2k)^n\big)$$

operations. Applying this to each homogeneous part of (6.7) gives the following lemma.

Lemma 6.4. *Let $n \in \mathbb{N}$. There exists a constant C depending only on n with the following property. Given any polynomial $p \in S_j^n$, the polynomials $\{q_\ell\}_{\ell=0}^{j}$ such that $q_\ell \in \mathcal{H}_\ell^n$ and*

$$p = \sum_{\ell=0}^{j} |\cdot|^{2(j-\ell)} q_\ell$$

may be determined in no more than Cj^{n+1} operations.

7. Translation of a far field expansion. The uniqueness of the far field expansions makes it possible to shift the center of a truncated expansion knowing only its coefficients and without any direct knowledge of the underlying centers and weights. As the operation count for indirect translation depends on the length of the series, not the number of centers, indirect translation can be significantly faster than direct formation of series for clusters with many centers.

The precise problem we address is the following. Let

$$s_p(x) = \sum_{\ell=0}^{p+k} Q_\ell(y)/|y|^{2\ell-k}, \qquad y = x - u \neq 0, \tag{7.1}$$

where Q_ℓ are homogeneous polynomials of degree ℓ, be an expansion similar to (3.1) or (5.3), but centered at $u \neq 0$ rather than 0. We wish to shift the center of expansion to the origin. That is, we seek homogeneous polynomials $\{\widehat{Q}_\ell\}$, $\widehat{Q}_\ell$ being of degree ℓ, so that

$$s_p(x) = \sum_{\ell=0}^{p+k} \widehat{Q}_\ell(x)/|x|^{2\ell-k} + \mathcal{O}(1/|x|^{p+1}) \tag{7.2}$$

as $|x| \to \infty$. We will show that translations of truncated expansions of the form (7.1) into expansions of the form (7.2) may be performed in $\mathcal{O}\left((p+k)^{n+1}\right)$ operations using simple polynomial manipulations.

7.1. The cost of multiplication. In this subsection it will be shown that the product of two homogeneous polynomials of degree ℓ in n variables may be computed in $\mathcal{O}(\ell^{n-1}\log \ell)$ operations.

Let p be a homogeneous polynomial of degree ℓ. Since p is homogeneous,

$$p(x) = p(x_1, x_2, \dots, x_n) = x_n^\ell p\left(\frac{x_1}{x_n}, \frac{x_2}{x_n}, \dots, \frac{x_{n-1}}{x_n}, 1\right), \qquad x_n \neq 0.$$

Furthermore given $x_n^\ell p(...)$ for all x with $x_n \neq 0$, $p(x)$ can be recovered on the hyperplane $x_n = 0$ by continuity. Thus for the purposes of the multiplication and division that are the subject of this section, we may consider multiplication and division of general, that is, probably inhomogeneous, polynomials of degree ℓ in $n-1$ variables, rather than of homogeneous polynomials of degree ℓ in n variables.

Let p and q be two polynomials of degree ℓ in $n-1$ variables. Then their product is

$$p(x)q(x) = \left(\sum_{|\alpha|\le\ell} a_\alpha x^\alpha\right)\left(\sum_{|\beta|\le\ell} b_\beta x^\beta\right) = \sum_{|\alpha|\le 2\ell}\left(\sum_{0\le\beta\le\alpha} a_\beta b_{\alpha-\beta}\right)x^\alpha,$$

the Cauchy product. The convolution producing the coefficients of the product can be computed in $\mathcal{O}(\ell^{n-1}\log \ell)$ operations by FFTs. It now follows that the homogeneous polynomial multiplication above can also be carried out in $\mathcal{O}(\ell^{n-1}\log \ell)$ operations.

7.2. Translation by convolution. In this subsection it will be shown that translation of the far field series may be performed by convolution.

Throughout this subsection when we speak of forming a polynomial, we mean finding its coefficients with respect to a basis, usually the monomial basis. When we speak of forming a truncated expansion of the type (5.3), we mean finding the coefficients of all the relevant numerator polynomials.

First we set

$$Q(y) = \sum_{\ell=0}^{p+k} Q_\ell(y)|y|^{2(p+k-\ell)}. \tag{7.3}$$

Then

$$s_p(x) = Q(y)/|y|^{2p+k}, \qquad y = x - u \neq 0. \tag{7.4}$$

Since we already have all of the Q_ℓ, all we need to do to form Q is form the polynomials $|\cdot|^{2(p+k-\ell)}$ and then form the products $Q_\ell(\cdot)|\cdot|^{2(p+k-\ell)}$. Form $|\cdot|^{2j}$, $j = 0, \ldots, p+k$, once and store. Each $|\cdot|^{2j-2}$ is homogeneous of degree $2j-2$ and therefore involves $\mathcal{O}(j^{n-1})$ coefficients. The polynomial $|\cdot|^{2j}$ may be obtained from $|\cdot|^{2j-2}$ with n additions for each coefficient in $|\cdot|^{2j-2}$. Hence the cost of forming the $|\cdot|^{2j}$'s is $\mathcal{O}((p+k)^n)$ operations. Each of the products $Q_\ell(\cdot)|\cdot|^{2(p+k-\ell)}$ is the product of two homogeneous polynomials and is of degree no greater than $2(p+k)$. Hence we can calculate each product in $\mathcal{O}\big((p+k)^{n-1}\log(p+k)\big)$ operations. As there are $p+k+1$ of these products in Q, forming Q takes $\mathcal{O}\big((p+k)^n \log(p+k)\big)$ operations.

We proceed to shift the center of expansion of Q by setting

$$\widetilde{Q}(x) = Q(x-u), \qquad x \in \mathcal{R}^n. \tag{7.5}$$

A translation of this sort can be done simply and quickly by convolution. For example, using the scaled monomial basis $V_\alpha(x) = x^\alpha/\alpha!$ (α a multi-index), we have

$$\begin{aligned}
p(x-u) &= \sum_{|\alpha|<k} a_\alpha\, V_\alpha(x-u) \\
&= \sum_{|\alpha|<k} a_\alpha \frac{(x-u)^\alpha}{\alpha!} \\
&= \sum_{|\alpha|<k} \frac{a_\alpha}{\alpha!} \sum_{\beta<\alpha} \binom{\alpha}{\beta} x^\beta (-u)^{(\alpha-\beta)} \\
&= \sum_{|\alpha|<k} a_\alpha \sum_{\beta<\alpha} \frac{x^\beta}{\beta!} \frac{(-u)^{(\alpha-\beta)}}{(\alpha-\beta)!} \\
&= \sum_{|\beta|<k} \frac{x^\beta}{\beta!} \sum_{\alpha<\beta} a_\alpha \frac{(-u)^{(\alpha-\beta)}}{(\alpha-\beta)!}.
\end{aligned}$$

Thus an n-dimensional convolution of $\{a_\alpha\}$ and $\{(-u)^\alpha/\alpha!\}$ gives the coefficients of the translated polynomial. Again this can be computed in $\mathcal{O}\big((p+k)^n \log(p+k)\big)$ operations by an FFT method. This gives us $\widetilde{Q}$ in terms of the monomial or scaled monomial basis.

The next task is to recast $\widetilde{Q}$ into a sum of products of powers of $|x|$ and homogeneous polynomials. By Theorem 6.3 we know that

$$\widetilde{Q}(x) = \sum_{\ell=0}^{p+k} q_\ell(x)|x|^{2(p+k-\ell)} + q_{\text{low}}(x), \tag{7.6}$$

where the q_ℓ are homogeneous of degree ℓ and q_{low} is some polynomial of degree $p+k-1$ or less. By Lemma 6.4, these homogeneous polynomials q_ℓ can be calculated from $\widetilde{Q}$ in $\mathcal{O}\big((p+k)^{n+1}\big)$ operations.

Combining (7.4) and (7.5) and appealing to Lemma 3.1 gives

$$
\begin{aligned}
s_p(x) &= Q(x-u)/|x-u|^{2p+k} \\
&= \widetilde{Q}(x)/|x-u|^{2p+k} \qquad (7.7)\\
&= \widetilde{Q}(x)\sum_{m=0}^{\infty} P_m^{(-2p-k)}(u^2, -2\langle x,u\rangle, x^2)/|x|^{2p+k+2m} \\
&= \left(\sum_{\ell=0}^{p+k} q_\ell(x)|x|^{2(p+k-\ell)} + q_{\text{low}}(x)\right) \\
&\qquad \times \left(\sum_{m=0}^{\infty} P_m^{(-2p-k)}(u^2, -2\langle x,u\rangle, x^2)/|x|^{2p+k+2m}\right) \\
&= \sum_{\ell=0}^{p+k}\sum_{m=0}^{\infty} q_\ell(x)P_m^{(-2p-k)}(u^2, -2\langle x,u\rangle, x^2)/|x|^{2(m+\ell)-k} + \mathcal{O}(1/|x|^{p+1}) \\
&= \sum_{\ell=0}^{p+k}\left(\sum_{j=0}^{\ell} q_j(x)P_{\ell-j}^{(-2p-k)}(u^2, -2\langle x,u\rangle, x^2)\right)/|x|^{2\ell-k} + \mathcal{O}(1/|x|^{p+1}) \\
&= \sum_{\ell=0}^{p+k} \widehat{Q}_\ell(x)/|x|^{2\ell-k} + \mathcal{O}(1/|x|^{p+1}).
\end{aligned}
$$

The sums of products

$$
\widehat{Q}_\ell(x) = \sum_{j=0}^{\ell} q_j(x)P_{\ell-j}^{(-2p-k)}(u^2, -2\langle x,u\rangle, x^2), \quad 0 \le \ell \le p+k, \qquad (7.8)
$$

can be computed simultaneously as homogeneous parts of the product

$$
\left[\sum_{j=0}^{p+k} q_j(\cdot)\right]\left[\sum_{m=0}^{p+k} P_m^{-2p+k}\left(u^2, -2\langle\cdot,u\rangle, (\cdot)^2\right)\right].
$$

Hence they can be computed by a single FFT convolution in $\mathcal{O}\left((p+k)^n \log(p+k)\right)$ operations.

8. Conversion to a near field series. The final step in the process of forming expansions for the FMM is to convert the far field series into a near field, or Taylor, series. At the implementation level, this step is almost identical to the first part of the translation of the far field series.

Define two nonintersecting discs:

$$
\begin{aligned}
D_{\text{eval}} &= \left\{x : |x| \le r\right\}, \\
D_{\text{src}} &= \left\{x : |x-u| \le \sqrt{(\theta r)^2 - \tau^2}\right\}, \qquad \theta > 0.
\end{aligned}
$$

Let

$$
s_p(x) = \sum_{\ell=0}^{p+k} Q_\ell(y)/|y|^{2\ell-k}, \qquad y = x-u \ne 0,
$$

be a far field series, such as (3.1) or (7.1), of $s(x) = \sum_{i=1}^{N} d_i \Phi(x - t_i)$ due to a cluster of centers $\{t_i\}$ located inside D_{src}. Then by Theorem 3.4, s_p approximates s well on D_{eval}. We wish to find a near field series that approximates s_p, and thus s, on D_{eval}.

Proceeding in an identical fashion to section 7.2, we see that we may calculate the polynomial $\widetilde{Q}$ such that

$$s_p(x) = \widetilde{Q}(x)/|x - u|^{2p+k}$$

in $\mathcal{O}\left((p+k)^n \log(p+k)\right)$ operations. When translating the far field expansion to another far field expansion, we essentially convolved $\widetilde{Q}$ with the far field series for $|\cdot - u|^{-(2p+k)}$. To get the near field, all we need do is convolve $\widetilde{Q}$ with the near field series for $|\cdot - u|^{-(2p+k)}$.

The next result gives an explicit expression for the Maclaurin series of $\Phi(\cdot - u) = ((\cdot - u)^2 + \tau^2)^{k/2}$ together with an estimate of the error in approximation by truncating this series. Specializing to the case $\tau = 0$ in this lemma gives the Maclaurin series for $|\cdot - u|^k$.

Lemma 8.1. *Let $k \in \mathbb{Z}$ be odd, and let $u \in \mathbb{R}^n \setminus \{0\}$ and $\tau \geq 0$. For all $x \in \mathbb{R}^n$ with $|x| < \sqrt{u^2 + \tau^2}$,*

(8.1)
$$\Phi(x - u) = \left((x-u)^2 + \tau^2\right)^{k/2} = \sum_{\ell=0}^{\infty} P_\ell^{(k)}(x^2, -2\langle u, x\rangle, u^2 + \tau^2)/\left(\sqrt{u^2 + \tau^2}\right)^{2\ell - k},$$

where the polynomials $P_\ell^{(k)}$ are defined in (1.4). *Moreover,*

$$T_q\big(\Phi(\cdot - u)\big)(x) := \sum_{\ell=0}^{q} P_\ell^{(k)}(x^2, -2\langle u, x\rangle, u^2 + \tau^2)/\left(\sqrt{u^2 + \tau^2}\right)^{2\ell - k} \tag{8.2}$$

is the Maclaurin polynomial of degree q of $\Phi(\cdot - u)$. When $|x| < \sqrt{u^2 + \tau^2}$ and $q \in \mathbb{N}$,

$$\left| \Phi(x - u) - \sum_{\ell=0}^{q} P_\ell^{(k)}(x^2, -2\langle u, x\rangle, u^2 + \tau^2)/\left(\sqrt{u^2 + \tau^2}\right)^{2\ell - k} \right| \tag{8.3}$$
$$\leq \begin{cases} \left(\sqrt{u^2+\tau^2}\right)^k \left(\dfrac{|x|}{\sqrt{u^2+\tau^2}}\right)^{q+1} \dfrac{\sqrt{u^2+\tau^2}}{\sqrt{u^2+\tau^2} - |x|} & \textit{if } k > 0, \\ \dbinom{q-k}{q+1} \left(\sqrt{u^2+\tau^2}\right)^k \left(\dfrac{|x|}{\sqrt{u^2+\tau^2}}\right)^{q+1} \left(\dfrac{\sqrt{u^2+\tau^2}}{\sqrt{u^2+\tau^2} - |x|}\right)^{-k} & \textit{if } k < 0. \end{cases}$$

Proof. Assume first that $x \neq 0$. Let $a = x^2$, $b = -2\langle u, x\rangle$, and $c = u^2 + \tau^2$. Then

$$\Phi(x - u) = \left(x^2 - 2\langle u, x\rangle + u^2 + \tau^2\right)^{k/2} = f_k(1),$$

where f_k is the function that is defined in (2.1). Since $a, c > 0$, $b^2 \leq 4ac$, and $1 = |z| < \sqrt{c/a} = \sqrt{u^2 + \tau^2}/|x|$, Lemma 2.3 may be applied with $\nu = q$ to yield (8.1) and (8.3) when $x \neq 0$. The results for $x = 0$ follow by continuity.

It remains to show that $T_q\big(\Phi(\cdot - u)\big)$ is the Maclaurin polynomial of $\Phi(\cdot - u)$. Observe from (1.4) that

$$P_\ell^{(k)}(a, b, c) = P_\ell^{(k)}(x^2, -2\langle u, x\rangle, u^2 + \tau^2)$$

either is a homogeneous polynomial of exact degree ℓ in x or is trivial. Hence, by (8.3), $T_q\big(\Phi(\cdot - u)\big)$ is a polynomial of total degree q in x such that

$$\Big|\Phi(x-u) - T_q\big(\Phi(\cdot - u)\big)(x)\Big| = \mathcal{O}\big(|x|^{q+1}\big) \text{ as } |x| \to 0.$$

The result follows since the only such polynomial is the Maclaurin polynomial. □

9. Numerical results.

In this section we present numerical results generated by a fast evaluator for generalized multiquadrics based upon the mathematics of this paper.

The implementation is built around a hierarchical subdivision of an initial box containing all the centers using a binary tree of panels. Associated with a panel are the centers lying within it, a far field expansion, and a distance from the panel's midpoint at which the far field expansion approximates the influence of the panel to sufficient accuracy. Panels are divided, generating children if they contain more than a critical number of centers.

Pseudocode for recursive and nonrecursive evaluators appropriate for use with such a binary tree evaluation structure is sketched in [3, pp. 8–11]. Nominally the discussion there is limited to an $\mathbb{R}^1$, rather than an $\mathbb{R}^n$, setting but the generalization is immediate.

Tables 9.1 and 9.2 give times in seconds on an Intel Pentium III 700-based machine for matrix-vector products of various sizes and (generalized) multiquadrics in $\mathbb{R}^2$.

In these tables an entry of the form $d_0.d_1d_2d_3(e)$ with d_0, d_1, d_2, d_3 decimal digits represents the number $d_0.d_1d_2d_3 \times 10^e$. In the numerical experiments the centers are uniformly distributed on $[0,1]^2$, and the multiquadric parameter τ was taken as $1/\sqrt{N}$, where N is the number of centers. All the coefficients d_i were taken as 1 and the task was to evaluate the spline at the centers to an infinity norm relative accuracy of 10^{-6}. In the first table ϕ is the ordinary multiquadric $\phi(r) = \sqrt{r^2+\tau^2}$, while in the second it is the smoother function $\phi(r) = (r^2+\tau^2)^{3/2}$. The code used was structured as a general evaluator, and the symmetry inherent in this matrix-vector product test problem was not exploited. Tables 9.3 and 9.4 give the analogous results

Table 9.1
Timings of matrix-vector products for $\phi(r) = \sqrt{r^2+\tau^2}$ and $\mathbb{R}^2$.

N	Direct time	Algorithm time	Ratio
1,000	6.3(-2)	3.9(-2)	1.6
2,000	2.97(-1)	7.8(-2)	3.8
4,000	1.19(0)	2.03(-1)	5.86
8,000	4.75(0)	4.84(-1)	9.81
16,000	2.50(1)	9.84(-1)	25.4
32,000	1.10(2)	2.23(0)	49.3

Table 9.2
Timings of matrix-vector products for $\phi(r) = (r^2+\tau^2)^{3/2}$ and $\mathbb{R}^2$.

N	Direct time	Algorithm time	Ratio
1,000	6.3(-2)	3.9(-2)	1.6
2,000	2.97(-1)	7.8(-2)	3.8
4,000	1.19(0)	1.72(-1)	6.92
8,000	4.75(0)	3.75(-1)	12.7
16,000	2.97(1)	8.12(-1)	36.6
32,000	1.22(2)	1.76(0)	69.3

Table 9.3
Timings of matrix-vector products for $\phi(r) = \sqrt{r^2 + \tau^2}$ *and* $\mathbb{R}^3$.

N	Direct time	Algorithm time	Ratio
2,000	2.97(-1)	3.12(-1)	0.95
4,000	1.19(0)	7.81(-1)	1.52
8,000	7.62(0)	1.88(0)	4.05
16,000	3.40(1)	4.40(0)	7.73
32,000	1.36(2)	1.05(1)	13.0
64,000	5.44(2)	2.34(1)	23.2

Table 9.4
Timings of matrix-vector products for $\phi(r) = (r^2 + \tau^2)^{3/2}$ *and* $\mathbb{R}^3$.

N	Direct time	Algorithm time	Ratio
2,000	3.12(-1)	3.29(-1)	0.94
4,000	1.25(0)	8.43(-1)	1.48
8,000	8.64(0)	1.94(0)	4.45
16,000	3.80(1)	4.50(0)	8.44
32,000	1.48(2)	1.02(1)	14.5
64,000	6.16(2)	2.23(1)	27.6

for a three-dimensional code. Here the points are uniformly distributed in the unit cube $[0,1]^3$, and the multiquadric parameter τ is taken as $(1/N)^{1/3}$.

It can be seen from the tables that this algorithm can be substantially faster than direct evaluation. Thus the methods of this paper will allow application of multiquadric radial basis functions to much bigger problems than previously possible.

REFERENCES

[1] M. Abramowitz and I. A. Stegun, eds., *Handbook of Mathematical Functions*, Dover, New York, 1965.

[2] R. K. Beatson, J. B. Cherrie, and C. T. Mouat, *Fast fitting of radial basis functions: Methods based on preconditioned GMRES iteration*, Adv. Comput. Math., 11 (1999), pp. 253–270.

[3] R. K. Beatson and L. Greengard, *A short course on fast multipole methods*, in Wavelets, Multilevel Methods and Elliptic PDEs, M. Ainsworth, J. Levesley, W. Light, and M. Marletta, eds., Oxford University Press, Oxford, 1997, pp. 1–37.

[4] R. K. Beatson and W. A. Light, *Fast evaluation of radial basis functions: Methods for two-dimensional polyharmonic splines*, IMA J. Numer. Anal., 17 (1997), pp. 343–372.

[5] R. K. Beatson and G. N. Newsam, *Fast evaluation of radial basis functions* I, Comput. Math. Appl., 24 (1992), pp. 7–19.

[6] R. K. Beatson and G. N. Newsam, *Fast evaluation of radial basis functions: Moment-based methods*, SIAM J. Sci. Comput., 19 (1998), pp. 1428–1449.

[7] E. W. Cheney and W. A. Light, *A Course in Approximation Theory*, Brooks–Cole, Pacific Grove, CA, 1999.

[8] R. V. Churchill and J. W. Brown, *Complex Variables and Applications*, 4th ed., McGraw–Hill, New York, 1984.

[9] L. Greengard and V. Rokhlin, *A fast algorithm for particle simulations*, J. Comput. Phys, 73 (1987), pp. 325–348.

[10] R. L. Hardy, *Theory and applications of the multiquadric biharmonic method*, Comput. Math. Appl., 19 (1990), pp. 163–208.

[11] G. Szegö, *Orthogonal Polynomials*, revised ed., Amer. Math. Soc. Colloq. Publ. 23, American Mathematical Society, New York, 1959.

SIAM J. SCI. COMPUT.
Vol. 23, No. 5, pp. 1572–1592

AN ALGEBRAIC MULTILEVEL MULTIGRAPH ALGORITHM*

RANDOLPH E. BANK† AND R. KENT SMITH‡

Abstract. We describe an algebraic multilevel multigraph algorithm. Many of the multilevel components are generalizations of algorithms originally applied to general sparse Gaussian elimination. Indeed, general sparse Gaussian elimination with minimum degree ordering is a limiting case of our algorithm. Our goal is to develop a procedure which has the robustness and simplicity of use of sparse direct methods, yet offers the opportunity to obtain the optimal or near-optimal complexity typical of classical multigrid methods.

Key words. algebraic multigrid, incomplete LU factorization, multigraph methods

AMS subject classifications. 65M55, 65N55

PII. S1064827500381045

1. Introduction.

In this work, we develop a multilevel multigraph algorithm. Algebraic multigrid methods are currently a topic of intense research interest [17, 18, 20, 46, 12, 48, 38, 11, 44, 3, 4, 1, 2, 5, 16, 7, 29, 28, 27, 42, 41, 21]. An excellent recent survey is given in Wagner [49]. In many "real world" calculations, direct methods are still widely used [6]. The robustness of direct elimination methods and their simplicity of use often outweigh the apparent benefits of fast iterative solvers. Our goal here is to try to develop an iterative solver that can compete with sparse Gaussian elimination in terms of simplicity of use and robustness and to provide the potential of solving a wide range of linear systems more efficiently. While we are not yet satisfied that our method has achieved this goal, we believe that it is a reasonable first step. In particular, the method of general sparse Gaussian elimination with minimum degree ordering is a point in the parameter space of our method. This implies that in the worst case, our method defaults to this well-known and widely used method, among the most computationally efficient of general sparse direct methods [26]. In the best case, however, our method can exhibit the near optimal order complexity of the classical multigrid method.

Our plan is to take well studied, robust, and widely used procedures and data structures developed for sparse Gaussian elimination, generalize them as necessary, and use them as the basic components of our multilevel solver. The overall iteration follows the classical multigrid V-cycle in form, in contrast to the algebraic hierarchical basis multigraph algorithm developed in [11].

In this work we focus on the class of matrices which are structurally symmetric; that is, the pattern of nonzeros in the matrix is symmetric, although the numerical values of the matrix elements may render it nonsymmetric. Such structurally symmetric matrices arise in the discretizations of partial differential equations, say, by the finite element method. For certain problems, the matrices are symmetric and positive definite, but for others the linear systems are highly nonsymmetric and/or

*Received by the editors November 13, 2000; accepted for publication (in revised form) July 13, 2001; published electronically January 16, 2002.
http://www.siam.org/journals/sisc/23-5/38104.html

†Department of Mathematics, University of California at San Diego, La Jolla, CA 92093 (rbank@ucsd.edu). The work of this author was supported by the National Science Foundation under contract DMS-9706090.

‡Bell Laboratories, Lucent Technologies, Murray Hill, NJ 07974. Current address: Agere Systems, Murray Hill, NJ 07974 (kentsmith@agere.com).

indefinite. Thus in practice this represents a very broad class of behavior. While our main interest is in scalar elliptic equations, as in the finite element code *PLTMG* [8], our algorithms can formally be applied to any structurally symmetric, nonsingular, sparse matrix.

Sparse direct methods typically have two phases. In the first (initialization) phase, equations are ordered, and symbolic and numerical factorizations are computed. In the second (solution) phase, the solution of the linear system is computed using the factorization. Our procedure, as well as other algebraic multilevel methods, also breaks naturally into two phases. The initialization consists of ordering, incomplete symbolic and numeric factorizations, and the computation of the transfer matrices between levels. In the solution phase, the preconditioner computed in the initialization phase is used to compute solution using the preconditioned composite step conjugate gradient (CSCG) or the composite step biconjugate gradient (CSBCG) method [9].

Iterative solvers often have tuning parameters and switches which require a certain level of a priori knowledge or some empirical experimentation to set in any particular instance. Our solver is not immune to this, although we have tried to keep the number of such parameters to a minimum. In particular, in the initialization phase, there are only three such parameters:

- ϵ, the drop tolerance used in the incomplete factorization (called *dtol* in our code).
- $maxfil$, an integer which controls to overall fill-in (storage) allowed in a given incomplete factorization.
- *maxlvl*, an integer specifying the maximum number of levels.

(The case $\epsilon = 0$, $maxfil = N$, $maxlvl = 1$ corresponds to sparse Gaussian elimination.) In the solution phase, there are only two additional parameters:

- *tol*, the tolerance used in the convergence test.
- *maxcg*, an integer specifying the maximum number of iterations.

Within our code, all matrices are generally treated within a single, unified framework; e.g., symmetric positive definite, nonsymmetric, and indefinite problems generally do not have specialized options. Besides the control parameters mentioned above, all information about the matrix is generated from the sparsity pattern and the values of the nonzeros, as provided in our sparse matrix data structure, a variant of the data structure introduced in the Yale sparse matrix package [23, 10]. For certain block matrices, the user may optionally provide a small array containing information about the block structure.

This input limits the complexity of the code, as well as eliminates parameters which might be needed to further classify a given matrix. On the other hand, it seems clear that a specialized solver directed at a specific problem or class of problems, and making use of this additional knowledge, is likely to outperform our algorithm on that particular class of problems. Although we do not think our method is provably "best" for any particular problem, we believe its generality and robustness, coupled with reasonable computational efficiency, make it a valuable addition to our collection of sparse solvers.

The rest of this paper is organized as follows. In section 2, we provide a general description of our multilevel approach. In section 3, we define the sparse matrix data structures used in our code. Our incomplete factorization algorithm is a standard drop tolerance approach with a few modifications for the present application. These are described in section 4. Our ordering procedure is the minimum degree algorithm. Once again, our implementation is basically standard with several modifications to the input graph relevant to our application. These are described in section 5. In section 6, we describe the construction of the transfer matrices used in the construction of the coarse grid correction. Information about the block structure of the matrix, if any is provided, is used only in the coarsening procedure. This is described in section 7. Finally, in section 8, we give some numerical illustrations of our method on a variety of (partial differential equation) matrices.

2. Matrix formulation.

Let A be a large sparse, nonsingular $N \times N$ matrix. We assume that the sparsity pattern of A is symmetric, although the numerical values need not be. We will begin by describing the basic two-level method for solving

$$Ax = b. \tag{2.1}$$

Let B be an $N \times N$ nonsingular matrix, called the *smoother*, which gives rise to the basic iterative method used in the multilevel preconditioner. In our case, B is an approximate factorization of A, i.e.,

$$B = (L + D)D^{-1}(D + U) \approx P^t AP, \tag{2.2}$$

where L is (strict) lower triangular, U is (strict) upper triangular with the same sparsity pattern as L^t, D is diagonal, and P is a permutation matrix.

Given an initial guess x_0, m steps of the smoothing procedure produce iterates x_k, $1 \le k \le m$, given by

$$\begin{aligned} r_{k-1} &= P^t(b - Ax_{k-1}),\\ B\delta_{k-1} &= r_{k-1},\\ x_k &= x_{k-1} + P^t\delta_{k-1}. \end{aligned} \tag{2.3}$$

The second component of the two-level preconditioner is the *coarse grid correction.* Here we assume that the matrix A can be partitioned as

$$\hat{P}A\hat{P}^t = \begin{pmatrix} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{pmatrix}, \tag{2.4}$$

where the subscripts f and c denote *fine* and *coarse*, respectively. Similar to the smoother, the partition of A in fine and coarse blocks involves a permutation matrix $\hat{P}$. The $\hat{N} \times \hat{N}$ coarse grid matrix $\hat{A}$ is given by

$$\begin{aligned} \hat{A} &= \begin{pmatrix} V_{cf} & I_{cc} \end{pmatrix} \begin{pmatrix} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{pmatrix} \begin{pmatrix} W_{fc} \\ I_{cc} \end{pmatrix} \\ &= V_{cf}A_{ff}W_{fc} + V_{cf}A_{fc} + A_{cf}W_{fc} + A_{cc}. \end{aligned} \tag{2.5}$$

The matrices V_{cf} and W_{fc}^t are $\hat{N} \times (N - \hat{N})$ matrices with identical sparsity patterns; thus $\hat{A}$ has a symmetric sparsity pattern. If $A^t = A$, we require $V_{cf} = W_{fc}^t$, so $\hat{A}^t = \hat{A}$.

Let

$$\hat{V} = \begin{pmatrix} V_{cf} & I_{cc} \end{pmatrix} \hat{P}, \qquad \hat{W} = \hat{P}^t \begin{pmatrix} W_{fc} \\ I_{cc} \end{pmatrix}. \tag{2.6}$$

In standard multigrid terminology, the matrices $\hat{V}$ and $\hat{W}$ are called *restriction* and *prolongation*, respectively. Given an approximate solution x_m to (2.1), the coarse grid correction produces an iterate x_{m+1} as follows.

$$\begin{aligned} \hat{r} &= \hat{V}(b - Ax_m), \\ \hat{A}\hat{\delta} &= \hat{r}, \\ x_{m+1} &= x_m + \hat{W}\hat{\delta}. \end{aligned} \tag{2.7}$$

As is typical of multilevel methods, we define the *two-level preconditioner* M implicitly in terms of the smoother and coarse grid correction. A single cycle takes an initial guess x_0 to a final guess x_{2m+1} as follows:

Two-Level Preconditioner

(i) x_k for $1 \le k \le m$ are defined using (2.3).

(ii) x_{m+1} is defined using (2.7).

(iii) x_k for $m+2 \le k \le 2m+1$ are defined using (2.3).

The generalization from two-level to multilevel consists of applying recursion to the solution of the equation $\hat{A}\hat{\delta} = \hat{r}$ in (2.7). Let ℓ denote the number of levels in the recursion. Let $\hat{M} \equiv \hat{M}(\ell)$ denote the preconditioner for $\hat{A}$; if $\ell = 2$, then $\hat{M} = \hat{A}$. Then (2.7) is generalized to

$$\begin{aligned} \hat{r} &= \hat{V}(b - Ax_m), \\ \hat{M}\hat{\delta} &= \hat{r}, \\ x_{m+1} &= x_m + \hat{W}\hat{\delta}. \end{aligned} \tag{2.8}$$

The general ℓ level preconditioner M is then defined as follows:

***ℓ*-Level Preconditioner**

(i) if $\ell = 1$, $M = A$; i.e., solve (2.1) directly.

(ii) if $\ell > 1$, then, starting from initial guess x_0, compute x_{2m+1} using (iii)–(v):

(iii) x_k for $1 \le k \le m$ are defined using (2.3).

(iv) x_{m+1} is defined by (2.8), using $p = 1$ or $p = 2$ iterations of the $\ell - 1$ level scheme for $\hat{A}\hat{\delta} = \hat{r}$ to define $\hat{M}$, and with initial guess $\hat{\delta}_0 = 0$.

(v) x_k for $m+2 \le k \le 2m+1$ are defined using (2.3).

The case $p = 1$ corresponds to the symmetric *V-cycle*, while the case $p = 2$ corresponds to the symmetric *W-cycle*. We note that there are other variants of both the V-cycle and the W-cycle, as well as other types of multilevel cycling strategies [30]. However, in this work (and in our code) we restrict attention to just the symmetric V-cycle with $m = 1$ presmoothing and postsmoothing iterations.

For the coarse mesh solution ($\ell = 1$), our procedure is somewhat nontraditional. Instead of a direct solution of (2.1), we compute an approximate solution using one smoothing iteration. We illustrate the practical consequences of this decision in section 8.

If A is symmetric, then so is M, and the ℓ-level preconditioner could be used as a preconditioner for a symmetric Krylov space method. If A is also positive definite, so is M, and the standard conjugate gradient method could be used; otherwise the CSCG method [9], SYMLQ [43], or a similar method could be used. In the nonsymmetric case, the ℓ-level preconditioner could be used in conjunction with the CSBCG method [9], GMRES [22], or a similar method.

To complete the definition of the method, we must provide algorithms to

- compute the permutation matrix P in (2.2);
- compute the incomplete factorization matrix B in (2.2);
- compute the fine-coarse partitioning $(\hat{P})$ in (2.4);

- compute the sparsity patterns and numerical values in the prolongation and restriction matrices in (2.6).

3. Data structures. Let A be an $N \times N$ matrix with elements A_{ij} and a symmetric sparsity structure; that is, both A_{ij} and A_{ji} are treated as nonzero elements (i.e. stored and processed) if $|A_{ij}| + |A_{ji}| > 0$. All diagonal entries A_{ii} are treated as nonzero regardless of their numerical values.

Our data structure is a modified and generalized version of the data structure introduced in the (symmetric) Yale sparse matrix package [23]. It is a rowwise version of the data structure described in [10]. In our scheme, the nonzero entries of A are stored in a linear array a and accessed through an integer array ja. Let η_i be the number of nonzeros in the strict upper triangular part of row i and set $\eta = \sum_{i=1}^{N} \eta_i$. The array ja is of length $N+1+\eta$, and the array a is of length $N+1+\eta$ if $A^t = A$. If $A^t \neq A$, then the array a is of length $N+1+2\eta$. The entries of $ja(i)$, $1 \leq i \leq N+1$, are pointers defined as follows:

$$\begin{aligned} ja(1) &= N+2, \\ ja(i+1) &= ja(i) + \eta_i, \quad 1 \leq i \leq N. \end{aligned}$$

The locations $ja(i)$ to $ja(i+1)-1$ contain the η_i column indices corresponding to the row i in the strictly upper triangular matrix.

In a similar manner, the array a is defined as follows:

$$\begin{aligned} a(i) &= A_{ii}, \quad 1 \leq i \leq N, \\ a(N+1) & \quad \text{is arbitrary}, \\ a(k) &= A_{ij}, \quad 1 \leq i \leq N, \quad j = ja(k), \quad ja(i) \leq k \leq ja(i+1)-1. \end{aligned}$$

If $A^t \neq A$, then

$$a(k+\eta) = A_{ji}, \quad 1 \leq i \leq N, \quad j = ja(k), \quad ja(i) \leq k \leq ja(i+1)-1.$$

In words, the diagonal is stored first, followed by the strict upper triangle stored rowwise. If $A^t \neq A$, then this is followed by the strict lower triangle stored columnwise. Since A is structurally symmetric, the column indexes for the upper triangle are identical to the row indexes for the lower triangle, and hence they need not be duplicated in storage.

As an example, let

$$A = \begin{pmatrix} A_{11} & A_{12} & A_{13} & 0 & 0 \\ A_{21} & A_{22} & 0 & A_{24} & 0 \\ A_{31} & 0 & A_{33} & A_{34} & A_{35} \\ 0 & A_{42} & A_{43} & A_{44} & 0 \\ 0 & 0 & A_{53} & 0 & A_{55} \end{pmatrix}.$$

Then

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
ja	7	9	10	12	12	12	2	3	4	4	5					
a	A_{11}	A_{22}	A_{33}	A_{44}	A_{55}		A_{12}	A_{13}	A_{24}	A_{34}	A_{35}	A_{21}	A_{31}	A_{42}	A_{43}	A_{53}
	Diagonal						Upper triangle					Lower triangle				

Although the YSMP data structure was originally devised for sparse direct methods based on Gaussian elimination, it is also quite natural for iterative methods based on incomplete triangular decomposition. Because we assume that A has a symmetric sparsity structure, for many matrix calculations a single indirect address computation in ja can be used to process both a lower and a upper triangular element in A. For example, the following procedure computes $y = Ax$:

```
procedure mult(N, ja, a, x, y)
    $lmtx \leftarrow ja(N+1) - ja(1)$
    $umtx \leftarrow 0$
    for $i \leftarrow 1$ to $N$
        $y(i) \leftarrow a(i)x(i)$
    end for
    for $i \leftarrow 1$ to $N$
        for $k \leftarrow ja(i)$ to $ja(i+1) - 1$
            $j \leftarrow ja(k)$
            $y(i) \leftarrow y(i) + a(k + umtx)x(j)$
            $y(j) \leftarrow y(j) + a(k + lmtx)x(i)$
        end for
    end for
end mult
```

For symmetric matrices, set $lmtx \leftarrow 0$, $umtx \leftarrow 0$. Also, $y = A^t\, x$ may be readily computed by setting $lmtx \leftarrow 0$, $umtx \leftarrow ja(N+1) - ja(1)$.

The data structure for storing $B = (L+D)D^{-1}(D+U)$ is quite analogous to that for A. It consists of two arrays, ju and u, corresponding to ja and a, respectively. The first $N+1$ entries of ju are pointers as in ja, while entries $ju(i)$ to $ju(i+1)-1$ contain column indices of the nonzeros of row i in of U. In the u array, the diagonal entries of D are stored in the first N entries. Entry $N+1$ is arbitrary. Next, the nonzero entries of U are stored in correspondence to the column indices in ju. If $L^t \neq U$, the nonzero entries of L follow, stored columnwise.

The data structure we use for the $N \times \hat{N}$ matrix $\hat{W}$ and the $\hat{N} \times\ N$ matrix $\hat{V}$ are similar. It consists of an integer array jv and a real array v. The nonzero entries of $\hat{W}$ are stored rowwise, including the rows of the block I_{cc}. As usual, the first $N+1$ entries of jv are pointers; entries $jv(i)$ to $jv(i+1)-1$ contain column indices for row i of $\hat{W}$. In the v array, the nonzero entries of $\hat{W}$ are stored rowwise in correspondence with jv but shifted by $N+1$ since there is no diagonal part. If $\hat{V}^t \neq \hat{W}$, this is followed by the nonzeros of $\hat{V}$, stored columnwise.

4. ILU factorization.

Our incomplete $(L+D)D^{-1}(D+U)$ factorization is similar to the row elimination scheme developed for the symmetric YSMP codes [23, 26]. For simplicity, we begin by discussing a complete factorization and then describe the modifications necessary for the incomplete factorization. Without loss of generality, assume that the permutation matrix $P = I$, so that $A = (L+D)D^{-1}(D+U)$.

After k steps of elimination, we have the block factorization

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} D_{11} + L_{11} & 0 \\ L_{21} & I \end{pmatrix} \begin{pmatrix} D_{11}^{-1} & 0 \\ 0 & S \end{pmatrix} \begin{pmatrix} D_{11} + U_{11} & U_{12} \\ 0 & I \end{pmatrix}, \tag{4.1}$$

where A_{11} is $k \times k$ and A_{22} is $N - k \times N - k$. We assume that at this stage, all the blocks on the right-hand side of (4.1) have been computed except for the Schur complement S, given by

$$S = A_{22} - L_{21} D_{11}^{-1} U_{12}. \tag{4.2}$$

Our goal for step $k + 1$ is to compute the first row and column of S, given by

$$\begin{aligned} Se_1 &= A_{22}e_1 - L_{21}(D_{11}^{-1}U_{12}e_1), \\ S^t e_1 &= A_{22}^t e_1 - U_{12}^t (D_{11}^{-1} L_{21}^t e_1). \end{aligned} \tag{4.3}$$

Because A and $(L + D)D^{-1}(D + U)$ have symmetric sparsity patterns, and our data structures take advantage of this symmetry, it is clear that the algorithms for computing Se_1 and $S^t e_1$ are the same and in practice differ only in the assignments of shifts for the u and a arrays, analogous to *lmtx* and *umtx* in procedure mult. Thus we will focus on the computation of just Se_1. At this point, we also assume that the array ju has been computed in a so-called *symbolic factorization* step.

The major substeps are as follows:

1. Copy the first column of A_{22} (stored in the data structures ja and a) into an expanded work vector z of size N.
2. Find the *multipliers* given by nonzeros of $D_{11}^{-1}U_{12}e_1$.
3. For each multiplier $\gamma = e_k^t D_{11}^{-1} U_{12} e_1$, update z using column k of L_{21} (i.e., $\gamma L_{21} e_k$).
4. Copy the nonzeros in z into the data structures ju and u.

In step 1, we need to know the nonzeros of the first column of A_{22}, which is precisely the information easily accessible in the ja and a data structures. In step 3, we need to know the nonzeros in columns of L_{21}, which again is precisely the information easily available in our data structure. In step 4, we copy a column of information into the lower triangular portion of the ju and u data structures. Indeed, the only difficult aspect of the algorithm is step 2, in which we need to know the sparsity structure of the first *column* of U_{12}, information that is *not* readily available in the data structure. This is handled in a standard fashion using a dynamic linked list structure and will not be discussed in detail here.

To generalize this to the incomplete factorization case, we first observe that the ju array can be computed concurrently with the numeric factorization simply by creating a list of the entries of the expanded array z that are updated in step 3. Next, we note that one may choose which nonzero entries from z to include in the factorization by choosing which entries to copy to the ju and u data structures in step 4. We do this through a standard approach using a drop tolerance ϵ. In particular, we neglect a pair of off-diagonal elements if

$$\max |L_{ij}|, |U_{ji}| \le \epsilon \sqrt{|D_{jj} A_{ii}|}, \tag{4.4}$$

$j = k + 1$ and $i > j$. Note D_{ii} has not yet been computed. It is well known that the fill-in generated through the application of a criterion such as (4.4) is a highly nonlinear and matrix dependent function of ϵ. This is especially problematic in the present context, since control of the fill-in is necessary in order to control the work per iteration in the multilevel iteration.

Several authors have explored possibilities of controlling the maximum number of fill-in elements allowed in each row of the incomplete decomposition [35, 47, 31]. However, for many cases of interest, and in particular for matrices arising from discretizations of partial differential equations ordered by the minimum degree algorithm, most of the fill-in in a complete factorization occurs in the later stages, even if all the rows initially have about the same number of nonzeros. Thus while it seems advisable to try to control the total fill-in, one should adaptively decide how to allocate the fill-in among the rows of the matrix. In our algorithm, in addition to the drop tolerance ϵ, the user provides a parameter $maxfil$, which specifies that the total number of nonzeros in U is not larger than $maxfil \cdot N$.

Our overall strategy is to compute the incomplete decomposition using the given drop tolerance. If it fails to meet the given storage bound, we increase the drop tolerance and begin a new incomplete factorization. We continue in this fashion until we complete a factorization within the given storage bound. Of course, such repeated factorizations are computationally expensive, so we developed some heuristics which allow us to predict a drop tolerance which will satisfy the storage bound.

As the factorization is computed, we make a histogram of the approximate sizes of all elements that exceed the drop tolerance and are accepted for the factorization. Let m denote the number of bins in the histogram; $m = 400$ in our code. Then for each pair of accepted off-diagonal elements, we find the largest $k \in [1, m]$ such that

$$\rho^{k-1} \leq \frac{\max |L_{ij}|, |U_{ji}|}{\epsilon\sqrt{|D_{jj}A_{ii}|}}. \tag{4.5}$$

Here $\rho > 1$ ($\rho = 10^{4/m}$ in our code). The histogram is realized as an integer array h of size m, where h_ℓ is the number of accepted elements that exceeded the drop tolerance by factors between $\rho^{\ell-1}$ and ρ^ℓ for $1 \leq \ell \leq m-1$; h_m contains the number of accepted elements exceeding the drop tolerance by ρ^{m-1}. If the factorization reaches the storage bound, we continue the factorization but allow no further fill-in. However, we continue to compute the histogram based on (4.5), profiling the elements we would have accepted had space been available. Then using the histogram, we predict a new value of ϵ such that the total number of elements accepted for U is no larger than $maxfil \cdot N/\theta$. Such a prediction of course cannot be guaranteed, since the sizes and numbers of fill-in elements depend in a complicated fashion on the specific history of the incomplete factorization process; indeed, the histogram cannot even completely profile the remainder of the factorization with the existing drop tolerance, since elements that would have been accepted could introduce additional fill-in at later stages of the calculation as well as influence the sizes of elements computed at later stages of the factorization. In our implementation, the factor θ varies between $\theta = 1.01$ and $\theta = 1.4$, depending on how severely the storage bound was exceeded. Its purpose is to introduce some conservative bias into the prediction with the goal that the actual fill-in accepted should not exceed $maxfil \cdot N$.

Finally, we note that there is no comprehensive theory regarding the stability of incomplete triangular decompositions. For certain classes of matrices (e.g., M-matrices and H-matrices), the existence of certain incomplete factorizations has been proved [39, 25, 24, 40, 51]. However, in the general case, with potentially indefinite and/or highly nonsymmetric matrices, one must contend in a practical way with the possibility of failure or near failure of the factorization. A common approach is to add a diagonal matrix, often a multiple of the identity, to A and compute an incomplete factorization of the shifted matrix. One might also try to incorporate some form of diagonal pivoting; partial or complete pivoting could potentially destroy the symmetric sparsity pattern of the matrix. However, any sort of pivoting greatly increases the complexity of the implementation, since the simple but essentially static data structures ja, a, ju, and u are not appropriate for such an environment.

Our philosophy here is to simply accept occasional failures and continue with the factorization. Our ordering procedure contains some heuristics directed towards avoiding or at least minimizing the possibility of failures. And when they do occur, failures often corrupt only a low dimensional subspace, so a Krylov space method such as conjugate gradients can compensate for such corruption with only a few extra iterations. In our implementation, a failure is revealed by some diagonal entries in D becoming close to zero. Off-diagonal elements L_{ji} and U_{ij} are multiplied by D_{ii}^{-1}, and the solution of $(L+D)D^{-1}(D+U)x = b$ also involves multiplication by D_{ii}^{-1}. For purposes of calculating the factorization and solution, the value of D_{ii}^{-1} is modified near zero as follows:

$$D_{ii}^{-1} = \begin{cases} 1/D_{ii} & \text{for } |D_{ii}| > \alpha, \\ D_{ii}/\alpha^2 & \text{for } |D_{ii}| \le \alpha. \end{cases} \tag{4.6}$$

Here α is a small constant; in our implementation, $\alpha = \mu\|A\|$, where μ is the machine epsilon. Although many failures could render the preconditioner well-defined but essentially useless, in practice we have noted that D_{ii}^{-1} is rarely modified for the large class of finite element matrices which are the main target of our procedure.

5. Ordering.

To compute the permutation matrix P in (2.2), we use the well-known minimum degree algorithm [45, 26]. Intuitively, if one is computing an incomplete factorization, an ordering which tends to minimize the fill-in in a complete factorization should tend to minimize the error

$$E = P^tAP - (L+D)D^{-1}(D+U).$$

For particular classes of matrices, specialized ordering schemes have been developed [34, 15, 37, 36]. For example, for matrices arising from convection dominated problems, ordering along the flow direction has been used with great success. However, in this general setting, we prefer to use just one strategy for all matrices. This reduces the complexity of the implementation and avoids the problem of developing heuristics to decide among various ordering possibilities. We remark that for convection dominated problems, minimum degree orderings perform comparably well to the specialized ones, provided some (modest) fill-in is allowed in the incomplete factorization. For us, this seems to be a reasonable compromise.

Our minimum degree ordering is a standard implementation, using the quotient graph model [26] and other standard enhancements. A description of the *graph* of the matrix is the main required input. Without going into detail, this is essentially a small variant of the basic ja data structure used to store the matrix A. We will denote this modified data structure as jc. Instead of storing only column indices for the strict upper triangle as in ja, entries $jc(i)$ to $jc(i+1)-1$ of the jc data structure contain column indices for *all* off-diagonal entries for row i of the matrix A.

We have implemented two small enhancements to the minimum degree ordering; as a practical matter, both involve changes to the input graph data structure jc that is provided to the minimum degree code. First, we have implemented a drop tolerance similar to that used in the the factorization. In particular the edge in the graph corresponding to off-diagonal entries A_{ij} and A_{ji} is not included in the jc data structure if

$$\max |A_{ij}|, |A_{ji}| \le \epsilon\sqrt{|A_{jj}A_{ii}|}. \tag{5.1}$$

This excludes many entries which are likely to be dropped in the subsequent incomplete factorization and hopefully will result in an ordering that tends to minimize the fill-in created by the edges that are kept.

The second modification involves some modest a priori diagonal pivoting designed to minimize the number failures (near zero diagonal elements) in the subsequent factorization. We first remark that pivoting or other procedures based on the values of the matrix elements (which can be viewed as weights on graph edges and nodes) would destroy many of the enhancements which allow the minimum degree algorithm to run in almost linear time. Our modification is best explained in the context of a simple 2×2 example. Let

$$A = \begin{pmatrix} 0 & c \\ b & a \end{pmatrix}$$

with $a, b, c \neq 0$. Clearly, A is nonsingular, but the complete triangular factorization of A does not exist. However,

$$P^t A P = \begin{pmatrix} a & b \\ c & 0 \end{pmatrix} = \begin{pmatrix} a & 0 \\ c & -bc/a \end{pmatrix} \begin{pmatrix} 1/a & 0 \\ 0 & -a/bc \end{pmatrix} \begin{pmatrix} a & b \\ 0 & -bc/a \end{pmatrix}. \tag{5.2}$$

Now suppose that $A_{ii} \approx 0$, $A_{jj}, A_{ij}, A_{ji} \neq 0$. Then these four elements form a submatrix of the form described above, and it seems an incomplete factorization of A is less likely to fail if the P is chosen such that vertex j is ordered before vertex i. This is done as follows: for each i such that $A_{ii} \approx 0$, we determine a corresponding j such that $A_{jj}, A_{ij}, A_{ji} \neq 0$; if there is more than one choice, we choose the one for which $|A_{ij}A_{ji}/A_{jj}|$ is maximized. To ensure that vertex i is ordered after vertex j, we replace the sparsity pattern for the off-diagonal entries for row (column) i with the union of those for rows (columns) i and j. If we denote the set of column indices for row i in the jc array as $adj(i)$, then

$$adj(j) \cup \{j\} \subseteq adj(i) \cup \{i\}. \tag{5.3}$$

Although the sets $adj(i)$ and $adj(j)$ are modified at various stages, it is well known that (5.3) is maintained throughout the minimum degree ordering process [26], so that at every step of the ordering process $deg(j) \leq deg(i)$, where $deg(i)$ is the degree of vertex i. As long as $deg(j) < deg(i)$, vertex j will be ordered before vertex i by the minimum degree algorithm. On the other hand, if $deg(i) = deg(j)$ at some stage of the ordering process, it remains so thereafter, and (5.3) becomes

$$adj(j) \cup \{j\} = adj(i) \cup \{i\}. \tag{5.4}$$

In words, i and j become so-called *equivalent vertices* and will be eliminated at the same time by the minimum degree algorithm (see [26] for details). Since the minimum degree algorithm sees these vertices as equivalent, they will be ordered in an arbitrary fashion when eliminated from the graph. Thus, as a simple postprocessing step, we must scan the ordering provided by the minimum degree algorithm and exchange the order of rows i and j if i was ordered first. Any such exchanges result in a new minimum degree ordering which is completely equivalent, in terms of fill-in, to the the original.

For many types of finite element matrices (e.g., the indefinite matrices arising from Helmholtz equations), this a priori scheme is useless because none of the diagonal entries of A is close to zero. However, this type of problem is likely to produce only isolated small diagonal entries in the factorization process, if it produces any at all. On the other hand, other classes of finite element matrices, notably those arising in from mixed methods, Stokes equations, and other saddle-point-like formulations, have many diagonal entries that are small or zero. In such cases, the a priori diagonal pivoting strategy can make a substantial difference and greatly reduce the numbers of failures in the incomplete triangular decomposition.

6. Computing the transfer matrices.

There are three major tasks in computing the prolongation and restriction matrices $\hat{V}$ and $\hat{W}$ of (2.6). First, one must determine the sparsity structure of these matrices; this involves choosing which unknowns are *coarse* and which are *fine*. This reduces to determining the permutation matrix $\hat{P}$ of (2.4). Second, one must determine how coarse and fine unknowns are related, the so-called *parent-child relations* [49]. This involves computing the sparsity patterns for the matrices V_{cf} and W_{fc}. Third, one must compute the numerical values for these matrices, the so-called *interpolation coefficients* [50].

There are many existing algorithms for coarsening graphs. For matrices arising from discretizations of partial differential equations, often the sparsity of the matrix A is related in some way to the underlying grid, and the problem of coarsening the *graph* of the matrix A can be formulated in terms of coarsening the *grid.* Some examples are given in [14, 13, 17, 18, 46, 12, 49]. In this case, one has the geometry of the grid to serve as an aid in developing and analyzing the coarsening procedure. There are also more general graph coarsening algorithms [32, 33, 19], often used to partition problems for parallel computation. Here our coarsening scheme is based upon another well-known sparse matrix ordering technique, the reverse Cuthill–McKee algorithm. This ordering tends to yield reordered matrices with minimal bandwidth and is widely used with generalized band elimination algorithms [26]. We now assume that the graph has been ordered in this fashion and that a *jc* data structure representing the graph in this ordering is available. Our coarsening procedure is just a simple postprocessing step of the basic ordering routine, in which the N vertices of graph are marked as $COARSE$ or $FINE$.

```
procedure coarsen(N, jc, type)
    for $i \leftarrow 1$ to $N$
        $type(i) \leftarrow UNDEFINED$
    end for
    for $i \leftarrow 1$ to $N$
        if $type(i) = UNDEFINED$, then
            $type(i) \leftarrow COARSE$
            for $j \leftarrow jc(i)$ to $jc(i+1) - 1$
                $type(jc(j)) \leftarrow FINE$
            end for
        end if
    end for
end coarsen
```

This postprocessing step, coupled with the the reverse Cuthill–McKee algorithm, is quite similar to a greedy algorithm for computing maximal independent sets using breadth-first search. Under this procedure, all coarse vertices are surrounded only by fine vertices. This implies that the matrix A_{cc} in (2.4) is a diagonal matrix. For the sparsity patterns of matrices arising from discretizations of scalar partial differential equations in two space dimensions, the number of coarse unknowns $\hat{N}$ is typically on the order of $N/4$ to $N/5$. Matrices with more nonzeros per row tend to have smaller values of $\hat{N}$. To define the parents of a coarse vertex, we take all the connections of the vertex to other fine vertices; that is, the sparsity structure of V_{cf} in (2.5) is the same as that of the block A_{cf}.

In our present code, we pick V_{cf} and W_{fc} according to the formulae

$$\begin{aligned} W_{fc} &= -R_{ff} D_{ff}^{-1} A_{fc}, \\ V_{cf} &= -A_{cf} D_{ff}^{-1} \tilde{R}_{ff}. \end{aligned} \tag{6.1}$$

Here D_{ff} is a diagonal matrix with diagonal entries equal to those of A_{ff}. In this sense, the nonzero entries in V_{cf} and W_{fc} are chosen as multipliers in Gaussian elimination. The nonnegative diagonal matrices R_{ff} and $\tilde{R}_{ff}$ are chosen such that nonzero rows of W_{fc} and columns of V_{cf}, respectively, have unit norms in ℓ_1.

Finally, the coarsened matrix $\hat{A}$ of (2.5) is "sparsified" using the drop tolerance and a criterion like (5.1) to remove small off-diagonal elements. Empirically, applying a drop tolerance to $\hat{A}$ at the end of the coarsening procedure has proved more efficient, and more effective, than trying to independently sparsify its constituent matrices. If the number of off-diagonal elements in the upper triangle exceeds $maxfil \cdot \hat{N}$, the drop tolerance is modified in a fashion similar to the incomplete factorization. The off-diagonal elements are profiled by a procedure similar to that for the incomplete factorization, but in this case the resulting histogram is exact. Based on this histogram, a new drop tolerance is computed, and (5.1) is applied to produce a coarsened matrix satisfying the storage bound.

7. Block matrices.

Our algorithm provides a simple but limited functionality for handling block matrices. Suppose that the $N \times N$ matrix A has the $K \times K$ block structure

$$A = \begin{pmatrix} A_{11} & \dots & A_{1K} \\ \vdots & \ddots & \vdots \\ A_{K1} & \dots & A_{KK} \end{pmatrix}, \tag{7.1}$$

where subscripts for A_{ij} are block indices and the diagonal blocks A_{jj} are square matrices. Suppose A_{jj} is of order N_j; then $\sum_{j=1}^{K} N_j = N$.

The matrix A is stored in the usual ja and a data structures as described in section 3 with no reference to the block structure. A small additional integer array ib of size $K+1$ is used to define the block boundaries as follows:

$$\begin{aligned} ib(1) &= 1, \\ ib(j+1) &= ib(j) + N_j, \quad 1 \le j \le K. \end{aligned}$$

In words, integers in the range $ib(j)$ to $ib(j+1)-1$, inclusive, comprise the index set associated with block A_{jj}. Note that $ib(K+1) = N+1$.

This block information plays a role only in the coarsening algorithm. First, the reverse Cuthill–McKee algorithm described in section 6 is applied to the block diagonal matrix

$$\bar{A} = \begin{pmatrix} A_{11} & & \\ & \ddots & \\ & & A_{KK} \end{pmatrix} \tag{7.2}$$

rather than A. As a practical matter, this involves discarding graph edges connecting vertices of different blocks in the construction of the graph array jc used as input. Such edges are straightforward to determine from the information provided in the ib array. The coarsening algorithm applied to the graph of $\bar{A}$ produces output equivalent to the application of the procedure independently to each diagonal block of $\bar{A}$. As a consequence, the restriction and prolongation matrices automatically inherit the block structure A. In particular,

$$\hat{V} = \begin{pmatrix} \hat{V}_{11} & & \\ & \ddots & \\ & & \hat{V}_{KK} \end{pmatrix} \quad \text{and} \quad \hat{W} = \begin{pmatrix} \hat{W}_{11} & & \\ & \ddots & \\ & & \hat{W}_{KK} \end{pmatrix}, \tag{7.3}$$

where $\hat{V}_{jj}$ and $\hat{W}_{jj}$ are are rectangular matrices ($\hat{N}_j \times N_j$ and $N_j \times \hat{N}_j$, respectively), having the structure of (2.6) that would have resulted from the application of the algorithm independently to A_{jj}. However, like the matrix A, $\hat{V}$ and $\hat{W}$ are stored in the standard jv and v data structures described in section 3 without reference to their block structures.

The complete matrix A is used in the construction of the coarsened matrix $\hat{A}$ of (2.5). However, because of (7.1) and (7.3)

$$\hat{A} = \hat{V} A \hat{W} = \begin{pmatrix} \hat{A}_{11} & \dots & \hat{A}_{1K} \\ \vdots & \ddots & \vdots \\ \hat{A}_{K1} & \dots & \hat{A}_{KK} \end{pmatrix},$$

so $\hat{A}$ also automatically inherits the $K \times K$ block structure of A. It is not necessary for the procedure forming $\hat{A}$ to have any knowledge of its block structure, as this block structure can be computed a priori by the graph coarsening procedure. Like A, $\hat{A}$ is stored in standard ja and a data structures without reference to its block structure. Since the blocks of A have arbitrary order, and are essentially coarsened independently, it is likely that eventually some of the $\hat{N}_j = 0$. That is, certain blocks may cease to exist on coarse levels. Since the block information is used only to discard certain edges in the construction of the graph array jc, "0×0" diagonal blocks present no difficulty.

8. Numerical experiments. In this section, we present a few numerical illustrations. In our first sequence of experiments, we consider several matrices loosely based on the classical case of 5-point centered finite difference approximations to $-\Delta u$ on a uniform square mesh. Dirichlet boundary conditions are imposed. This leads to the $n \times n$ block tridiagonal system

$$A = \begin{pmatrix} T & -I & & & \\ -I & T & -I & & \\ & \ddots & \ddots & \ddots & \\ & & -I & T & -I \\ & & & -I & T \end{pmatrix}$$

with T the $n \times n$ tridiagonal matrix

$$T = \begin{pmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 4 & -1 \\ & & & -1 & 4 \end{pmatrix}.$$

This is a simple test problem easily solved by standard multigrid methods. In contrast to this example we also consider the block tridiagonal system

$$\bar{A} = 8I - A.$$

Both A and $\bar{A}$ have the same eigenvectors and the same eigenvalues, although the association of eigenvectors and eigenvalues are reversed in the case of $\bar{A}$. That is, the so-called *smooth* eigenvectors are associated with large eigenvalues, while *rough* eigenvectors are associated with smaller eigenvalues. Although $\bar{A}$ does not arise naturally in the context of numerical discretizations of partial differential equations, it is of interest because it defies much of the conventional wisdom for multigrid methods.

Third, we consider block 3×3 systems of the form

$$S = \begin{pmatrix} A & 0 & C_x \\ 0 & A & C_y \\ C_x^t & C_y^t & -D \end{pmatrix},$$

where A is the discrete Laplacian and D is a symmetric positive definite "stabilization" matrix with a sparsity pattern similar to A. However, the nonzeros in D are of size $O(h^2)$, compared to size $O(1)$ nonzero elements in A. C_x and C_y also have sparsity patterns similar to that of A, but these matrices are nonsymmetric and their nonzero entries are of size $O(h)$. Such matrices arise in stabilized discretizations of the Stokes equations. One third of the eigenvalues of S are negative, so S is quite indefinite. In addition to the *ja* and *a* arrays, for the matrix S we also provided an *ib* array as described in section 7 to define its 3×3 block structure. We emphasize again that this block information is used only in the computation of the graph input to the coarsening procedure and is not involved in any aspect of the incomplete factorization smoothing procedure. With many small diagonal elements, this class of matrices provides a good test of the a priori pivoting strategy used in conjunction with the minimum degree ordering.

In Table 8.1, *Levels* refers to the number of levels used in the calculation. In our implementation the parameter *maxlvl*, which limits the number of levels allowed, was set sufficiently large that it had no effect on the computation. The drop tolerance was set to $\epsilon = 10^{-2}$ for all matrices. The fill-in control parameter $maxfil$ was set sufficiently large that it had no effect on the computation. The initial guess for all problems was $x_0 = 0$.

In Table 8.1, the parameter *Digits* refers to

$$Digits = -\log \frac{\|r_k\|}{\|r_0\|}. \tag{8.1}$$

In these experiments, we asked for six digits of accuracy. The column labeled *Cycles* indicates the number of multigrid cycles (accelerated by CSCG) that were used to

Table 8.1
Performance comparison.

n	N	Levels	Digits	Cycles	Init.	Solve
Discrete Laplacian A, $\epsilon = 10^{-2}$						
10	100	6	6.3	2	4.4e-3	1.2e-3
20	400	7	8.2	3	2.1e-2	6.9e-3
40	1600	8	8.6	4	9.4e-2	3.7e-2
80	6400	8	6.6	4	4.1e-1	2.0e-1
160	25600	9	6.9	5	1.9e 0	1.2e 0
320	102400	11	7.1	6	9.6e 0	7.4e 0
$\bar{A} = 8I - A$, $\epsilon = 10^{-2}$						
10	100	6	8.8	2	4.2e-3	1.2e-3
20	400	7	6.3	2	1.9e-2	5.0e-3
40	1600	8	8.1	3	9.2e-2	3.0e-2
80	6400	8	7.2	3	4.0e-1	1.6e-1
160	25600	9	6.8	3	1.9e 0	7.9e-1
320	102400	11	6.6	3	9.5e 0	4.2e 0
Stokes matrix S, $\epsilon = 10^{-2}$						
10	300	6	7.4	2	3.0e-2	5.3e-3
20	1200	7	8.2	3	2.3e-1	4.5e-2
40	4800	8	7.9	5	1.5e 0	5.1e-1
80	19200	9	6.5	5	8.1e 0	2.6e 0
160	76800	9	6.0	8	41.4e 0	20.6e 0

achieve the indicated number of digits. Finally, the last two columns, labeled *Init.* and *Solve*, record the CPU time, measured in seconds, for the initialization and solution phases of the algorithm, respectively. Initialization includes all the orderings, incomplete factorizations, and computation of transfer matrices used in the multigraph preconditioner. Solution includes the time to solve (2.1) to at least six digits given the preconditioner. These experiments were run on an SGI Octane R10000 250mhz, using double precision arithmetic and the f90 compiler.

In analyzing these results, it is clear that our procedure does reasonably well on all three classes of matrices. Although it appears that the rate of convergence is not independent of N, it seems apparent that the work is growing no faster than logarithmically. CPU times for larger vales of N are affected by cache performance as well as the slightly larger number of cycles.

For the highly indefinite Stokes matrices S, it is important to also note the robustness, that the procedure solved all of the problems. With more nonzeros per row on average, the incomplete factorization was more expensive to compute than for the other cases. This is reflected in relatively larger initialization and solve times.

In our next experiment, we illustrate the effect of the parameters $maxlvl$ and ϵ. For the matrix A with $N = 160000$, we solved the problem for $\epsilon = 10^{-k}$, $1 \leq k \leq 3$, and $1 \leq maxlvl \leq 7$. We terminated the iteration when the solution had six digits, as measured by (8.1). We also provide the total storage for the ja and ju arrays for all matrices, measured in thousands of entries. Since the matrices are symmetric, this is also the total (floating point) storage for all matrices A and approximate LDU factorizations.

Here we see that our method behaves in a very predictable way. In particular, decreasing the drop tolerance or increasing the number of levels improves the conver-

Table 8.2
Dependence of convergence of ϵ and maxlvl, discrete Laplacian A, $N = 160000$.

ϵ	*maxlvl*	Digits	Cycles	Init.	Solve	$\sum \lvert ja\rvert$	$\sum \lvert ju\rvert$
10^{-1}	1	6.0	401	4.3	182.7	479	643
	2	6.0	166	9.3	156.1	878	962
	3	6.1	96	13.2	116.9	1077	1119
	4	6.1	79	15.0	107.3	1176	1178
	5	6.0	75	15.8	106.6	1225	1188
	6	–	–	–	–	–	–
	7	–	–	–	–	–	–
10^{-2}	1	6.0	119	5.8	62.4	479	1236
	2	6.1	56	12.1	64.9	878	2106
	3	6.0	32	14.6	49.3	977	2323
	4	6.4	18	15.1	29.2	1002	2376
	5	6.5	9	15.3	15.5	1008	2388
	6	7.2	7	15.2	12.6	1010	2390
	7	6.1	6	15.3	10.9	1011	2391
10^{-3}	1	6.0	41	8.0	24.9	479	1999
	2	6.1	22	16.6	31.7	878	3649
	3	6.6	13	19.4	25.4	977	4053
	4	6.5	7	20.2	15.2	1002	4147
	5	6.0	4	20.3	9.7	1008	4167
	6	6.5	4	20.3	9.5	1010	4170
	7	6.5	4	20.4	9.4	1011	4171
≈ 0	1	11.1	1	52.4	1.8	479	5626

gence behavior of the method. On the other hand, the timings do not always follow the same trend. For example, for the case $\epsilon = 10^{-3}$ increasing the number of levels from *maxlvl* $= 1$ to *maxlvl* $= 2$ decreases the number of cycles but increases the time. This is because for *maxlvl* $= 1$, our method defaults to the standard conjugate gradient iteration with the incomplete factorization preconditioner. When *maxlvl* > 1, one presmoothing and one postsmoothing step are used for the largest matrix. With the additional cost of the recursion, the overall cost of the preconditioner is more than double the cost for the case *maxlvl* $= 1$.

We also note that, unlike the classical multigrid method, where the coarsest matrix is solved exactly, in our code we have chosen to approximately solve the coarsest system using just one smoothing iteration using the incomplete factorization. When the maximum number of levels are used, as in Table 8.1, the smallest system is typically 1×1 or 2×2, and this is an irrelevant remark. However, in the case of Table 8.2, the fact that the smallest system is not solved exactly significantly influences the overall rate of convergence. This is why, unlike methods where the coarsest system is solved exactly, increasing the number of levels tends to improve the rate of convergence. In the case $\epsilon = 10^{-1}$, the coarsest matrix had an exact LDU factorization for the case *maxlvl* $= 5$ (because the matrix itself was nearly diagonal), and setting *maxlvl* > 5 did not increase the number of levels. The cases $\epsilon = 10^{-2}$ and $\epsilon = 10^{-3}$ used a maximum of 10 and 9 levels, respectively, but the results did not change significantly from the case *maxlvl* $= 7$.

We also include in Table 8.2 the case $\epsilon = 0$, *maxlvl* $= 1$, sparse Gaussian elimination. (In fact, our code uses $\mu\|A\|$ as the drop tolerance when the user specifies

$\epsilon = 0$ to avoid dividing by zero.) Here we see that Gaussian elimination is reasonably competitive on this problem. However, we generally expect the initialization cost for $\epsilon = 0$ to grow like $O(N^{3/2})$. For $maxlvl = 1$ and $\epsilon > 0$, we expect the solution times to grow like $O(N^p)$, $p > 1$. For the best multilevel choices, we expect both initialization and solution times to behave like $O(N) - O(N \log N)$.

In our final series of tests, we study the convergence of the method for a suite of test problems generated from the finite element code *PLTMG* [8]. These example problems were presented in our earlier work [11], where a more complete description of the problems, as well as numerical results for our hierarchical basis multigraph method and the classical AMG algorithm of Ruge and Stüben [46], can be found. As a group, the problems feature highly nonuniform, adaptively generated meshes, relatively complicated geometry, and a variety of differential operators. For each test case, both the sparse matrix and the right-hand side were saved in a file to serve as input for the iterative solvers. A short description of each test problem is given below.

Problem Superior. This problem is a simple Poisson equation

$$-\Delta u = 1$$

with homogeneous Dirichlet boundary conditions on a domain in the shape of Lake Superior. This is the classical problem on a fairly complicated domain. The solution is generally very smooth but has some boundary singularities.

Problem Hole. This problem features discontinuous, anisotropic coefficients. The overall domain is the region between two concentric circles, but this domain is divided into three subregions. On the inner region, the problem is

$$-\delta \Delta u = 0$$

with $\delta = 10^{-2}$. In the middle region, the equation is

$$-\Delta u = 1,$$

and in the outer region the equation is

$$-u_{xx} - \delta u_{yy} = 1.$$

Homogeneous Dirichlet boundary conditions are imposed on the inner (hole) boundary, homogeneous Neumann conditions on the outer boundary, and the natural continuity conditions on the internal interfaces. While the solution is also relatively smooth, singularities exist at the internal interfaces.

Problem Texas. This is an indefinite Helmholtz equation

$$-\Delta u - 2u = 1$$

posed in a region shaped like the state of Texas. Homogeneous Dirichlet boundary conditions are imposed. The length scales of this domain are roughly 16 × 16, so this problem is fairly indefinite.

Problem UCSD. This is a simple constant coefficient convection-diffusion equation

$$-\nabla \cdot (\nabla u + \beta u) = 1,$$

$\beta = (0, 10^5)^T$ posed on a domain in the shape of the UCSD logo. Homogeneous Dirichlet boundary conditions are imposed. Boundary layers are formed at the bottom of the region and the top of various obstacles.

Problems Jcn 0 *and Jcn* 180. The next two problems are solutions of the current continuity equation taken from semiconductor device modeling. This equation is a convection-diffusion equation of the form

$$-\nabla \cdot (\nabla u + \beta u) = 0,$$

$\beta = 0$ in most of the rectangular domain. However, in a curved band in the interior of the domain, $|\beta| \approx 10^4$ and is directed radially. Dirichlet boundary conditions $u = 10^{-5}$ and $u = 10^{10}$ are imposed along the bottom boundary and along a short segment on the upper left boundary, respectively. Homogeneous Neumann boundary conditions are specified elsewhere. The solutions vary exponentially across the domain which is typical of semiconductor problems.

In the first problem, Jcn 0, the convective term is chosen so the device is *forward biased.* In this case, a sharp internal layer develops along the top interface boundary. In the second problem, Jcn 180, the sign of the convective term is reversed, resulting in two sharp internal layers along both interface boundaries.

Table 8.3
Performance comparison.

N	Levels	Digits	Cycles	Init.	Solve
Superior, $\epsilon = 10^{-3}$					
5k	7	7.2	3	2.4e-1	1.0e-1
20k	9	7.3	5	1.4e 0	9.4e-1
80k	9	6.1	7	10.5e 0	6.8e 0
Hole, $\epsilon = 10^{-4}$					
5k	7	6.3	3	4.3e-1	1.5e-1
20k	7	8.2	4	2.4e 0	1.3e 0
80k	8	6.1	5	16.1e 0	7.6e 0
Texas, $\epsilon = 10^{-5}$					
5k	7	12.3	2	4.2e-1	1.1e-1
20k	8	8.2	2	3.0e 0	6.9e-1
80k	9	9.8	5	27.4e 0	10.0e 0
UCSD, $\epsilon = 10^{-3}$					
5k	6	11.1	2	2.0e-1	1.4e-1
20k	6	9.7	2	1.2e 0	7.8e-1
80k	7	8.8	2	10.5e 0	4.0e 0
Jcn 0, $\epsilon = 10^{-4}$					
5k	7	6.4	1	4.5e-1	1.7e-1
20k	7	6.5	1	2.3e 0	8.5e-1
80k	8	10.5	2	15.1e 0	6.2e 0
Jcn 180, $\epsilon = 10^{-5}$					
5k	7	12.3	2	4.9e-1	2.8e-1
20k	7	7.6	2	2.6e 0	1.4e 0
80k	8	7.1	3	18.0e 0	9.3e 0

We summarize the results in Table 8.3. As before, perhaps the most important point is that the method solved all of the problems. While convergence rates are not independent of h, once again the growth appears to be at worst logarithmic.

Below we make some additional remarks.

- For all problems, decreasing the drop tolerance will tend to increase the effectiveness of the preconditioner, although it generally will also make the preconditioner more costly to apply. Thus one might optimize the selection of the drop tolerance to minimize the decreasing number of cycles against the increasing cost per cycle. In these experiments, we did not try such systematic optimization, but we did adjust the drop tolerance in a crude way such that more difficult problems performed in a fashion similar to the easy ones.
- Problem Texas is by far the most difficult in this test suite. While we set $maxfil = 35$, the problem with order $80k$ was the only one which came close to achieving this storage limit. Most were well below this limit, and many averaged less than 10 nonzeros per row in L and U factors.
- For the nonsymmetric problems the CSBCG method is used for acceleration. Since the CSBCG requires the solution of a conjugate system with A^t, two matrix multiplies and two preconditioning steps are required for each iteration. As noted in section 3, with our data structures, applying a transposed matrix and preconditioner costs the same as applying the original matrix or preconditioner. Since these are the dominant costs in the CSBCG methods, the cost per cycle is approximately double that for an equivalent symmetric system.

REFERENCES

[1] O. Axelsson and H. Lu, *On eigenvalue estimates for block incomplete factorization methods*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 1074–1085.

[2] O. Axelsson and H. Lu, *Conditioning analysis of block incomplete factorizations and its application to elliptic equations*, Numer. Math., 78 (1997), pp. 189–209.

[3] O. Axelsson and M. Neytcheva, *The algebraic multilevel iteration methods—theory and applications*, in Proceedings of the Second International Colloquium in Numerical Analysis, Plovdiv, Bulgaria, 1993, pp. 13–23.

[4] O. Axelsson and B. Polman, *Stabilization of algebraic multilevel iteration methods; additive methods*, in AMLI'96: Proceedings of the Conference on Algebraic Multilevel Iteration Methods with Applications, University of Nijmegen, Nijmegen, The Netherlands, 1996, pp. 49–62.

[5] O. Axelsson and P. S. Vassilevski, *Algebraic multilevel preconditioning methods* I, Numer. Math., 56 (1989), pp. 157–177.

[6] I. Babuška, *private communication*, 2000.

[7] Z.-Z. Bai, *A class of hybrid algebraic multilevel preconditioning methods*, Appl. Numer. Math., 19 (1996), pp. 389–399.

[8] R. E. Bank, *PLTMG: A Software Package for Solving Elliptic Partial Differential Equations: Users' Guide* 8.0, Software Environ. Tools 5, SIAM, Philadelphia, 1998.

[9] R. E. Bank and T. F. Chan, *An analysis of the composite step biconjugate gradient method*, Numer. Math., 66 (1993), pp. 295–319.

[10] R. E. Bank and R. K. Smith, *General sparse elimination requires no permanent integer storage*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 574–584.

[11] R. E. Bank and R. K. Smith, *The incomplete factorization multigraph algorithm*, SIAM J. Sci. Comput., 20 (1999), pp. 1349–1364.

[12] R. E. Bank and C. Wagner, *Multilevel ILU decomopsition*, Numer. Math., 82 (1999), pp. 543–576.

[13] R. E. Bank and J. Xu, *The hierarchical basis multigrid method and incomplete LU decomposition*, in Proceedings of the Seventh International Symposium on Domain Decomposition Methods for Partial Differential Equations, D. Keyes and J. Xu, eds., AMS, Providence, RI, 1994, pp. 163–173.

[14] R. E. Bank and J. Xu, *An algorithm for coarsening unstructured meshes*, Numer. Math., 73 (1996), pp. 1–36.

[15] M. Benzi, D. B. Szyld, and A. van Duin, *Orderings for incomplete factorization preconditioning of nonsymmetric problems*, SIAM J. Sci. Comput., 20 (1999), pp. 1652–1670.

[16] D. Braess, *Towards algebraic multigrid for elliptic problems of second order*, Computing, 55 (1995), pp. 379–393.

[17] A. Brandt, S. McCormick, and J. Ruge, *Algebraic Multigrid (AMG) for Automatic Multigrid Solution with Application to Geodetic Computations*, Tech. rep., Institute for Computational Studies, Colorado State University, Fort Collins, CO, 1982.

[18] A. Brandt, S. McCormick, and J. Ruge, *Algebraic multigrid (AMG) for sparse matrix equations*, in Sparsity and Its Applications, D. J. Evans, ed., Cambridge University Press, Cambridge, UK, 1984.

[19] T. F. Chan, S. Go, and J. Zou, *Boundary treatments for multilevel methods on unstructured meshes*, SIAM J. Sci Comput., 21 (1999), pp. 46–66.

[20] A. J. Cleary, R. D. Falgout, V. E. Henson, and J. E. Jones, *Coarse-grid selection for parallel algebraic multigrid*, in Solving Irregularly Structured Problems in Parallel, Lecture Notes in Comput. Sci. 1457, Springer-Verlag, Berlin, 1998, pp. 104–115.

[21] J. E. Dendy, *Black box multigrid*, J. Comput. Phys., 48 (1982), pp. 366–386.

[22] S. C. Eisenstat, H. C. Elman, and M. H. Schultz, *Variational iterative methods for nonsymmetric systems of linear equations*, SIAM J. Numer. Anal., 20 (1983), pp. 345–357.

[23] S. C. Eisenstat, M. H. Schultz, and A. H. Sherman, *Algorithms and data structures for sparse symmetric Gaussian elimination*, SIAM J. Sci. Statist. Comput., 2 (1981), pp. 225–237.

[24] H. C. Elman, *A stability analysis of incomplete LU factorizations*, Math. Comp., 47 (1986), pp. 191–217.

[25] H. C. Elman and X. Zhang, *Algebraic analysis of the hierarchical basis preconditioner*, SIAM J. Matrix Anal., 16 (1995), pp. 192–206.

[26] A. George and J. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1981.

[27] C.-H. Guo, *Incomplete block factorization preconditioning for linear systems arising in the numerical solution of the helmholtz equation*, Appl. Numer. Math., 19 (1996), pp. 495–508.

[28] C.-H. Guo, *Incomplete block factorization preconditioning for indefinite elliptic problems*, Numer. Math., 83 (1999), pp. 621–639.

[29] R. Guo and R. D. Skeel, *An algebraic hierarchical basis preconditioner*, Appl. Numer. Math., 9 (1992), pp. 21–32.

[30] W. Hackbusch, *Multigrid Methods and Applications*, Springer-Verlag, Berlin, 1985.

[31] W. Hackbusch and G. Wittum, *Incomplete Decompositions—Theory, Algorithms and Applications*, Notes Numer. Fluid Mech. 41, Vieweg, Braunschweig, 1993.

[32] G. Karypis and V. Kumar, *Analysis of Multilevel Graph Partitioning*, Tech. rep. 95-037, Department of Computer Science, University of Minnesota, Minneapolis, MN, 1995.

[33] G. Karypis and V. Kumar, *A fast and high quality multilevel scheme for partitioning irregular graphs*, SIAM J. Sci. Comput., 20 (1998), pp. 359–392.

[34] S. LeBorne, *Ordering techniques for convection dominated problems on unstructured three dimensional grids*, in Proceedings of the 11th International Symposium of Domain Decomposition Methods for Partial Differential Equations, C. Lai, P. Bjorstad, M. Cross, and O. Widlund, eds., DDM.org, 1999, pp. 529–536.

[35] C.-J. Lin and J. J. Moré, *Incomplete Cholesky factorizations with limited memory*, SIAM J. Sci. Comput., 21 (1999), pp. 24–45.

[36] M.-M. Magolu, *Ordering strategies for modified block incomplete factorizations*, SIAM J. Sci. Comput., 16 (1995), pp. 378–399.

[37] M.-M. Magolu, *Taking advantage of the potentialities of dynamically modified block incomplete factorizations*, SIAM J. Sci. Comput., 19 (1998), pp. 1083–1108.

[38] J. Mandel, M. Brezina, and P. Vanek, *Energy optimization of algebraic multigrid bases*, Computing, 62 (1999), pp. 205–228.

[39] T. Manneseth, *An analysis of the robustness of some incomplete factorizations*, SIAM J. Sci. Comput., 16 (1995), pp. 1428–1450.

[40] A. Messaoudi, *On the stability of the incomplete LU-factorizations and characterizations of H-matrices*, Numer. Math., 69 (1995), pp. 321–331.

[41] Y. Notay, *Using approximate inverses in algebraic multilevel methods*, Numer. Math., 80 (1998), pp. 397–417.

[42] Y. Notay, *A multilevel block incomplete factorization preconditioning*, Appl. Numer. Math., 31 (1999), pp. 209–225.

[43] C. C. Paige and M. A. Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617–629.

[44] A. Reusken, *A multigrid method based on incomplete Gaussian elimination*, J. Numer. Linear

Algebra Appl., 3 (1996), pp. 369–390.

[45] D. J. Rose, *A graph theoretic study of the numeric solution of sparse positive definite systems*, in Graph Theory and Computing, Academic Press, New York, 1972, pp. 183–217.

[46] J. W. Ruge and K. Stüben, *Algebraic multigrid*, in Multigrid Methods, S. F. McCormick, ed., Frontiers Appl. Math. 3, SIAM, Philadelphia, 1987, pp. 73–130.

[47] Y. Saad, *ILUT: a dual threshold incomplete LU factorization*, Numer. Linear Algebra Appl., 1 (1994), pp. 387–402.

[48] P. Vanek, M. Brezina, and J. Mandel, *Convergence of Algebraic Multigrid Based on Smoothed Aggregation*, Tech. rep. 126, Center for Computational Mathematics, University of Colorado at Denver, Denver, CO, 1994.

[49] C. Wagner, *Introduction to Algebraic Multigrid*, Tech. rep., Interdisziplinäres Zentrum für Wissenschaftliches Rechnen,Universität Heidelberg, Heidelberg, Germany, 1999.

[50] W. L. Wan, T. F. Chan, and B. Smith, *An energy-minimizing interpolation for robust multigrid methods*, SIAM J. Sci. Comput., 21 (2000), pp. 1632–1649.

[51] G. Wittum, *On the robustness of ILU smoothing*, SIAM J. Sci. Comput., 10 (1989), pp. 699–717.

SIAM J. SCI. COMPUT.
Vol. 23, No. 5, pp. 1593–1614

A ROBUST AND EFFICIENT LINEARIZATION SCHEME FOR DOUBLY NONLINEAR AND DEGENERATE PARABOLIC PROBLEMS ARISING IN FLOW IN POROUS MEDIA*

MARIÁN SLODIČKA[†]

Abstract. We study a nonlinear degenerate convection-diffusion model problem having an application in groundwater aquifer and petroleum reservoir simulation. The true solution typically possesses low regularity, and therefore special numerical techniques for its approximation are needed. We design a robust, efficient, and reliable linear relaxation approximation scheme. We prove the convergence of iterations at each time step in the $H^1(\Omega)$-norm. Finally, the convergence of the approximate solution in corresponding functional spaces to its exact counterpart for the parabolic problem is shown.

Key words. flow in porous media, nonlinear degenerate parabolic equation, time discretization, linearization, relaxation scheme, convergence of approximate solution

AMS subject classifications. 35K55, 65M12, 65N12

PII. S1064827500381860

1. Introduction.

Consider an open bounded set $\Omega \subset \mathbb{R}^N$, $N \geq 1$, with a Lipschitz continuous boundary Γ.

In this paper we study the following nonlinear initial boundary value problem (IBVP):

$$\begin{aligned} \partial_t\theta(u) - \nabla\cdot(K(\theta(u))[\nabla u - \boldsymbol{g}]) &= f && \text{in } Q_T = I\times\Omega,\\ u &= u_D && \text{in } I\times\Gamma,\\ u(0) &= u_0 && \text{in } \Omega, \end{aligned} \tag{1.1}$$

where the time interval I is given as $I = (0,T)$ and T is a positive constant.

The nonlinear function θ is supposed to be continuous and monotonically nondecreasing; i.e., the following relation is valid:

$$0 \leq \theta' \leq L \quad \text{a.e. in } \mathbb{R}. \tag{1.2}$$

We do not adopt any growth conditions on θ, which is a standard matter in many other papers. The typical shape of θ function will be shown later.

The function K and the vector $\boldsymbol{g}$ obey

$$\begin{aligned} |K(x) - K(y)| &\leq C|x-y| \qquad \forall x,y\in\mathbb{R},\\ 0 < K_{\min} &\leq K \leq K_{\max},\\ |\boldsymbol{g}| &\leq C, \end{aligned} \tag{1.3}$$

where C denotes a generic positive constant. In many practical situations the function K can approach 0, which is not allowed in our paper. This is the only restrictive assumption we need. Let us note that the generalization from a scalar K to a matrix form $\boldsymbol{K}$ is possible without any additional difficulties.

*Received by the editors December 1, 2000; accepted for publication (in revised form) August 7, 2001; published electronically January 16, 2002. This work was supported by VEO project 011 VO 697 of Ghent University.

http://www.siam.org/journals/sisc/23-5/38186.html

[†]Department of Mathematical Analysis, Faculty of Engineering, Ghent University, Galglaan 2, B-9000 Ghent, Belgium (ms@cage.rug.ac.be).

The initial datum satisfies

$$u_0 \in L_2(\Omega). \tag{1.4}$$

The source term f and the function u_D are tacitly assumed to be smooth enough for our purposes.

We denote the usual functional spaces $L_2(\Omega)$, $H^1(\Omega)$, $H_0^1(\Omega)$, $H^{-1}(\Omega)$ (the dual space to $H_0^1(\Omega)$), $L_2(I, L_2(\Omega))$, $L_2(I, H_0^1(\Omega))$, $L_\infty(I, L_2(\Omega))$, and $L_2(I, H^{-1}(\Omega))$; see Kufner, John, and Fučík [20].

The notation (w, z) stands for the standard L_2-inner product (or sometimes the duality pairing) of any real or vector-valued functions w, z.

Equation (1.1(a)) changes its type from parabolic to elliptic, depending on the values of $\theta(u)$. Thus we do not expect the solution u to be smooth, and therefore we define a weak solution as follows.

Definition 1.1. *A function u is a weak solution to the IBVP* (1.1) *if*

1. $u - u_D \in L_2(I, H_0^1(\Omega))$,
2. $\theta(u) \in C(I, H^{-1}(\Omega)) \cap L_\infty(I, L_2(\Omega))$,
3. $\partial_t \theta(u) \in L_2(I, H^{-1}(\Omega))$,

and the following integral identity is satisfied:

$$(\partial_t \theta(u), \varphi) + (K(\theta(u))[\nabla u - \boldsymbol{g}], \nabla \varphi) = (f, \varphi) \tag{1.5}$$

for all $\varphi \in H_0^1(\Omega)$ and a.e. in $(0, T)$.

1.1. The physical context. The starting point in mathematical modeling of processes in porous media is *Darcy's law*, which was originally introduced for saturated soils and has been extended to unsaturated soils by *Buckingham*:

$$\boldsymbol{q} = -\frac{k}{\mu}(\nabla p - \rho \boldsymbol{g}), \tag{1.6}$$

where ρ is the density of the fluid, $\boldsymbol{g}$ is the gravity acceleration vector directed downward, p is the pressure, k is the permeability of the medium, and μ is the dynamic viscosity. The factor

$$K = \frac{k}{\mu}$$

is called hydraulic conductivity.

The relation between the specific discharge vector $\boldsymbol{q}$ and the hydraulic gradient ∇p expressed by Darcy's law is linear, indicating laminar flow conditions. This remains valid for small *Reynold's numbers*, namely between 1 and 10.

If the fluid fills the relative volume θ, one gets the continuity equation

$$\partial_t(\rho\theta) + \nabla \cdot (\rho \boldsymbol{q}) = \rho f, \tag{1.7}$$

where f stands for possible sources/sinks. The continuity equation (1.7) together with Darcy's law (1.6) implies for incompressible fluids ($\rho = \text{const}$)

$$\partial_t \theta - \nabla \cdot (K[\nabla p - \rho \boldsymbol{g}]) = f, \tag{1.8}$$

which is known as Richards' equation.

Several studies have shown the nonlinear dependence of θ on the pressure p and also the nonlinear dependence of K on θ. A number of empirical and semiempirical functions have been proposed in the past to represent the water retention curves (cf., e.g., Brooks and Corey [8], Brutsaert [9], Mualem [24], Vauclin, Khanji, and Vachaud, [39], van Genuchten [38], and Bumb, Murphy, and Everett [10]).

Van Genuchten [38] has introduced an attractive class of S functions of the form $S(p) = (1 + |\alpha p|^n)^{-m}$ and he has presented the model

$$\begin{cases} \theta(p) = \begin{cases} \theta_r + \dfrac{\theta_s - \theta_r}{(1 + |\alpha p|^n)^m} & \text{for } p \leq 0, \\ \theta_s & \text{for } p \geq 0, \end{cases} \\ K(S) = K_s S^{\frac{1}{2}}[1 - (1 - S^{\frac{1}{m}})^m]^2, \\ S = \dfrac{\theta - \theta_r}{\theta_s - \theta_r}, \end{cases} \tag{1.9}$$

with coefficients α, n, and $m = 1 - \frac{1}{n}$. The meaning of other symbols is explained in Table 1.1. The typical behavior of $\theta(p), K(\theta)$ is shown in Figure 1.1.

Table 1.1
Notations for van Genuchten's model.

p	$[m]$	pressure head
θ	[-]	water content
θ_s	[-]	saturated value of the soil-water content
θ_r	[-]	residual value of the soil-water content
K	$[\frac{m}{s}]$	hydraulic conductivity
K_s	$[\frac{m}{s}]$	hydraulic conductivity of the saturated zone

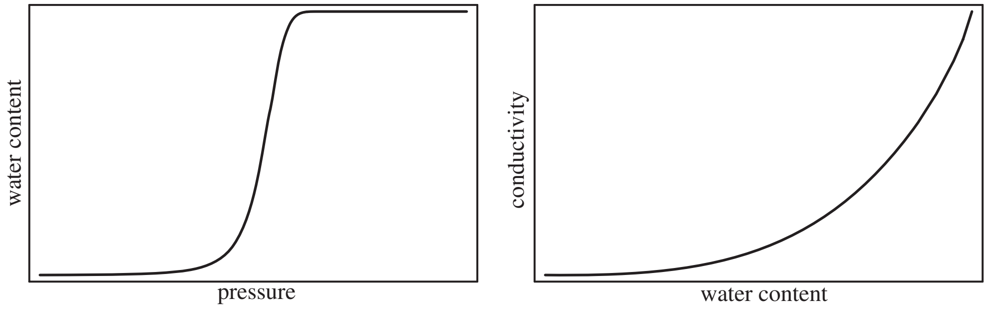


Fig. 1.1. *Soil-water retention curve and hydraulic conductivity versus water content using van Genuchten's model.*

Remark 1.2. Van Genuchten's model describes any geological layer by a set of parameters θ_r, θ_s, K_s, α, and n. The functional dependences are given by (1.9). Analyzing properties of the function $\theta(p)$, one can validate (1.2), where the Lipschitz constant depends on van Genuchten's parameters.

The function K can, in fact, approach 0. This happens in dry soils when $\theta = \theta_r$ only. In many practical applications the condition $K \geq K_{\min}$ is acceptable.

1.2. The state of the art. The van Genuchten model is widely used in the groundwater modeling. Combining the equations of van Genuchten and Richards, we arrive at a doubly nonlinear degenerate parabolic equation. Many mathematical papers have been devoted to the qualitative and quantitative analysis of such problems (cf., e.g., van Duijn and Peletier [37], Alt and Luckhaus [1], and Otto [28]). The authors have proved existence and uniqueness theorems for elliptic-parabolic equations in a more general form, which is suitable for saturated and unsaturated single-phase flow through porous medium.

The idea of studying abstract partial differential equations by means of the theory of nonlinear semigroups of contractions in Banach spaces was proposed by Brezis [6]. This approach not only has theoretical but also numerical aspects. It suggests some algorithms for time discretization to approximate the exact solution. Some of them are based on the so-called Crandall–Liggett formula (cf. Crandall and Liggett [12]) and others on the nonlinear Chernoff formula (cf. Magenes, Nochetto, and Verdi [22], Nochetto [27], [26], and Verdi [40]).

Optimal rates of convergence for degenerate parabolic problems for nonlinear schemes were established by Rulla [32] for semidiscretization in space and by Rulla and Walkington [33] for the full discretization in two dimensions. Some convergence results can also be found in Eymard, Gutnic, and Hilhorst [13] for the finite volume method and in Arbogast, Wheeler, and Zhang [3] or Woodward and Dawson [42] for the mixed finite element method.

Some strongly convergent schemes for the Richards' equation were formulated in [23] upon modification of the Newton–Raphson method, including the method of lines formulations based upon ordinary differential equations. Discretizing the partial differential equation in space by, e.g., finite differences or finite elements, one arrives at a system of differential algebraic equations. For solving this problem we refer the reader to [36].

Many authors apply the backward Euler method for the time discretization. Then a nonlinear elliptic problem has to be solved at each successive time point of a suitable time partitioning of the time interval. Here two ways are possible: to solve a nonlinear elliptic problem[1] or to apply a linearization first. To enhance the robustness of the standard Newton method for solving a system of nonlinear algebraic equations, special techniques (line-search backtracking procedure, preconditioning, etc.) can be used; see [17].

Jäger and Kačur [15], [16], and Kačur [19] have developed effective relaxation schemes for linearization of nonlinear elliptic problems at each time step. The iteration scheme (according to the relaxation parameter κ) has to stop when a prescribed residual condition (iteration error) is satisfied. The convergence for $\kappa \to \infty$ in one dimension of such a relaxation scheme has been shown in Kačur [19]. The proof in more dimensions has not been presented because of lack of regularity of the solution. Moreover, the regularization of the nonlinear function is made using the time step τ. Therefore, one can expect that the convergence of iterations at each time point should depend on τ. In fact, Kačur [19] has shown the estimate

$$|u - u_\kappa| \leq C\tau^r \; |u - u_{\kappa-1}|, \quad r > 0,$$

which represents a contraction only in the case when the time step τ is sufficiently small.

[1] This means that after the full discretization a nonlinear system of algebraic equations has to be solved. This can be done using standard iterative methods, but the discussion of possible iteration error is neglected. The convergence of these methods strongly depends on the choice of the starting iteration, which should be relatively close to the solution. Usually, the foregoing time step solution is chosen. This means that the time step must be chosen sufficiently small in order to find a solution of the nonlinear algebraic system. Recall that the standard method of applying a fixed point iteration to the time step solution will have a wider radius of convergence than applying Newton's method; however, Newton's method has a faster rate of convergence once within its convergence radius.

Another way of linearization relies on Newton's method. It is based either on a regularization of a nonlinear function (cf. Amiez and Gremaud [2], Slodička [34], [35]) or on a shifting of the data (cf. Pop [30], Pop and Yong [31]) instead of a modification of the nonlinear function. Whereas the additional iterations on each time step are omitted, the regularization or the shifting of the data must depend on the time step; otherwise, the convergence cannot be reached.

An attractive group of linearization schemes represents the method of upper and lower solutions (cf. Pao [29] and Wang and Pao [41]). The linearization of a nonlinear problem relies on the ordering properties of solutions. One defines recursive sequences starting from a sub- or a supersolution, respectively. The disadvantage is that one cannot use the solution from the previous time step as a starting iteration.

The case of a non-Lipschitz continuous function θ has been analyzed, e.g., in Barrett and Knabner [5] or Kačur [19].

1.3. Results. The aim of this paper is to present an efficient and robust linear approximation scheme for (1.1), to prove its convergence, and to address the error estimates. The time discretization is based on the backward Euler method. The concept behind our linear approximation is a clever relaxation scheme for solving a nonlinear elliptic BVP at each time point of the time partitioning.

Let us denote the time step $\tau = \frac{T}{n}$ for any $n \in \mathbb{N}$. The approximate solution $u_i \approx u(t_i)$ at a given time point $t_i = i\tau$ is obtained in an iteration process with respect to the relaxation parameter κ. The linearized scheme for a fixed $i \in \{1, \dots, n\}$ and running $\kappa = 1, 2, \dots$ reads as

$$\begin{aligned} u_{i,0} &= u_{i-1}, \\ \frac{L}{\tau}(u_{i,\kappa}, \varphi) + (K(\theta(u_{i-1}))\nabla u_{i,\kappa}, \nabla\varphi) &= (f_i, \varphi) + (K(\theta(u_{i-1}))\boldsymbol{g}, \nabla\varphi) + \frac{L}{\tau}(u_{i,\kappa-1}, \varphi) \\ &\quad - \frac{1}{\tau}(\theta(u_{i,\kappa-1}), \varphi) + \frac{1}{\tau}(\theta(u_{i-1}), \varphi) \end{aligned} \tag{1.10}$$

for all $\varphi \in H_0^1(\Omega)$, where $f_i = f(t_i)$. We recall that L is the Lipschitz constant of the function θ; see (1.2). The iteration process stops when the following condition is satisfied:

$$\|u_{i,\kappa} - u_{i,\kappa-1}\| \le C_d \tau^d, \tag{1.11}$$

where $C_d > 0,\ d > 1$ are fixed constants and $\|\cdot\|$ stands for the usual $L_2(\Omega)$-norm. After stopping the iterations at $\kappa = \kappa_{i,\text{last}}$, we denote $u_i := u_{i,\kappa_{i,\text{last}}}$ and switch to the next time step.

We prove the convergence of relaxation iterations for $\Omega \subset \mathbb{R}^N$, $N \ge 1$. (Compare with Kačur [19], where the convergence in one dimension and for small τ has been shown.) Our iteration scheme with respect to the parameter κ can start from arbitrary data $u_{i,0} = u_{i-1} = u_{i-1,\kappa_{i-1,\text{last}}}$, which makes the proposed numerical method reliable and robust. For $i = 1$, the iteration scheme arises from the initial data u_0. This is one of the highlights of the paper because up until now there does not exist any other linearization scheme with this property. Numerical examples confirming the efficiency of our approximation method are presented in section 5.

Throughout the paper we assume that a generic positive constant C is independent of the iteration parameter κ and the time step τ.

2. A priori estimates. Let us note that the problem (1.10), for given i and κ, admits a unique solution $u_{i,\kappa} \in H_0^1(\Omega)$. This immediately follows from the theory of linear elliptic equations (cf. Gilbarg and Trudinger [14]) taking into account the assumptions (1.2)–(1.4).

The main task of this section is to derive a priori estimates for u_i and $\theta(u_i)$ which are uniform with respect to $i = 1, \ldots, n$ and κ. In Lemma 3.1 we will prove the relation (1.11). Thus we can assume throughout the whole section that the iteration process at each time step t_i stops after $\kappa = \kappa_{i,\text{last}}$ iterations.

We introduce the notation

$$\delta z_i = \frac{z_i - z_{i-1}}{\tau}.$$

We rewrite the iteration scheme (1.10) as follows ($\kappa = \kappa_{i,\text{last}}$, $u_i = u_{i,\kappa_{i,\text{last}}}$):

$$\begin{aligned}(\delta\theta(u_i), \varphi) + (K(\theta(u_{i-1}))\nabla u_i, \nabla\varphi) &= (f_i, \varphi) + (K(\theta(u_{i-1}))\boldsymbol{g}, \nabla\varphi) \\ &\quad + \frac{L}{\tau}(u_{i,\kappa-1} - u_{i,\kappa}, \varphi) \\ &\quad + \frac{1}{\tau}(\theta(u_{i,\kappa}) - \theta(u_{i,\kappa-1}), \varphi)\end{aligned} \tag{2.1}$$

for all $\varphi \in H_0^1(\Omega)$. This form is more convenient for our purposes.

Lemma 2.1. *Let* (1.2), (1.3), *and* (1.4) *be satisfied. Moreover, we assume* (1.11). *Then there exists a positive constant C such that*

$$\|\theta(u_j)\|^2 + \sum_{i=1}^{j} \|\theta(u_i) - \theta(u_{i-1})\|^2 + \sum_{i=1}^{j} \|\nabla\theta(u_i)\|^2\tau \le C$$

holds for all $j = 1, \ldots, n$.

Proof. We put $\varphi = \theta(u_i)\tau$ in (2.1), sum it up for $i = 1, \ldots, j$, and get

$$\begin{aligned}\sum_{i=1}^{j}(\theta(u_i) - \theta(u_{i-1}), \theta(u_i)) &+ \sum_{i=1}^{j}(K(\theta(u_{i-1}))\nabla u_i, \nabla\theta(u_i))\tau \\ = \sum_{i=1}^{j}(f_i, \theta(u_i))\tau &+ \sum_{i=1}^{j}(K(\theta(u_{i-1}))\boldsymbol{g}, \nabla\theta(u_i))\tau \\ &+ \frac{L}{\tau}\sum_{i=1}^{j}(u_{i,\kappa-1} - u_{i,\kappa}, \theta(u_i))\tau \\ &+ \frac{1}{\tau}\sum_{i=1}^{j}(\theta(u_{i,\kappa}) - \theta(u_{i,\kappa-1}), \theta(u_i))\tau.\end{aligned} \tag{2.2}$$

Using (1.3) we deduce

$$(K(\theta(u_{i-1}))\nabla u_i, \nabla\theta(u_i)) = \left(\frac{K(\theta(u_{i-1}))}{\theta'(u_i)}\nabla\theta(u_i), \nabla\theta(u_i)\right) \ge \frac{K_{\min}}{L}\|\nabla\theta(u_i)\|^2.$$

The following identity holds:

$$2\sum_{i=1}^{j}(a_i - a_{i-1})a_i = a_j^2 - a_0^2 + \sum_{i=1}^{j}(a_i - a_{i-1})^2.$$

The relations (1.2) and (1.4) and the triangle inequality imply

$$\|\theta(u_0)\| \le \|\theta(u_0) - \theta(0)\| + \|\theta(0)\| \le L\|u_0\| + C \le C.$$

Thus we can estimate the left-hand side of (2.2) from below by

$$C_0[\|\theta(u_j)\|^2 + \sum_{i=1}^{j} \|\theta(u_i) - \theta(u_{i-1})\|^2 + \sum_{i=1}^{j} \|\nabla\theta(u_i)\|^2\tau] - C.$$

Using the Cauchy–Schwarz and Young's ($|ab| \le \varepsilon a^2 + C_\varepsilon b^2$, where $\varepsilon > 0$ and $C_\varepsilon = C(\frac{1}{\varepsilon})$) inequalities, we deduce

$$\begin{aligned}|(f_i, \theta(u_i)) + (K(\theta(u_{i-1}))\boldsymbol{g}, \nabla\theta(u_i))| &\le C\|\theta(u_i)\| + C\|\nabla\theta(u_i)\| \\ &\le C_\varepsilon + C\|\theta(u_i)\|^2 + \varepsilon\|\nabla\theta(u_i)\|^2.\end{aligned}$$

Now, applying (1.11) and the Lipschitz continuity of θ, we estimate in a similar way

$$\begin{aligned}\left|\frac{L}{\tau}(u_{i,\kappa-1} - u_{i,\kappa}, \theta(u_i)) + \frac{1}{\tau}(\theta(u_{i,\kappa}) - \theta(u_{i,\kappa-1}), \theta(u_i))\right| &\le \frac{C}{\tau}\|u_{i,\kappa-1} - u_{i,\kappa}\|\|\theta(u_i)\| \\ &\le C\tau^{d-1}\|\theta(u_i)\| \\ &\le CT^{d-1}\|\theta(u_i)\| \\ &\le C + C\|\theta(u_i)\|^2.\end{aligned}$$

Therefore, the upper bound of the right-hand side in (2.2) reads as

$$\varepsilon\sum_{i=1}^{j} \|\nabla\theta(u_i)\|^2\tau + C_\varepsilon + C\sum_{i=1}^{j} \|\theta(u_i)\|^2\tau.$$

Fixing sufficiently small positive ε, we arrive at

$$\|\theta(u_j)\|^2 + \sum_{i=1}^{j} \|\theta(u_i) - \theta(u_{i-1})\|^2 + \sum_{i=1}^{j} \|\nabla\theta(u_i)\|^2\tau \le C + C\sum_{i=1}^{j} \|\theta(u_i)\|^2\tau.$$

The rest of the proof follows from Gronwall's lemma. □

Let us introduce the following notation $\Phi_\theta(z) := \int_0^z \theta(s)\mathrm{d}s$ for the Kirchhoff transformation. The function θ is monotonically increasing; therefore,

$$\theta(z_1)(z_2 - z_1) \le \Phi_\theta(z_2) - \Phi_\theta(z_1) \le \theta(z_2)(z_2 - z_1) \tag{2.3}$$

holds for all $z_1, z_2 \in \mathbb{R}$. Further, we can write for any $z \in \mathbb{R}$

$$\tilde{\Phi}_\theta(z) := z\theta(z) - \Phi_\theta(z) \ge 0. \tag{2.4}$$

According to the Lipschitz continuity of the function θ we estimate

$$\tilde{\Phi}_\theta(z) \le C(1 + z^2) \quad \forall z \in \mathbb{R}. \tag{2.5}$$

LEMMA 2.2. *Let the assumptions of Lemma* 2.1 *be satisfied. Then there exists a positive constant C such that*

$$\sum_{i=1}^{j} \|\nabla u_i\|^2\tau \le C$$

holds for all $j = 1, \dots, n$.

Proof. We choose $\varphi = u_i\tau$ in (2.1), sum it up for $i = 1, \dots, j$ and get

$$\begin{aligned}\sum_{i=1}^{j}(\theta(u_i) - \theta(u_{i-1}), u_i) + \sum_{i=1}^{j}(K(\theta(u_{i-1}))\nabla u_i, \nabla u_i)\tau \\ = \sum_{i=1}^{j}(f_i, u_i)\tau + \sum_{i=1}^{j}(K(\theta(u_{i-1}))\boldsymbol{g}, \nabla u_i)\tau \\ + \frac{L}{\tau}\sum_{i=1}^{j}(u_{i,\kappa-1} - u_{i,\kappa}, u_i)\tau \\ + \frac{1}{\tau}\sum_{i=1}^{j}(\theta(u_{i,\kappa}) - \theta(u_{i,\kappa-1}), u_i)\tau.\end{aligned}$$

Using (2.3)–(2.5) we deduce for the first term

$$\begin{aligned}\sum_{i=1}^{j}(\theta(u_i) - \theta(u_{i-1}), u_i) &= (\theta(u_j), u_j) - (\theta(u_0), u_0) - \sum_{i=1}^{j}(u_i - u_{i-1}, \theta(u_{i-1})) \\ &\geq (\theta(u_j), u_j) - (\theta(u_0), u_0) - \sum_{i=1}^{j}\int_\Omega[\Phi_\theta(u_i) - \Phi_\theta(u_{i-1})] \\ &= (\theta(u_j), u_j) - (\theta(u_0), u_0) - \int_\Omega \Phi_\theta(u_j) + \int_\Omega \Phi_\theta(u_0) \\ &= \int_\Omega[\tilde{\Phi}_\theta(u_j) - \tilde{\Phi}_\theta(u_0)] \\ &\geq -C(1 + \|u_0\|^2) \\ &\geq -C.\end{aligned}$$

The second term satisfies

$$\sum_{i=1}^{j}(K(\theta(u_{i-1}))\nabla u_i, \nabla u_i)\tau \geq K_{\min}\sum_{i=1}^{j}\|\nabla u_i\|^2\tau.$$

The right-hand side can be estimated from the top using the Cauchy–Schwarz inequality and (1.11), followed by an application of Young's and Friedrichs' inequalities as well as Lemma 2.1 by

$$\varepsilon\sum_{i=1}^{j}\|\nabla u_i\|^2\tau + C_\varepsilon,$$

where $\varepsilon \in \mathbb{R}_+$. Summarizing all estimates and choosing sufficiently small ε, we conclude the proof. □

Let us denote the norm in $H^1(\Omega)$ by $\|\cdot\|_1$. We recall that the norm in the space $H^{-1}(\Omega)$ is defined by

$$\|\cdot\|_{-1} := \|\cdot\|_{H^{-1}(\Omega)} = \sup_{\substack{\varphi\in H_0^1(\Omega)\\ \|\varphi\|_1\leq 1}} |(\cdot, \varphi)|.$$

Lemma 2.3. *Let the assumptions of Lemma* 2.1 *be satisfied. There is a positive constant C such that for any $j = 1, \dots, n$,*

$$\sum_{i=1}^{j}\|\delta\theta(u_i)\|_{-1}^2\tau \leq C.$$

Proof. We start from the relation (2.1). We leave the first term on the left and the rest we shift to the right; thus we have

$$\begin{aligned}(\delta\theta(u_i),\varphi) &= (f_i,\varphi) + (K(\theta(u_{i-1}))\boldsymbol{g},\nabla\varphi) - (K(\theta(u_{i-1}))\nabla u_i,\nabla\varphi)\\ &\quad + \frac{L}{\tau}(u_{i,\kappa-1}-u_{i,\kappa},\varphi) + \frac{1}{\tau}(\theta(u_{i,\kappa})-\theta(u_{i,\kappa-1}),\varphi).\end{aligned}$$

For the first three terms on the right, we deduce using the Cauchy–Schwarz inequality and (1.3)

$$|(f_i,\varphi) + (K(\theta(u_{i-1}))\boldsymbol{g},\nabla\varphi) - (K(\theta(u_{i-1}))\nabla u_i,\nabla\varphi)| \le C(1+\|\nabla u_i\|)\|\nabla\varphi\| + C\|\varphi\|.$$

The last two terms on the right can be estimated using the Lipschitz continuity of the function θ, the Cauchy–Schwarz inequality, and (1.11) as follows:

$$\begin{aligned}\left|\frac{L}{\tau}(u_{i,\kappa-1}-u_{i,\kappa},\varphi) + \frac{1}{\tau}(\theta(u_{i,\kappa})-\theta(u_{i,\kappa-1}),\varphi)\right| &\le \frac{C}{\tau}\|u_{i,\kappa-1}-u_{i,\kappa}\|\|\varphi\|\\ &\le C\tau^{d-1}\|\varphi\|\\ &\le CT^{d-1}\|\varphi\|\\ &\le C\|\varphi\|.\end{aligned}$$

Hence we can write

$$|(\delta\theta(u_i),\varphi)| \le C(1+\|\nabla u_i\|)\|\nabla\varphi\| + C\|\varphi\|,$$

from which we deduce

$$\|\delta\theta(u_i)\|_{-1} \le C(1+\|\nabla u_i\|).$$

Taking the second power, multiplying the inequality by τ, summing it up for $i = 1,\dots,j$, and applying Lemma 2.2, we get the desired result. □

3. Convergence of the relaxation scheme. In this section we prove the convergence of $u_{i,\kappa}$ for $\kappa\to\infty$. Recall that $u_{i,\kappa}$ is the unique solution to the linear elliptic problem (1.10).

To enhance the readability of the proof, we introduce the notation

$$v_\kappa = u_{i,\kappa}, \qquad F = f_i + \frac{\theta(u_{i-1})}{\tau}, \qquad K = K(\theta(u_{i-1})).$$

Thus we consider the following problem:

$$\begin{aligned}L(v_\kappa,\varphi) + \tau(K\nabla v_\kappa,\nabla\varphi) &= \tau(F,\varphi) + \tau(K\boldsymbol{g},\nabla\varphi) + L(v_{\kappa-1},\varphi)\\ &\quad - (\theta(v_{\kappa-1}),\varphi)\end{aligned} \tag{3.1}$$

for all $\varphi\in H_0^1(\Omega)$. We show that v_κ converges for $\kappa\to\infty$ to a solution $v\in H_0^1(\Omega)$ of the following nonlinear elliptic problem:

$$(\theta(v),\varphi) + \tau(K\nabla v,\nabla\varphi) = \tau(F,\varphi) + \tau(K\boldsymbol{g},\nabla\varphi) \quad \forall\varphi\in H_0^1(\Omega). \tag{3.2}$$

The left-hand side of (3.2) is a maximal monotone and coercive operator from $H_0^1(\Omega)$ to $H^{-1}(\Omega)$. The right-hand side of (3.2) is a bounded linear functional on $H_0^1(\Omega)$. Hence, accounting for Brézis [7, Corollaire 2.4, p. 31], the problem (3.2) admits exactly one solution $v\in H_0^1(\Omega)$.

Now we define the following function:

$$h(s) := \theta(s) - Ls, \quad s \in \mathbb{R}. \tag{3.3}$$

Subtracting (3.2) from (3.1), we get the variational formulation for the error $v_\kappa - v$

$$L(v_\kappa - v, \varphi) + \tau(K\nabla[v_\kappa - v], \nabla\varphi) = (h(v) - h(v_{\kappa-1}), \varphi) \quad \forall\varphi \in H_0^1(\Omega). \tag{3.4}$$

Lemma 3.1. *Let* (1.2), (1.3)*, and* (1.4) *be fulfilled. Then there exists a positive constant* $\lambda = \lambda(\Omega, K_{\min})$ *such that*

$$\begin{aligned}\|v_\kappa - v\|^2 &\le \left(1 - \frac{\tau\lambda}{L+\tau\lambda}\right)^\kappa \|v_0 - v\|^2\\ \|\nabla[v_\kappa - v]\|^2 &\le \frac{L+\tau\lambda}{\tau K_{\min}}\left(1 - \frac{\tau\lambda}{L+\tau\lambda}\right)^\kappa \|v_0 - v\|^2\end{aligned}$$

holds for all $\kappa = 1, 2, \dots$.

Proof. Choose $\varphi = v_\kappa - v \in H_0^1(\Omega)$ in (3.4) and get

$$L(v_\kappa - v, v_\kappa - v) + \tau(K\nabla[v_\kappa - v], \nabla[v_\kappa - v]) = (h(v) - h(v_{\kappa-1}), v_\kappa - v). \tag{3.5}$$

The crucial point is to estimate the right-hand side. To do this, we use (1.2) and deduce

$$-L \le h' = \theta' - L \le 0 \quad \text{a.e. in } \mathbb{R}.$$

Hence the derivative of the function h is bounded by the constant L, i.e., $|h'| \le L$ a.e. in $\mathbb{R}$.

Applying the Cauchy–Schwarz and Young's inequalities, we deduce

$$\begin{aligned}|(h(v) - h(v_{\kappa-1}), v_\kappa - v)| &\le \|h(v) - h(v_{\kappa-1})\|\|v_\kappa - v\|\\ &\le L\|v - v_{\kappa-1}\|\|v_\kappa - v\|\\ &\le \frac{L}{2}\|v - v_{\kappa-1}\|^2 + \frac{L}{2}\|v_\kappa - v\|^2.\end{aligned}$$

The left-hand side of (3.5) can be estimated from below by

$$L\|v_\kappa - v\|^2 + \tau K_{\min}\|\nabla[v_\kappa - v]\|^2.$$

Friedrichs' inequality (cf., e.g., Křížek and Neittaanmäki [21, p. 26]) implies the existence of a positive real number $\lambda = \lambda(\Omega, K_{\min})$ such that

$$\lambda\|w\|^2 \le K_{\min}\|\nabla w\|^2 \quad \forall w \in H_0^1(\Omega). \tag{3.6}$$

According to this fact,

$$\left(L + \frac{\tau\lambda}{2}\right)\|v_\kappa - v\|^2 + \frac{\tau K_{\min}}{2}\|\nabla[v_\kappa - v]\|^2$$

is also a lower bound to the left-hand side of (3.5).

Summarizing the foregoing results, we arrive at

$$(L + \tau\lambda)\|v_\kappa - v\|^2 + \tau K_{\min}\|\nabla[v_\kappa - v]\|^2 \le L\|v_{\kappa-1} - v\|^2,$$

which after a simple calculation gives

$$\|v_\kappa - v\|^2 + \frac{\tau K_{\min}}{L+\tau\lambda}\|\nabla[v_\kappa - v]\|^2 \le \left(1 - \frac{\tau\lambda}{L+\tau\lambda}\right)\|v_{\kappa-1} - v\|^2. \tag{3.7}$$

We omit the second term on the left for a moment and obtain the recursion formula

$$\|v_\kappa - v\|^2 \le \left(1 - \frac{\tau\lambda}{L+\tau\lambda}\right)\|v_{\kappa-1} - v\|^2.$$

After κ iterations, this implies

$$\|v_\kappa - v\|^2 \le \left(1 - \frac{\tau\lambda}{L+\tau\lambda}\right)^\kappa \|v_0 - v\|^2.$$

The rest of the proof comes from the last inequality and (3.7). □

Let us discuss the results of Lemma 3.1 with respect to the scheme (1.10). We point out first that the choice of the time step τ is free. Further, the relaxation iterations can start from arbitrary starting iterations lying in the space $L_2(\Omega)$, and nevertheless they converge in the $H^1(\Omega)$-norm to a function $v \in H_0^1(\Omega)$, which is given by (3.2). Please note that $v \approx u(t_i)$ at the time step i. These properties make the relaxation process reliable and robust.

Lemma 3.1 says that $v_\kappa \to v$ as $\kappa \to \infty$ in the space $L_2(\Omega)$. Thus, for any $\tau > 0$ and any $d > 1$, there exists $\kappa_0 \in \mathbb{N}$ such that $\|v_\kappa - v\| \le \tau^d$ holds for any $\kappa \ge \kappa_0$. Setting $\kappa_{\text{last}} = \kappa_0 + 1$ and using the triangle inequality, we deduce

$$\|v_{\kappa_{\text{last}}} - v_{\kappa_{\text{last}}-1}\| \le \|v_{\kappa_{\text{last}}} - v\| + \|v - v_{\kappa_{\text{last}}-1}\| \le 2\tau^d.$$

Therefore, Lemma 3.1 validates the stopping criterion (1.11). Let us note that the value of κ_{last} can change with the time point t_i. Nevertheless, the constant $C_d = 2$ remains fixed.

4. Convergence of the scheme (1.10). Up until now, we have successively determined u_i for $i = 1, \dots, n$ as the solution of the linear relaxation scheme (1.10). Each $u_i = u_{i,\kappa_{i,\text{last}}}$ has been obtained after a finite number of relaxation iterations so that the stopping criterion (1.11) has been achieved. Now we define a function $w_n(t)$ as a piecewise linear function in time which is defined in terms of u_i $(i = 1, \dots, n)$ as follows:

$$\begin{aligned} &w_n(0) = \theta(u_0),\\ &w_n(t) = \theta(u_{i-1}) + (t - t_{i-1})\delta\theta(u_i) \quad \text{for } t \in (t_{i-1}, t_i] \end{aligned}$$

and the step functions $\overline{w}_n, \overline{u}_n$

$$\begin{aligned} &\overline{w}_n(0) = \theta(u_0), \qquad &&\overline{w}_n(t) = \theta(u_i),\\ &\overline{u}_n(0) = u_0, &&\overline{u}_n(t) = u_i \quad \text{for } t \in (t_{i-1}, t_i]. \end{aligned}$$

In a similar way as $\overline{u}_n$ we also define the function $\overline{f}_n$.

Now we will rewrite the relation (2.1) in terms of the functions we just introduced. First, in light of (1.11) and the Lipschitz continuity of the function θ, the last two terms on the right-hand side of (2.1) can be estimated as follows:

$$\left|\frac{L}{\tau}(u_{i,\kappa-1} - u_{i,\kappa}, \varphi) + \frac{1}{\tau}(\theta(u_{i,\kappa}) - \theta(u_{i,\kappa-1}), \varphi)\right| \le 2LC_d\tau^{d-1}\|\varphi\|$$

uniformly with respect to i ($\kappa = \kappa_{i,\mathrm{last}}$).

Thus we can rewrite the relation (2.1) in an equivalent form

$$\begin{aligned}(\partial_t w_n(t), \varphi) + (K(\overline{w}_n(t-\tau))\nabla \overline{u}_n(t), \nabla\varphi) \\ = (\overline{f}_n(t), \varphi) + (K(\overline{w}_n(t-\tau))\boldsymbol{g}, \nabla\varphi) + \mathcal{O}(\tau^{d-1})\|\varphi\|\end{aligned} \tag{4.1}$$

for all $\varphi \in H_0^1(\Omega)$.

We would like to examine the limit as $n \to \infty$ in (4.1), i.e., we want to prove the convergence (in a suitable function space) of the approximation $\overline{u}_n$ to a weak solution u to the IBVP (1.1), as well as to show the convergence of $w_n, \overline{w}_n$ to $\theta(u)$ and the convergence of $\partial_t w_n$ towards $\partial_t \theta(u)$. We will do it in a few steps. First, we rewrite the a priori estimates from Lemmas 2.1–2.3 in terms of the just defined functions as follows:

$$\begin{gathered}\int_0^T \|\partial_t w_n(t)\|_{-1}^2 \mathrm{d}t \le C, \\ \max_{t\in[0,T]} \|w_n(t)\| \le C, \\ \int_0^T \|w_n(t) - \overline{w}_n(t)\|^2 \mathrm{d}t \le \frac{C}{n}, \\ \int_0^T \|\nabla \overline{w}_n(t)\|^2 \mathrm{d}t \le C, \\ \int_0^T \|\nabla \overline{u}_n(t)\|^2 \mathrm{d}t \le C.\end{gathered} \tag{4.2}$$

The third inequality follows from

$$\|w_n(t) - \overline{w}_n(t)\| \le 2\|\theta(u_i) - \theta(u_{i-1})\|$$

for $t \in (t_{i-1}, t_i]$ and $i = 1, \dots, n$.

Next, using (4.2) and standard results from the functional analysis, we prove the existence of a subsequence of $\overline{u}_n$, which converges to some function $u \in L_2(I, H_0^1(\Omega))$, and the existence of subsequences of $\overline{w}_n$ and w_n, which converge in suitable function spaces towards some function w (see Theorem 4.1). Further, in Theorem 4.2, we improve the result concerning the convergence of a subsequence of w_n and show the relation between w and u, namely, $w = \theta(u)$. Theorem 4.3 proves that u is a weak solution to the IBVP (1.1).

The relative compactness of $\{w_n\}, \{\overline{u}_n\}$ are guaranteed by the following theorem.

THEOREM 4.1. *Let the assumptions of Lemma* 3.1 *be satisfied. Then*

(i) *there exists a function* $w \in C(I, H^{-1}(\Omega)) \cap L_\infty(I, L_2(\Omega))$ *with* $\partial_t w \in L_2(I, H^{-1}(\Omega))$ *and a subsequence of* $\{w_n\}$ *(denoted again by the same symbol) for which*

$$\begin{array}{rll} w_n \to w & \text{in } C(I, H^{-1}(\Omega)), & \\ w_n(t) \rightharpoonup w(t) & \text{in } L_2(\Omega), & \forall t \in [0,T], \\ \overline{w}_n(t) \rightharpoonup w(t) & \text{in } L_2(\Omega), & \forall t \in [0,T], \\ \partial_t w_n \rightharpoonup \partial_t w & \text{in } L_2(I, H^{-1}(\Omega)); & \end{array}$$

(ii) *there exists a function* $u \in L_2(I, H_0^1(\Omega))$ *and a subsequence of* $\{\overline{u}_n\}$ *(denoted by the same symbol again) for which*

$$\overline{u}_n \rightharpoonup u \quad \text{in } L_2(I, H_0^1(\Omega)).$$

Proof. (i) The assertion follows from (4.2) and Kačur [18, Lemma 1.3.13, p. 25].

(ii) The reflexivity of the space $L_2(I, H_0^1(\Omega))$ and the a priori estimate

$$\int_0^T \|\nabla \overline{u}_n(t)\|^2 \mathrm{d}t \leq C$$

imply the desired result. □

The next step is to prove relative compactness of $\{w_n\}$ in $L_2(Q_T)$.

THEOREM 4.2. *Let the assumptions of Lemma* 3.1 *be satisfied. Then*

(i) *there exists a subsequence of* $\{w_n\}$ *(denoted by the same symbol again) for which*

$$w_n \to w \quad \textit{in } L_2(Q_T);$$

(ii) $w = \theta(u)$*, where the function* u *is given by Theorem* 4.1.

Proof. (i) Let $w_n(t, \boldsymbol{x}) = 0$ if $t \notin [0, T]$ or $\boldsymbol{x} \notin \Omega$. According to the Kolmogorov argument (cf. Kufner, John, and Fučík [20, p. 88]), it is sufficient to prove that

1. $\int_0^T \int_\Omega |w_n(t, \boldsymbol{x})|^2 \mathrm{d}\boldsymbol{x}\mathrm{d}t \leq C$ for all n;
2. $\int_0^T \int_\Omega |w_n(t+s, \boldsymbol{x}+\boldsymbol{h}) - w_n(t, \boldsymbol{x})|^2 \mathrm{d}\boldsymbol{x}\mathrm{d}t \to 0$ for $s, |\boldsymbol{h}| \to 0$ uniformly with respect to n.

The first assertion is a consequence of a priori estimates (4.2). Further, one can deduce by a simple calculation that

$$\begin{aligned}
&\int_0^T \int_\Omega |w_n(t+s, \boldsymbol{x}+\boldsymbol{h}) - w_n(t, \boldsymbol{x}+\boldsymbol{h})|^2 \mathrm{d}\boldsymbol{x}\mathrm{d}t \\
&\quad = \int_0^T \int_t^{t+s} (\partial_t w_n(y, \boldsymbol{x}+\boldsymbol{h}), w_n(t+s, \boldsymbol{x}+\boldsymbol{h}) - w_n(t, \boldsymbol{x}+\boldsymbol{h}))\mathrm{d}y\mathrm{d}\boldsymbol{x}\mathrm{d}t \\
&\quad = \int_0^T \int_0^{s} (\partial_t w_n(t+y, \boldsymbol{x}+\boldsymbol{h}), w_n(t+s, \boldsymbol{x}+\boldsymbol{h}) - w_n(t, \boldsymbol{x}+\boldsymbol{h}))\mathrm{d}y\mathrm{d}\boldsymbol{x}\mathrm{d}t \\
&\quad \leq \int_0^s \int_0^T \|\partial_t w_n(t+y)\|_{-1}(\|w_n(t+s\|_1 + \|w_n(t)\|_1)\mathrm{d}y\mathrm{d}t \\
&\quad \leq C \int_0^s \sqrt{\int_0^T \|\partial_t w_n(t)\|_{-1}^2 \mathrm{d}t} \sqrt{\int_0^T \|w_n(t)\|_1^2 \mathrm{d}t} \\
&\quad \leq Cs.
\end{aligned}$$

Using $\int_0^T \|\overline{w}_n(t)\|_1^2 \mathrm{d}t \leq C$ we estimate

$$\begin{aligned}
\int_0^T \int_\Omega |w_n(t, \boldsymbol{x}+\boldsymbol{h}) - w_n(t, \boldsymbol{x})|^2 \mathrm{d}\boldsymbol{x}\mathrm{d}t &\leq C \int_0^T \int_\Omega |\overline{w}_n(t, \boldsymbol{x}+\boldsymbol{h}) - \overline{w}_n(t, \boldsymbol{x})|^2 \mathrm{d}\boldsymbol{x}\mathrm{d}t \\
&\leq \omega(|\boldsymbol{h}|),
\end{aligned}$$

where ω is a continuous real function with $\lim_{s \to 0_+} \omega(s) = 0$ (see, e.g., Nečas [25]).

The second assertion follows from these estimates and the triangle inequality. Hence the sequence $\{w_n\}$ is relatively compact in $L_2(Q_T)$. According to Theorem 4.1 we conclude the proof of (i).

(ii) Using (i) and the a priori estimate $\int_0^T \|w_n(t) - \overline{w}_n(t)\|^2 \mathrm{d}t \leq \frac{C}{n}$, we get

$$\overline{w}_n \to w \quad \text{in } L_2(Q_T). \tag{4.3}$$

The compact embedding $L_2(I, H_0^1(\Omega)) \hookrightarrow\hookrightarrow L_2(I, L_2(\Omega))$ and the weak convergence $\overline{u}_n \rightharpoonup u$ in $L_2(I, H_0^1(\Omega))$ (see Theorem 4.1) imply

$$\begin{aligned}\int_0^T \|\overline{w}_n(t) - \theta(u(t))\|^2 \mathrm{d}t &= \int_0^T \|\theta(\overline{u}_n(t)) - \theta(u(t))\|^2 \mathrm{d}t \\ &\le C \int_0^T \|\overline{u}_n(t) - u(t)\|^2 \mathrm{d}t \quad \to 0 \quad \text{as } n \to \infty.\end{aligned}$$

Hence $w = \theta(u)$. □

Now we use the convergence results from Theorems 4.1 and 4.2 and the a priori estimates (4.2) to examine the limit as $n \to \infty$ in (4.1).

THEOREM 4.3. *Let the assumptions of Lemma* 3.1 *be satisfied. Then the function u given by Theorem* 4.1 *is a weak solution to the IBVP* (1.1) *in the sense of Definition* 1.1.

Proof. In view of Theorems 4.1 and 4.2, it is sufficient to examine the limit for $n \to \infty$ in the relation (4.1).

The idea of the proof is the following. First, we integrate (4.1) over $(0, t)$ for any $t \in (0, T)$. Then we apply the a priori estimates (4.2) and Theorems 4.1 and 4.2 in order to examine the limit as $n \to \infty$. Finally, after the differentiating with respect to the time variable t, we conclude the proof. We demonstrate this for the second term of (4.1), only. This part is the most complicated. The rest can be shown in a standard way.

Smooth functions are dense in $H_0^1(\Omega)$. Thus, for any $\varepsilon \in \mathbb{R}_+$ and any $\varphi \in H_0^1(\Omega)$, there exists a $\varphi_\varepsilon \in C_0^\infty(\overline{\Omega})$ such that $\|\varphi - \varphi_\varepsilon\|_1 < \varepsilon$.

Therefore, we can write

$$\begin{aligned}&\left| \lim_{n\to\infty} \int_0^t (K(\overline{w}_n(s-\tau))\nabla \overline{u}_n(s), \nabla\varphi)\mathrm{d}s - \int_0^t (K(\theta(u(s))\nabla u(s), \nabla\varphi)\mathrm{d}s \right| \\ &\quad = \left| \lim_{n\to\infty} \int_0^t (K(\overline{w}_n(s-\tau))\nabla \overline{u}_n(s) - K(\theta(u(s))\nabla u(s), \nabla[\varphi - \varphi_\varepsilon])\mathrm{d}s \right. \\ &\qquad \left. + \lim_{n\to\infty} \int_0^t (K(\overline{w}_n(s-\tau))\nabla \overline{u}_n(s) - K(\theta(u(s))\nabla u(s), \nabla\varphi_\varepsilon)\mathrm{d}s \right| \\ &\quad \le C\varepsilon + \left| \lim_{n\to\infty} \int_0^t (K(\overline{w}_n(s-\tau))\nabla \overline{u}_n(s) - K(\theta(u(s))\nabla u(s), \nabla\varphi_\varepsilon)\mathrm{d}s \right|.\end{aligned}$$

The Cauchy–Schwarz and the triangle inequalities imply for the second term

$$\begin{aligned}&\left| \int_0^t ([K(\overline{w}_n(s-\tau)) - K(\theta(u(s)))]\nabla \overline{u}_n(s), \nabla\varphi_\varepsilon)\mathrm{d}s \right| \\ &\quad \le \|\nabla\varphi_\varepsilon\|_{L_\infty(\Omega)} \sqrt{\int_0^T \|\nabla \overline{u}_n(s)\|^2 \mathrm{d}s} \sqrt{\int_0^T \|K(\overline{w}_n(s-\tau)) - K(\theta(u(s)))\|^2 \mathrm{d}s} \\ &\quad \le C\|\nabla\varphi_\varepsilon\|_{L_\infty(\Omega)} \sqrt{\int_0^T \|\overline{w}_n(s-\tau) - \theta(u(s))\|^2 \mathrm{d}s} \\ &\quad \le C\|\nabla\varphi_\varepsilon\|_{L_\infty(\Omega)} \sqrt{\int_0^T [\|\overline{w}_n(s-\tau) - \overline{w}_n(s)\|^2 + \|\overline{w}_n(s) - \theta(u(s))\|^2]\,\mathrm{d}s}.\end{aligned}$$

Hence, in light of (4.2) and (4.3), we get

$$\lim_{n\to\infty} \left| \int_0^t ([K(\overline{w}_n(s-\tau)) - K(\theta(u(s)))]\nabla \overline{u}_n(s), \nabla\varphi_\varepsilon)\mathrm{d}s \right| = 0.$$

Therefore, we arrive at

$$\left|\lim_{n\to\infty}\int_0^t (K(\overline{w}_n(s-\tau))\nabla\overline{u}_n(s),\nabla\varphi)\mathrm{d}s - \int_0^t (K(\theta(u(s))\nabla u(s),\nabla\varphi)\mathrm{d}s\right| \le C\varepsilon,$$

which is valid for all $\varepsilon \in \mathbb{R}_+$. Passing to the limit for $\varepsilon \to 0_+$, we conclude

$$\lim_{n\to\infty}\int_0^t (K(\overline{w}_n(s-\tau))\nabla\overline{u}_n(s),\nabla\varphi)\mathrm{d}s = \int_0^t (K(\theta(u(s))\nabla u(s),\nabla\varphi)\mathrm{d}s. \qquad \square$$

Up until now, we have proved the convergence for a subsequence only. The convergence of the whole sequence follows from the uniqueness of the solution.

Consider for a moment the following nonlinear scheme for $i = 1, \dots, n$:

$$\begin{aligned} v_0 &= u_0, \\ (\delta\theta(v_i),\varphi) + (K(\theta(v_{i-1}))\nabla v_i,\nabla\varphi) &= (f_i,\varphi) + (K(\theta(v_{i-1}))\boldsymbol{g},\nabla\varphi) \end{aligned} \tag{4.4}$$

for all $\varphi \in H_0^1(\Omega)$, which has been studied in this or a simplified form by many authors. Also some error estimates in various functional spaces have been derived, depending on the regularity of u_0 and the data functions. When comparing linearization scheme (4.1) with (4.4) we see that our approximate solution $u_{i,\kappa}$ can be arbitrary close to v_i. This follows from the convergence of the relaxation iterations at any time point of the time partitioning. Hence, if the rate of convergence for (4.4) is $\mathcal{O}(\tau^\alpha)$, then (4.1) has the same precision considering sufficiently large d. Thus we have not lost anything in the linearization process.

Up until now, we have presented a new linear relaxation scheme for a uniform time partitioning. The reason was to keep the proofs easier for the reader. Clearly, the relaxation scheme can also be applied for nonequidistant time steps, and the relaxation iterations will converge at any t_i.

5. Numerical experiments.

In this section we present some numerical examples in order to demonstrate the efficiency and robustness of the proposed linearization scheme (1.10). The first one is devoted to a nonlinear IBVP of the type (1.1) without the convection term. Here an exact solution is known. The second example shows the robustness and the efficiency of the relaxation iterations, which is demonstrated on a nonlinear elliptic BVP. The third example is devoted to the real applications. Here we present two different models for description of the soil properties.

For two-dimensional examples, we have chosen the mixed nonconforming finite elements on a uniform triangular mesh for the space discretization. Recall that these are equivalent to the mixed-hybrid method (see, e.g., Arnold and Brezzi [4]). One-dimensional examples have been solved by the piecewise linear Galerkin finite element method.

5.1. Hornung–Messing problem.

We consider a horizontal flow (independent of the z-coordinate) in the domain $\Omega = (0,1)\times(0,1)$, which is given by the following IBVP:

$$\begin{aligned} \partial_t b(p) - \nabla\cdot(K_b(p)\nabla p) &= 0 \quad && \text{in } (0,1)\times\Omega, \\ p &= u \quad && \text{in } (0,1)\times\Gamma, \\ p(0) &= u(0) \quad && \text{in } \Omega, \end{aligned} \tag{5.1}$$

where the nonlinear functions b and K_b are defined as

$$b(p) := \begin{cases} \dfrac{\pi^2}{2} - 2\arctan^2(p) & \text{for } p < 0, \\ \dfrac{\pi^2}{2} & \text{for } p \geq 0 \end{cases} \tag{5.2}$$

and

$$K_b(p) := \begin{cases} \dfrac{2}{1+p^2} & \text{for } p < 0, \\ 2 & \text{for } p \geq 0. \end{cases} \tag{5.3}$$

The functions $b(p)$ and $K(b) := K_b$ have typical shapes; cf. Figure 1.1. Their derivatives are depicted in Figure 5.1. We have chosen the Lipschitz constant L for the function b as $L = 2$ in our computations.

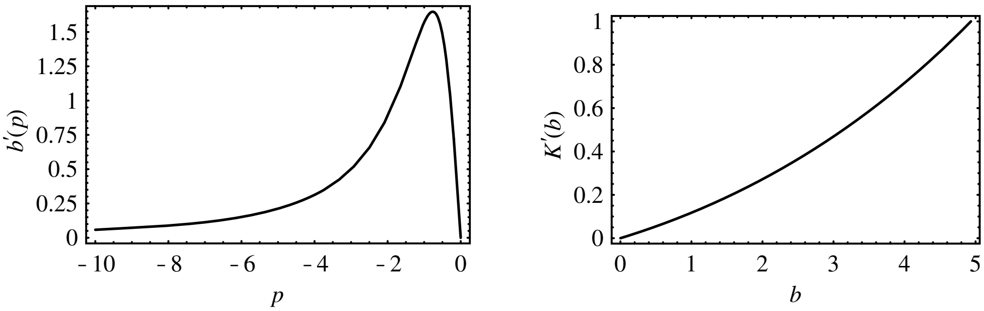


FIG. 5.1. *Behavior of the functions $b'(p)$ and $K'(b)$.*

The function u is given as

$$u(x,y,t) := \begin{cases} -\dfrac{s}{2} & \text{for } s < 0, \\ -\tan\left(\dfrac{e^s-1}{e^s+1}\right) & \text{for } s \geq 0, \end{cases} \tag{5.4}$$

where $s = x - y - t$.

It can be checked that the exact solution of the IBVP (5.1)–(5.4) is $p = u$.

The behavior of the function $b(p(0,x,y))$ is shown in Figure 5.2.

The doubly nonlinear degenerate IBVP (5.1)–(5.4) has been solved using the numerical scheme (1.10). We have computed eight relaxation iterations at each time step, i.e., $\kappa = 1, 2, \dots, 8$. We have used a uniform triangular mesh with $h = \Delta x = \Delta y = 0.04$ or $h = \Delta x = \Delta y = 0.02$. The time step τ takes one of the values $\tau = 0.2$, 0.1, or 0.05. Figure 5.3 represents the error in $L_2(\Omega)$-norm between the discrete and exact solutions. The pictures for $h = 0.04$ and $h = 0.02$ hardly vary with changing h.

Here, at first glance, an interesting effect appears where larger time steps (for later time values) can give smaller error. This can be explained using Lemma 3.1. We have proved there that for any positive τ the relaxation iterations converge. However, if we compare the error bounds for a fixed number of relaxation iterations and for different values of the time steps $\tau_1 < \tau_2$, then, in fact, it can happen that the error corresponding to τ_2 can be less than the error for τ_1. This effect can be removed when we increase the number of relaxation iterations for τ_1; see the next example.

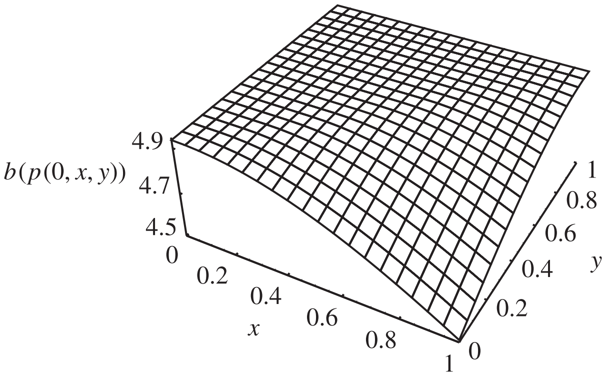


FIG. 5.2. *Behavior of* $b(p(0,x,y))$.

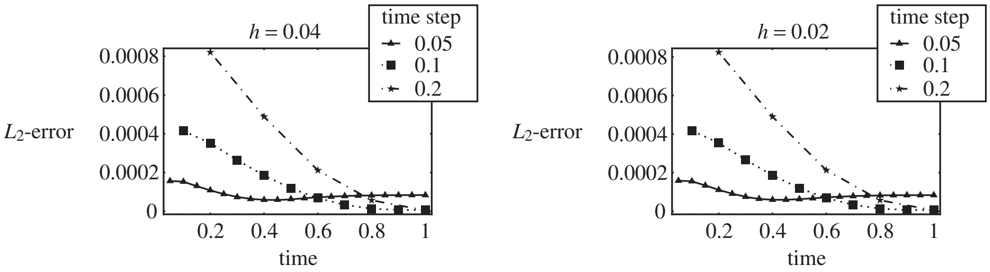


FIG. 5.3. $L_2(\Omega)$*-error for the space step* $h = 0.04$ (*respectively,* $h = 0.02$) *and the time steps* $\tau = 0.2$, 0.1, 0.05.

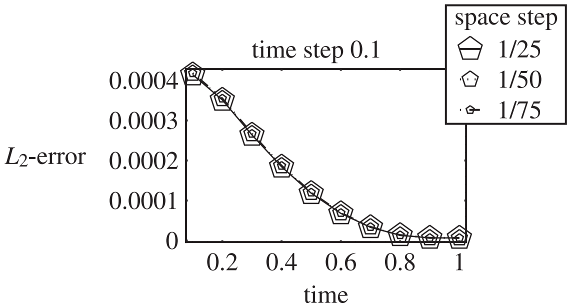


FIG. 5.4. $L_2(\Omega)$*-error for the space steps* $h = \frac{1}{25}, \frac{1}{50}, \frac{1}{75}$, *the time step* $\tau = 0.1$, *and eight relaxation iterations.*

Now we fix the time step $\tau = 0.1$ and the number of relaxation iterations $\kappa_{i,\text{last}} = 8$. Figure 5.4 shows the behavior of the error in the $L_2(\Omega)$-norm versus the time for various space steps. The graph indicates that the error remains the same with the decreasing h. Thus the error coming from the time discretization and/or the linearization is still dominant with respect to the error of the space discretization.

5.2. Nonlinear elliptic BVP. Let $\Omega = (0,1) \times (0,1)$ be the unit square in $\mathbb{R}^2$. The nonlinear function θ is defined as

$$\theta(s) = \begin{cases} \arctan(s) & \text{for } s < 1, \\ \dfrac{\pi}{4} & \text{elsewhere.} \end{cases}$$

This is clearly continuous. For the derivative, we have

$$\theta'(s) = \begin{cases} \dfrac{1}{1+s^2} & \text{for } s < 1, \\ 0 & \text{elsewhere;} \end{cases}$$

thus $0 \le \theta' \le 1$ a.e. in $\mathbb{R}$.

We consider the following nonlinear elliptic BVP: Find $u \in H^1(\Omega)$ such that

$$\begin{aligned} \theta(v) - \Delta v = f &\quad \text{in } \Omega, \\ v = w &\quad \text{on } \Gamma, \end{aligned}$$

where the right-hand side f is defined in such a way that the exact solution of this BVP is

$$v(x,y) = w(x,y) := x^3 - y^2 + x + \sin(\pi x)\sin(\pi y).$$

We have used the linearization scheme (3.1) with $L = K = \tau = 1$, $F = f$, and $\boldsymbol{g} = \mathbf{0}$ for computations. We have started the relaxation iterations from v_0, which was far away from the exact solution v. More exactly, we have chosen

$$v_0(\boldsymbol{x}) = 100\text{ran}(\boldsymbol{x}),$$

where ran is a real random function, the range of which is uniformly distributed over the interval $(-1, 1)$.

We have used a uniform triangular mesh consisting of 5000 elements corresponding to $\Delta x = \Delta y = 0.02$, and we have computed 25 relaxation iterations. The results are depicted in Figures 5.5 and 5.6.

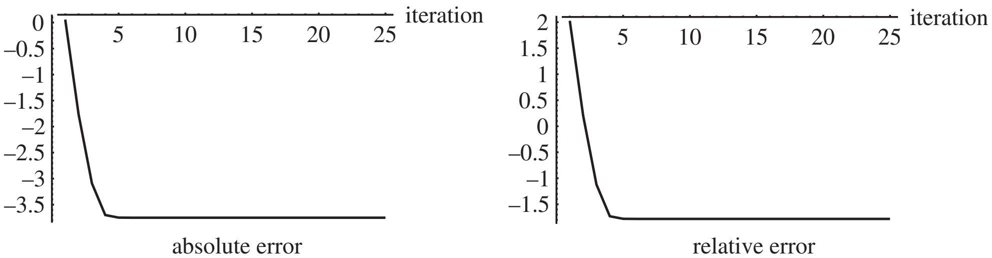


FIG. 5.5. *Logarithm of the norm* $\|v_\kappa - w\|$ *versus iterations.*

Conclusion. The graphs in Figures 5.5 and 5.6 are similar. The rapidly decreasing part is followed by a constant section. At the beginning of the iteration process the linearization error was dominant. Later the discretization error becomes superior. The more or less constant part of a graph means that the discretization error has been achieved.

The proposed linearization scheme is robust, and the approximations converge towards the exact solution independently of the fact of where the iteration process started. This is a big difference from other frequently used algorithms. The iterations are efficient (fast) if $\frac{\tau\lambda}{L+\tau\lambda}$ is not close to the 1 (for an elliptic problem we put $\tau = 1$).

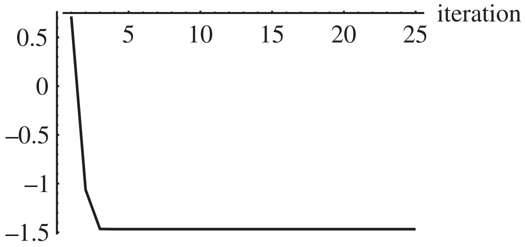


FIG. 5.6. *Logarithm of the norm* $\|\nabla(v_\kappa - w)\|$ *versus iterations.*

5.3. Richards' equation. The standard form of the unsaturated flow equation in one dimension is written as

$$\partial_t\theta(h) - \partial_z(K(h)\partial_z h) - \partial_z K(h) = 0. \tag{5.5}$$

We consider a sand column of a given length with Dirichlet boundary conditions and an initial datum. We consider two models for the representation of the soil properties, i.e., the functions θ and K. We recall that there is no analytical solution known for such a problem. We compute the approximate solution and compare the result with Celia, Bouloutas, and Zarba [11].

We use the linearization scheme (1.10) for computations. The relaxation iterations stops when the discrete $L_2(\Omega)$-norm of $u_{i,\kappa} - u_{i,\kappa-1}$ becomes less than 10^{-3}. The space discretization is based on the standard piecewise linear Galerkin finite element method.

5.3.1. Haverkamp–Celia model. We consider a soil column of length 40cm. Boundary conditions are $h(t,40) = h_{\text{bottom}} = -61.5$cm and $h(t,0) = h_{\text{top}} = -20.7$cm. Initial condition is given as $h(0,z) = h_{\text{bottom}}$. The soil properties expressed by the functions θ and K are

$$\begin{aligned} \theta(h) &= \frac{\alpha(\theta_s - \theta_r)}{\alpha + |h|^\beta} + \theta_r, \\ K(h) &= K_s\frac{A}{A + |h|^\gamma}, \end{aligned} \tag{5.6}$$

where $\alpha = 1.611 \times 10^6$, $\theta_s = 0.287$, $\theta_r = 0.075$, $\beta = 3.96$, $K_s = 0.00944\frac{\text{cm}}{\text{s}}$, $A = 1.175\times10^6$, and $\gamma = 4.74$. These data are taken from Celia, Bouloutas, and Zarba [11]. After the analysis of the behavior of θ' on $[h_{\text{bottom}}, h_{\text{top}}]$, we have chosen the Lipschitz constant L for the function θ as $L = 0.0034$ in (1.10).

Two pressure head profiles at an elapsed time of 360s corresponding to $\tau = 1$s, $\Delta z = 1$mm, and $\tau = 10$s, $\Delta z = 1$cm are shown in Figure 5.7. Our results conform the ones from Celia, Bouloutas, and Zarba [11].

5.3.2. Van Genuchten model. Let the relevant material properties θ and K be given by van Genuchten model (1.9), where $\alpha = 0.0335\frac{1}{\text{cm}}$, $\theta_s = 0.368$, $\theta_r = 0.102$, $n = 2$, $m = 0.5$, and $K_s = 0.00922\frac{\text{cm}}{\text{s}}$. These data are taken from Celia, Bouloutas, and Zarba [11]. We consider a soil column of the depth 100cm. Boundary conditions are $h(t,100) = h_{\text{bottom}} = -1000$cm and $h(t,0) = h_{\text{top}} = -75$cm. The initial condition is given as $h(0,z) = h_{\text{bottom}}$. We have taken the time step $\tau = 10$s and the space step $\Delta z = 1$cm for computations. After the analysis of the behavior of θ' on $[h_{\text{bottom}}, h_{\text{top}}]$, we have chosen the Lipschitz constant $L = 0.001133$ in our computations.

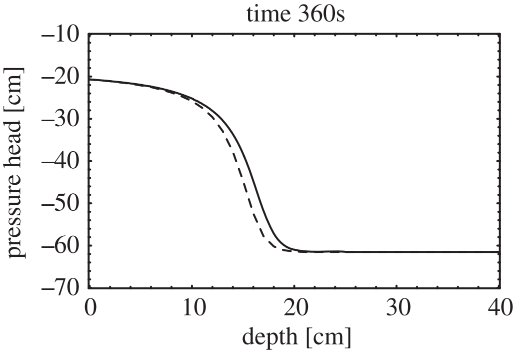


FIG. 5.7. *Solid line: Pressure head for $\tau = 1$s, $\Delta z = 1$mm. Dashed line: Pressure head for $\tau = 10$s, $\Delta z = 1$cm.*

Figure 5.8 shows the solution profiles at the times $t = 3$h, $t = 6$h, $t = 12$h, and $t = 24$h. The solution profile at $t = 24$h corresponds very well to those of Celia, Bouloutas, and Zarba [11] which have been computed by various numerical schemes. Figure 5.8 shows also the oscillations ahead of the moisture front. This is an typical effect of the piecewise linear finite element method which has also been observed by other numerical schemes; see Celia, Bouloutas, and Zarba [11]. This effect appears where the change of the gradient of the solution is large.

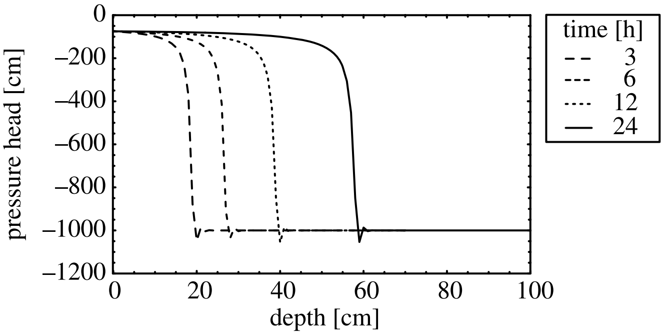


FIG. 5.8. *Solution profiles.*

REFERENCES

[1] H. Alt and S. Luckhaus, *Quasilinear elliptic-parabolic differential equations*, Math. Z., 183 (1983), pp. 311–341.

[2] G. Amiez and P. Gremaud, *On a numerical approach to Stefan-like problems*, Numer. Math., 59 (1991), pp. 71–89.

[3] T. Arbogast, M.-F. Wheeler, and N.-Y. Zhang, *A nonlinear mixed finite element method for a degenerate parabolic equation arising in flow in porous media*, SIAM J. Numer. Anal., 33 (1996), pp. 1669–1687.

[4] D. Arnold and F. Brezzi, *Mixed and nonconforming finite element methods: Implementation, postprocessing and error estimates*, RAIRO Modél. Math. Anal. Numér., 19 (1985), pp. 7–32.

[5] J. W. Barrett and P. Knabner, *Finite element approximation of the transport of reactive solutes in porous media, Part* II: *Error estimates for equilibrium adsorption processes*, SIAM J. Numer. Anal., 34 (1997), pp. 455–479.

[6] H. Brezis, *On some degenerate non-linear parabolic equations*, in Nonlinear Functional Analysis,, F. Browder, ed., AMS, Providence, RI, 1970, pp. 28–38.

[7] H. Brézis, *Operateurs maximaux monotones et semi-groupes de contractions dans les espaces de Hilbert*, North-Holland Math. Stud. 5, Notas de Matematica 50, North-Holland, Amsterdam, London, American Elsevier, New York, 1973.

[8] R. Brooks and A. Corey, *Hydraulic Properties of Porous Media*, Hydrology paper 3, Civil Engineering Department, Colorado State University, Fort Collins, CO, 1964.

[9] W. Brutsaert, *Probability laws for pore-size distribution*, Soil Sci., 101 (1966), pp. 85–92.

[10] A. Bumb, C. Murphy, and L. Everett, *A comparison of three functional forms for representing soil moisture characteristics*, Ground Water, 30 (1992), pp. 177–185.

[11] M. Celia, E. Bouloutas, and R. Zarba, *A general mass-conservative numerical solution for the unsaturated flow equation*, Water Resour. Res., 26 (1990), pp. 1483–1496.

[12] M. Crandall and T. Liggett, *Generation of semigroups of nonlinear transformations on general Banach spaces*, Amer. J. Math., 93 (1971), pp. 265–298.

[13] R. Eymard, M. Gutnic, and D. Hilhorst, *The finite volume method for richards equation*, Comput. Geosci., 3 (1999), pp. 259–294.

[14] D. Gilbarg and N. S. Trudinger, *Elliptic Partial Differential Equations of Second Order*, Springer-Verlag, Berlin, Heidelberg, 1983.

[15] W. Jäger and J. Kačur, *Solution of porous medium type systems by linear approximation schemes*, Numer. Math., 60 (1991), pp. 407–427.

[16] W. Jäger and J. Kačur, *Solution of doubly nonlinear and degenerate parabolic problems by relaxation schemes*, RAIRO Modél. Math. Anal. Numér., 29 (1995), pp. 605–627.

[17] J. Jones and C. Woodward, *Newton-Krylov-multigrid solvers for large-scale, highly heterogeneous, variably saturated flow problems*, Adv. Water Resour., 24 (2001), pp. 763–774.

[18] J. Kačur, *Method of Rothe in Evolution Equations*, Teubner-Texte Math. 80, Teubner, Leipzig, 1985.

[19] J. Kačur, *Solution to strongly nonlinear parabolic problems by a linear approximation scheme*, IMA J. Numer. Anal., 19 (1999), pp. 119–145.

[20] A. Kufner, O. John, and S. Fučík, *Function Spaces*, Academia, Prague, 1977.

[21] M. Křížek and P. Neittaanmäki, *Mathematical and Numerical Modelling in Electrical Engineering Theory and Applications*, Math. Model. Theory Appl. 1, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1996.

[22] E. Magenes, R. Nochetto, and C. Verdi, *Energy error estimates for a linear scheme to approximate nonlinear parabolic equations*, RAIRO Modél. Math. Numér., 21 (1987), pp. 655–678.

[23] C. Miller and C. Kelley, *A comparison of strongly-convergent solution schemes for sharp-front infiltration problems*, in Computational Methods in Water Resources X, A. Peters, G. Wittum, B. Herrling, U. Meissner, C. Brebbia, W. Gray, and G. Pinder, eds., Kluwer Academic Publishers, Dordrecht, The Netherlands, 1994, pp. 325–332.

[24] Y. Mualem, *A new model for predicting the hydraulic conductivity of unsaturated porous media*, Water Resour. Res., 12 (1976), pp. 513–522.

[25] J. Nečas, *Les méthodes directes en théorie des équations elliptiques*, Academia, Prague, 1967.

[26] R. Nochetto, *Error estimates for multidimensional Stefan problems with general boundary conditions*, in Free Boundary Problems: Applications and Theory, Vol. III, A. A. Bossavit and M. Fremond, eds., Res. Notes Math. 120, Pitman, Boston, 1985, pp. 50–60.

[27] R. Nochetto, *Error estimates for two-phase Stefan problems in several space variables* I: *Linear boundary conditions*, Calcolo, 22 (1985), pp. 457–499.

[28] F. Otto, *L^1-contraction and uniqueness for quasilinear elliptic-parabolic equations*, J. Differential Equations, 131 (1996), pp. 20–38.

[29] C. Pao, *Nonlinear Parabolic and Elliptic Equations*, Plenum Press, New York, 1992.

[30] I. Pop, *Regularization Methods in the Numerical Analysis of Some Degenerate Parabolic Equations*, Preprint 98-43(SFB 359), IWR, Universität Heidelberg, Heidelberg, Germany, 1998.

[31] I. Pop and W.-A. Yong, *A maximum principle based numerical approach to porous medium equations*, in ALGORITMY'97, 14th Conference on Scientific Computing, A. Handlovičová, M. Komorníková, and K. Mikula, eds., Faculty of Civil Engineering, Department of Mathematics and Descriptive Geometry, STU Bratislava, Bratislava, Slovakia, 1997, pp. 207–218.

[32] J. Rulla, *Error analysis for implicit approximations to solutions to Cauchy problems*, SIAM J. Numer. Anal., 33 (1996), pp. 68–87.

[33] J. Rulla and N. J. Walkington, *Optimal rates of convergence for degenerate parabolic problems in two dimensions*, SIAM J. Numer. Anal., 33 (1996), pp. 56–67.

[34] M. Slodička, *Solution of Nonlinear Parabolic Problems by Linearization*, Preprint M3-92, Fac-

ulty of Mathematics, Physics and Informatics, Comenius University, Bratislava, Slovakia, 1992.

[35] M. Slodička, *Some Finite Elements Schemes Arising in Modeling of Flow Through Porous Media*, Habilitationsschrift, Mathematisch-Naturwissenschaftlichen Fakultät, Universität Augsburg, Augsburg, Germany, 1999.

[36] M. Tocci, C. Kelley, C. Miller, and C. Kees, *Inexact Newton methods and the method of lines for solving Richard's equation in two space dimensions*, Comput. Geosci., 2 (1998), pp. 291–309.

[37] C. J. van Duijn and L. A. Peletier, *Nonstationary filtration in partially saturated porous media*, Arch. Rational Mech. Anal., 78 (1982), pp. 173–198.

[38] M. van Genuchten, *A closed-form for predicting the hydraulic conductivity of unsaturated soils*, Soil Sci. Soc. Amer. J., 44 (1980), pp. 892–898.

[39] M. Vauclin, D. Khanji, and G. Vachaud, *Experimental and numerical study of a transient two dimensional unsaturated-saturated water table recharge problem*, Water Resour. Res., 15 (1979), pp. 1089–1101.

[40] C. Verdi, *On the numerical approach to a two-phase Stefan problem with nonlinear flux*, Calcolo, 22 (1985), pp. 351–381.

[41] J. Wang and C. Pao, *Finite difference reaction-diffusion equations with nonlinear diffusion coefficients*, Numer. Math., 85 (2000), pp. 485–502.

[42] C. S. Woodward and C. N. Dawson, *Analysis of expanded mixed finite element methods for a nonlinear parabolic equation modeling flow into variably saturated porous media*, SIAM J. Numer. Anal., 37 (2000), pp. 701–724.

SIAM J. SCI. COMPUT.
Vol. 23, No. 5, pp. 1615–1654

COMPUTATION OF THE SHALLOW WATER EQUATIONS USING THE UNIFIED COORDINATES*

W. H. HUI[†] AND S. KOUDRIAKOV[†]

Abstract. Two general coordinate systems have been used extensively in computational fluid dynamics: the Eulerian and the Lagrangian. The Eulerian coordinates cause excessive numerical diffusion across flow discontinuities, slip lines in particular. The Lagrangian coordinates, on the other hand, can resolve slip lines sharply but cause severe grid deformation, resulting in large errors and even breakdown of the computation. Recently, in the spirit of the arbitrary Lagrangian–Eulerian (ALE) approach, W.H. Hui, P.Y. Li, and Z.W. Li, [J. Comput. Phys., 153 (1999), pp. 596–637] have introduced a unified coordinate system which moves with velocity $h\mathbf{q}$, $\mathbf{q}$ being the velocity of the fluid particle. It includes the Eulerian system as a special case when $h = 0$, and the Lagrangian when $h = 1$, and was shown for the two-dimensional Euler equations of gas dynamics to be superior to both Eulerian and Lagrangian systems. The main purpose of this paper is to adopt this unified coordinate system to solve the shallow water equations. It will be shown that computational results using the unified system are superior to existing results based on either the Eulerian system or Lagrangian system in that it (a) resolves slip lines sharply, especially for steady flow, (b) avoids grid deformation and computation breakdown in Lagrangian coordinates, and (c) avoids spurious flow produced by Lagrangian coordinates.

Key words. unified description, Eulerian description, Lagrangian description, two-dimensional shallow water equations, slip lines, spurious vorticity due to Lagrangian description

AMS subject classifications. 76A99, 76B15, 35L60

PII. S1064827500367415

1. Introduction.

Two general coordinate systems have been used extensively for describing fluid motion: the Eulerian and the Lagrangian. Computationally, each system has its advantages as well as disadvantages.

In using the Eulerian coordinates the computational cells are fixed in space, while fluid particles move across cell interfaces in any direction. It is this convective flux that causes excessive numerical diffusion in the numerical solution. Indeed, slip lines are smeared badly and shocks are also smeared, albeit somewhat better than slip lines. Moreover, the smearing of slip lines ever increases with time and distance unless special treatments, such as artificial compression or subcell resolution, are employed [2], [3], [4] which are, however, not always reliable. In this regard it is interesting to note that some works using the ghost fluid method [5], [6] based on the empirical isobaric fix have just appeared which give good slip line and shock resolution.

Computational cells in the Lagrangian coordinates, on the other hand, are literally fluid particles. Consequently, there is no convective flux across cell interfaces, and numerical diffusion is thus minimized. However, the very fact that the computational cells exactly follow fluid particles can result in severe grid deformation, causing inaccuracy and even breakdown of the computation. To prevent this from happening, the most famous Lagrangian method in use at the present time—the arbitrary Lagrangian–Eulerian (ALE) technique [8], [9], [10]—uses continuous rezoning

*Received by the editors February 8, 2000; accepted for publication (in revised form) June 18, 2001; published electronically January 30, 2002. The research of the first author was funded by the Research Grants Council of Hong Kong.

http://www.siam.org/journals/sisc/23-5/36741.html

[†]Department of Mathematics and Center for Scientific Computation, The Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong (whhui@ust.hk, hp1@semt2.smts.cea.fr).

and remapping to the Eulerian grid. Unfortunately, this process requires interpolations of geometry and flow variables which result in loss of accuracy, manifested as numerical diffusion which the ALE approach wants to avoid in the first place. Indeed, it was demonstrated in [11] that rezoning results in diffusive errors of the type encountered in Eulerian solutions and that continuously rezoned Lagrangian computation is equivalent to an Eulerian computation. Another disadvantage of the Lagrangian coordinates is that, except in the simple case of one-dimensional unsteady flow, the governing equations for inviscid flow are not easily written in conservation form, making it difficult to capture shocks correctly.

Recently, Hui, Li, and Li [1] have introduced a unified coordinate system which moves with velocity $h\mathbf{q}$, where $\mathbf{q}$ is velocity of the fluid particle. It includes the Eulerian coordinates as a special case when $h = 0$ and the Lagrangian when $h = 1$, and, more importantly, it has a new degree of freedom in choosing the arbitrary function h to improve the quality of computational results. In particular, it was shown in [1] that for the two-dimensional Euler equations of gas dynamics, choosing the function h to preserve grid angles results in a coordinate system which is superior to both Eulerian and Lagrangian systems.

The unified coordinates approach shares the same spirit of the ALE approach in that it combines the best features of the Lagrangian and Eulerian approach, [10, p. 198], or in that the coordinates move at an arbitrary velocity [7, p. 239]. However, the strategies are quite different. In the ALE approach, "the general strategy is to perform a Lagrangian time step and to follow it with a remap step that maps the solution from the distorted Lagrangian mesh on to the spatially fixed Eulerian mesh or the ALE mesh" [7, p. 236], [10, p. 198]. This is usually done by employing a staggered grid. Furthermore, the rezoning strategies are not generally prescribed; instead, "rezoning requires the intervention of the user, ..., and the success of the method depends heavily on the skill and patience of the user" [7, p. 322]. In our unified coordinates approach we propose a specific rule for grid (mesh) movement: the grid should move with velocity $h\mathbf{q}$, $\mathbf{q}$ being fluid velocity, and h is determined to preserve grid angles, resulting in sharp resolution of slip lines, especially for steady flow. In our method all computations are done entirely in the transformed space without a staggered grid and with no explicit rezoning/remapping to the Eulerian or ALE space; explicit remapping causes numerical diffusion.

The purpose of this paper is to adopt this unified coordinate approach to solve the shallow water equations; it will be shown that computational results using the unified system are superior to existing results based on either the Eulerian or Lagrangian system.

The paper is arranged as follows. In section 2 the shallow water equations in conservation form are derived using the unified coordinates. Sections 3 and 4 study the cases of one-dimensional and two-dimensional flow, respectively. Section 5 gives results for several test examples computed using the unified coordinates and compares them with Eulerian or Lagrangian computation. Finally, conclusions are given in section 6.

2. Shallow water equations in the unified coordinates. The shallow water equations in conservation form using Cartesian coordinates are

$$(1)\ \frac{\partial}{\partial t}\begin{pmatrix}\zeta\\ \zeta u\\ \zeta v\end{pmatrix}+\frac{\partial}{\partial x}\begin{pmatrix}\zeta u\\ \zeta u^2+\frac{1}{2}g\zeta^2\\ \zeta uv\end{pmatrix}+\frac{\partial}{\partial y}\begin{pmatrix}\zeta v\\ \zeta uv\\ \zeta v^2+\frac{1}{2}g\zeta^2\end{pmatrix}=\begin{pmatrix}0\\ \zeta\left[g\frac{\partial D}{\partial x}-m\right]\\ \zeta\left[g\frac{\partial D}{\partial y}-n\right]\end{pmatrix},$$

where g is the acceleration due to gravity, $\zeta(x, y, t)$ is the total water height measured from the bottom, $u(x, y, t)$ and $v(x, y, t)$ are the components of the fluid velocity in the horizontal x and y direction, respectively, $D(x, y)$ is the water depth from a fixed reference level, and m and n are the components of the bottom friction force due to its roughness.

Introduce a transformation of coordinates from (t, x, y) to (λ, ξ, η),

$$\left\{ \begin{array}{l} dt = d\lambda, \\ dx = hud\lambda + Ad\xi + Ld\eta, \\ dy = hvd\lambda + Bd\xi + Md\eta, \end{array} \right. \tag{2}$$

where h is arbitrary. Let

$$\frac{D_h}{Dt} \equiv \frac{\partial}{\partial t} + hu\frac{\partial}{\partial x} + hv\frac{\partial}{\partial y} \tag{3}$$

denote the time derivative following the *pseudoparticle*, whose velocity is $h\mathbf{q}$, $\mathbf{q} = (u, v)$. Then, under the assumption $AM - BL \neq 0$ (nonsingularity of the transformation (2)), it is easy to show

$$\frac{D_h\xi}{Dt} = 0, \quad \frac{D_h\eta}{Dt} = 0, \tag{4}$$

that is to say, the coordinates (ξ, η) are material functions of the pseudoparticles, hence they are their permanent identifications. Accordingly, computational cells move and deform with pseudoparticles, rather than with fluid particles as in Lagrangian coordinates.

Remarks.

(a) Unlike most transformations used in grid generation, which are flow-independent, the unique feature of transformation (2) is its dependence on the fluid velocity.

(b) In (2), h is an arbitrary function of coordinates (λ, ξ, η). On the other hand, (A, B, L, M) are determined by the compatibility conditions. For example, for dx to be a total differential,

$$\frac{\partial A}{\partial \lambda} = \frac{\partial (hu)}{\partial \xi}, \tag{5}$$

$$\frac{\partial L}{\partial \lambda} = \frac{\partial (hu)}{\partial \eta}. \tag{6}$$

When (5)–(6) are satisfied, the other compatibility condition, namely,

$$\frac{\partial A}{\partial \eta} = \frac{\partial L}{\partial \xi}, \tag{7}$$

is also satisfied, provided it is at $\lambda = 0$, which can always be ensured in numerical computation. Similar compatibility conditions hold for (B, M).

(c) In the special case when $h = 0$, (A, B, L, M) are independent of λ. Then the coordinates (ξ, η) are independent of time λ and are hence fixed in space. This coordinate system is thus Eulerian. Transformation (2) is then flow-independent and is just like any other transformation from Cartesian coordinates (x, y) to curvilinear coordinates (ξ, η) used in grid generation. In particular, if $A = M = 1$ and $L = B = 0$, (ξ, η) are identical with Cartesian coordinates (x, y).

(d) In the special case when $h = 1$, on the other hand, the pseudoparticles coincide with fluid particles and (ξ, η) are the material functions of fluid particles, hence they are Lagrangian coordinates. The conventional choice of the Lagrangian coordinates, i.e., $(\xi, \eta) = (x, y)|_{t=0}$, is just a special choice of material functions, corresponding to choosing $A = M = 1$ and $L = B = 0$. It does not offer any particular advantage in numerical computation; rather (ξ, η) should better be left to be suitably chosen to initialize numerical computation. In particular, the computational domain in the (ξ, η) space can always be easily made regular, e.g., rectangular, even if it is irregular in the physical space. This cannot be done with the conventional choice of the Lagrangian coordinates.

(e) In the general case, h is arbitrary. It has been shown [1] that the unified coordinates for $h \neq 0$ always yield sharp slip line resolution in steady flow. Furthermore, h may be chosen to advantage: to avoid excessive numerical diffusion in the Eulerian coordinates, and/or to avoid severe grid deformation in the Lagrangian coordinates.

Under the transformation (2) the shallow water equations (1) become

$$\frac{\partial \mathbf{E}}{\partial \lambda} + \frac{\partial \mathbf{F}}{\partial \xi} + \frac{\partial \mathbf{G}}{\partial \eta} = \mathbf{S}, \tag{8}$$

where

$$\mathbf{E} = \begin{pmatrix} \zeta\triangle \\ \zeta\triangle u \\ \zeta\triangle v \\ A \\ B \\ L \\ M \end{pmatrix}, \quad \mathbf{F} = \begin{pmatrix} \zeta(1-h)I \\ \zeta(1-h)Iu + \frac{1}{2}g\zeta^2 M \\ \zeta(1-h)Iv - \frac{1}{2}g\zeta^2 L \\ -hu \\ -hv \\ 0 \\ 0 \end{pmatrix},$$

$$\mathbf{G} = \begin{pmatrix} \zeta(1-h)J \\ \zeta(1-h)Ju - \frac{1}{2}g\zeta^2 B \\ \zeta(1-h)Jv + \frac{1}{2}g\zeta^2 A \\ 0 \\ 0 \\ -hu \\ -hv \end{pmatrix}, \quad \mathbf{S} = \begin{pmatrix} 0 \\ \zeta\left[gMD_\xi - gBD_\eta - m\triangle\right] \\ \zeta\left[-gLD_\xi + gAD_\eta - n\triangle\right] \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \tag{9}$$

with

$$\triangle = AM - BL, \quad I = uM - vL, \quad J = vA - uB. \tag{10}$$

We note that the shallow water equations (8) written in the unified coordinates are in conservation form. We also point out that although system (8) is larger than (1), computationally the extra computing time required for solving the last four equations of (8) is very small, typically 3–5%, because the bulk of computing time is spent on solving the Riemann problems for the first three equations of (8), which require the same amount of computing time as system (1).

In the remainder of this paper we shall consider only horizontal bottom and neglect the friction term there, and hence $\mathbf{S} = 0$.

3. One-dimensional shallow water flow. For the special case of one-dimensional flow, transformation (2) simplifies to

$$\left\{ \begin{array}{l} dt = d\lambda, \\ dx = hud\lambda + Ad\xi, \end{array} \right. \tag{11}$$

and the shallow water equations (8) become

$$\frac{\partial \mathbf{E}}{\partial \lambda} + \frac{\partial \mathbf{F}}{\partial \xi} = \mathbf{0}, \tag{12}$$

where

$$\mathbf{E} = \left(\begin{array}{c} \zeta A \\ \zeta A u \\ A \end{array} \right), \quad \mathbf{F} = \left(\begin{array}{c} \zeta(1-h)u \\ \zeta(1-h)u^2 + \frac{1}{2} g \zeta^2 \\ -hu \end{array} \right). \tag{13}$$

3.1. Hyperbolicity. As we shall use the Godunov-MUSCL scheme to compute system (12), we need to consider the hyperbolicity property of that system in order to ensure the existence and uniqueness of the solution to the Riemann problem, which is the main ingredient of the Godunov scheme.

It is well known that the one-dimensional shallow water equations in Eulerian coordinates are hyperbolic. However, since transformation (11) involves the dependent variable u, there is no guarantee that the resulting system (12) will necessarily be hyperbolic also; now we check this.

The eigenvalues of (12) can be found by direct computation, and the results are:

$$\sigma_1 = 0, \tag{14}$$

$$\sigma_{\pm} = \frac{(1-h)u \pm \sqrt{g\zeta}}{A}. \tag{15}$$

The corresponding right eigenvectors, when the primitive variables $\mathbf{U} = (\zeta, u, A)^T$ are used, are

$$\mathbf{r_1} = (0, 0, 1)^T, \tag{16}$$

$$\mathbf{r}_{\pm} = \left(1, \pm\sqrt{g/\zeta}, \mp\frac{h}{\sigma_{\pm}}\sqrt{g/\zeta} \right)^T. \tag{17}$$

It can easily be shown that the σ_1-field is linearly degenerate, while the $\sigma_{\pm}$-fields are genuinely nonlinear. The eigenvectors are linearly independent, forming a complete basis in the state space; system (12) is therefore hyperbolic for all values of h, despite the fact that transformation (11) involves the dependent variable u. This includes the Eulerian coordinates as a special case when $h = 0$ and the Lagrangian one when $h = 1$. This result is the same as one-dimensional Euler equations of gas dynamics [12].

3.2. Riemann problem. The one-dimensional shallow water equations (12) will be solved using the Godunov method with the MUSCL update to high resolution, of which the main ingredient is the solution of the Riemann problem.

The Riemann problem is

$$\left\{ \begin{array}{ll} \dfrac{\partial \mathbf{E}}{\partial \lambda} + \dfrac{\partial \mathbf{F}}{\partial \xi} = 0, & \lambda > 0, \\ \mathbf{E}(0, \xi) = \left\{ \begin{array}{ll} \mathbf{E}_l, & \xi < 0, \\ \mathbf{E}_r, & \xi > 0, \end{array} \right. & \end{array} \right. \tag{18}$$

where $\mathbf{E}_l$ and $\mathbf{E}_r$ are the constant vectors representing the flow states on the left and the right side, respectively. Here we shall consider the case when h is a constant in the range $0 \le h \le 1$. With $h = \text{const}$, (18) is a system of conservation law equations with constant coefficients and its solution depends on $\mu = \frac{\xi}{\lambda}$ alone; i.e., it is a self-similar solution of the form $\mathbf{E} = \mathbf{E}(\mu)$. It consists of at most four uniform flow regions, including $\mathbf{E}_l$ and $\mathbf{E}_r$, separated by three elementary waves: a shock (or expansion), a contact line, and an expansion (or shock). These elementary wave solutions are now given.

3.2.1. Expansion wave. The centered expansion wave solution from the $\sigma_\pm$ characteristic fields can be derived from the following system of ODEs:

$$\frac{du}{d\zeta} = \pm\sqrt{g/\zeta}, \tag{19}$$

$$\frac{dA}{d\zeta} = \mp\frac{h\sqrt{g/\zeta}}{\sigma_\pm}. \tag{20}$$

The solution for (u, A) relates the flow state $\mathbf{U} = (\zeta, u, A)^T$ in the expansion wave to the initial state $\mathbf{U_0} = (\zeta_0, u_0, A_0)^T$ upstream of the wave through the following expressions:

$$u \mp 2\sqrt{g\zeta} = u_0 \mp 2\sqrt{g\zeta_0}, \tag{21}$$

$$A = A_0\left(\frac{\sqrt{\zeta_0} - C_\pm}{\sqrt{\zeta} - C_\pm}\right)^{\frac{2h}{3-2h}}, \quad C_\pm = \frac{1-h}{3-2h}\left(2\sqrt{\zeta_0} \mp \frac{u_0}{\sqrt{g}}\right). \tag{22}$$

To find the solution inside the expansion wave, we consider the characteristic ray through the origin $(0, 0)$ and a general point (λ, ξ) inside the wave. The slope of the characteristic is

$$\frac{d\xi}{d\lambda} = \frac{\xi}{\lambda} = \mu = \sigma_\pm = \frac{(1-h)u \pm \sqrt{g\zeta}}{A}. \tag{23}$$

This, together with (21) and (22), gives $\zeta(\mu)$, $u(\mu)$, and $A(\mu)$ implicitly; in the special case of $h = 0$ or $h = 1$, these functions can be written explicitly.

3.2.2. Shocks. We start from the following Rankine–Hugoniot jump conditions of system (18):

$$s[\zeta A] = [(1-h)\zeta u], \tag{24}$$

$$s[\zeta u A] = [(1-h)\zeta u^2 + \frac{g}{2}\zeta^2], \tag{25}$$

$$s[A] = -[hu], \tag{26}$$

where [.] denotes the jump across the discontinuity whose speed is denoted by $s = \frac{d\xi}{d\lambda}$.

We denote the preshock flow state by $\mathbf{U_0} = (\zeta_0, u_0, A_0)^T$ and the postshock flow state by $\mathbf{U} = (\zeta, u, A)^T$, respectively. Then the shock jump relations after some algebraic manipulations can be expressed as follows:

$$u = u_0 \pm \sqrt{g(\zeta + \zeta_0)(\zeta - \zeta_0)^2/(2\zeta\zeta_0)}, \tag{27}$$

$$A = A_0 - \frac{h(u - u_0)}{s_{sh\pm}}, \tag{28}$$

where

$$s_{sh\pm} = \frac{(1-h)u_0}{A_0} \pm \frac{\sqrt{g\zeta(\zeta+\zeta_0)/(2\zeta_0)}}{A_0}. \tag{29}$$

Formulas (27)–(28) hold for $h : 0 \leq h \leq 1$.

3.2.3. Contact (or slip) lines. The degenerate wave corresponds to speed $s = \sigma_1 = 0$. From the Rankine–Hugoniot jump conditions (24)–(26) we have

$$\zeta = \zeta_0, \tag{30}$$

$$u = u_0. \tag{31}$$

The only variable which can change its value across this wave is A. Since the flow variables ζ and u are continuous, *there is no flow discontinuity in the form of a contact (slip) line.*

We compare here the computation of the one-dimensional shallow water flow with the one-dimensional flow of gas dynamics. As seen from the analysis, the one-dimensional shallow water equations have just one type of flow discontinuity (shocks), but there is no flow contact (slip) lines which exist in the one-dimensional Euler equations of gas dynamics. In the latter case it was shown [12] that the Lagrangian system of coordinates is the best for resolution of the contact lines. However, with no contact line to resolve, Eulerian and Lagrangian coordinates are on equal footing for accuracy, as is verified in our computation. The adaptive Godunov scheme [13], [14], which resolves shocks crisply, can now be applied to either the Lagrangian coordinates or the Eulerian ones, or indeed for any h.

4. Two-dimensional shallow water flow.

4.1. Hyperbolicity. It is well known that the system of unsteady shallow water equations (1) in Cartesian coordinates is hyperbolic, meaning that all its eigenvalues are real, and there exist a complete set of linearly independent eigenvectors. Because the transformation from (t, x, y) to the unified coordinates (λ, ξ, η) involves the dependent variables (u, v), there is no guarantee that the resulting system (8) will necessarily be hyperbolic. We now study the hyperbolicity of the system (8). To do that we rewrite the system (8) as

$$\mathbf{A}\frac{\partial \mathbf{U}}{\partial \lambda} + \mathbf{B}\frac{\partial \mathbf{U}}{\partial \xi} + \mathbf{C}\frac{\partial \mathbf{U}}{\partial \eta} = \mathbf{S}_1, \tag{32}$$

where

$$\mathbf{U} = \begin{pmatrix} \zeta \\ u \\ v \\ A \\ B \\ L \\ M \end{pmatrix}, \quad \mathbf{S}_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ uh_\xi \\ vh_\xi \\ uh_\eta \\ vh_\eta \end{pmatrix}, \tag{33}$$

and

$$\mathbf{A} = \frac{\partial \mathbf{E}}{\partial \mathbf{U}}, \quad \mathbf{B} = \frac{\partial \mathbf{F}}{\partial \mathbf{U}}, \quad \mathbf{C} = \frac{\partial \mathbf{G}}{\partial \mathbf{U}}. \tag{34}$$

System (32) is said to be hyperbolic (also called strongly hyperbolic, or fully hyperbolic) in λ if [15]

(i) all the eigenvalues σ of

$$\det(\sigma\mathbf{A} - \alpha\mathbf{B} - \beta\mathbf{C}) = 0$$

are real for every pair $(\alpha, \beta) \in R^2 : \alpha^2 + \beta^2 = 1$; and

(ii) associated with the eigenvalues there exist a complete set of seven linearly independent right eigenvectors in the state space.

System (32) is said to be weakly hyperbolic in λ if (i) is satisfied and (ii) is not.

The eigenvalues of (32) can be found by direct computation, and the results are as follows.

Case (a). $h \neq 1$. In this case we get

$$\sigma_1 = 0 \qquad (\textit{multiplicity } 4), \tag{35}$$

$$\sigma_2 = (1-h)(\alpha' u + \beta' v), \tag{36}$$

$$\sigma_\pm = \sigma_2 \pm \sqrt{g\zeta(\alpha'^2 + \beta'^2)}, \tag{37}$$

where

$$\alpha' = (\alpha M - \beta B)/\triangle, \qquad \beta' = -(\alpha L - \beta A)/\triangle.$$

The corresponding right eigenvectors are

$$\mathbf{r}_1 = (0,0,0,1,0,0,0)^T, \tag{38}$$

$$\mathbf{r}_2 = (0,0,0,0,1,0,0)^T, \tag{39}$$

$$\mathbf{r}_3 = (0,0,0,0,0,1,0)^T, \tag{40}$$

$$\mathbf{r}_4 = (0,0,0,0,0,0,1)^T \tag{41}$$

for σ_1,

$$\mathbf{r}_5 = \left(0, \beta', -\alpha', -\frac{h}{\mu}\alpha\beta', \frac{h}{\mu}\alpha\alpha', -\frac{h}{\mu}\beta\beta', \frac{h}{\mu}\beta\alpha'\right)^T \tag{42}$$

for σ_2, and

$$\mathbf{r}_{6,7} = \left(1, \frac{\alpha' g}{m_\pm}, \frac{\beta' g}{m_\pm}, \frac{-\alpha\alpha' gh}{\sigma_\pm m_\pm}, \frac{-\alpha\beta' gh}{\sigma_\pm m_\pm}, \frac{-\beta\alpha' gh}{\sigma_\pm m_\pm}, \frac{-\beta\beta' gh}{\sigma_\pm m_\pm}\right)^T \tag{43}$$

for $\sigma_\pm$, where

$$m_\pm = \pm\sqrt{g\zeta(\alpha'^2 + \beta'^2)}.$$

The eigenvectors $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_7$ are linearly independent, forming a complete basis in the state space; system (32) is therefore hyperbolic for $h \neq 1$. This includes the Eulerian coordinates as a special case when $h = 0$.

Case (b). $h = 1$ (Lagrangian case). In this case the eigenvalues are

$$\sigma_1 = 0 \qquad (\textit{multiplicity } 5), \tag{44}$$

$$\sigma_\pm = \pm\sqrt{g\zeta(\alpha'^2 + \beta'^2)}. \tag{45}$$

The eigenvectors associated with $\sigma_\pm$ are

$$\mathbf{r}_\pm = \left(1, \frac{\alpha' g}{\sigma_\pm}, \frac{\beta' g}{\sigma_\pm}, \frac{-\alpha\alpha' g}{\sigma_\pm^2}, \frac{-\alpha\beta' g}{\sigma_\pm^2}, \frac{-\beta\alpha' g}{\sigma_\pm^2}, \frac{-\beta\beta' g}{\sigma_\pm^2}\right)^T. \tag{46}$$

Associated with $\sigma_1 = 0$ (multiplicity 5),

$$\operatorname{rank}(\sigma\mathbf{A} - \alpha\mathbf{B} - \beta\mathbf{C})\Big|_{\sigma=\sigma_1} = 3;$$

hence there exist the four, and only four, following linearly independent right eigenvectors:

$$\mathbf{r}_1 = (0,0,0,1,0,0,0)^T, \tag{47}$$

$$\mathbf{r}_2 = (0,0,0,0,1,0,0)^T, \tag{48}$$

$$\mathbf{r}_3 = (0,0,0,0,0,1,0)^T, \tag{49}$$

$$\mathbf{r}_4 = (0,0,0,0,0,0,1)^T. \tag{50}$$

We therefore arrive at the conclusion that *the system of unsteady two-dimensional shallow water equations in Lagrangian coordinates is weakly hyperbolic*, lacking one eigenvector, although all eigenvalues are real. It is interesting to note that similar results were obtained in the gas dynamics case; namely, the two-dimensional and three-dimensional Euler equations written in the unified coordinates are hyperbolic for any h, $0 \leq h < 1$, but they are only weakly hyperbolic for $h = 1$ (Lagrangian case). Moreover, it is shown in [1] that for the smooth solutions the system of two-dimensional Euler equations of gas dynamics written in the classical Lagrangian coordinates is equivalent to the same system written in the unified coordinates with $h = 1$. The steps of the proof can easily be repeated for the system of two-dimensional shallow water equations to show that the system of two-dimensional shallow water equations written in the classical Lagrangian coordinates is also weakly hyperbolic.

To avoid possible difficulties, such as the existence and uniqueness of weak solution, arising from the lack of a complete set of eigenvectors, we shall use $h = 0.999$ (or any h very close to 1.0) for which the two-dimensional shallow water equations are hyperbolic.

In summary, use of Lagrangian coordinates in CFD for two-dimensional shallow water equations can not only cause severe grid deformation, but it also renders the two-dimensional shallow water equations weakly hyperbolic with all its possible consequences on numerical computation. More will be said about the Lagrangian case at the end of section 4.4.

4.2. Determination of *h*. As mentioned earlier, the chief advantage of the unified coordinates is the new degree of freedom in choosing h. Many choices are possible, and the simplest one would be to choose a constant value for it, as was done in section 3. Numerical experiments for constant h will be presented in section 6 to show its effects on grid deformation and on resolution of flow discontinuities. In general, it is necessary to restrict h to within the range $0 \leq h \leq 1$. For $h > 1$, the eigenvalue σ_2 in (36) has an opposite sign to that for $h < 1$, indicating that signals propagate in the wrong direction. Our computations for $h > 1$ break down immediately. On the other hand, for $h < 0$, which means the pseudoparticles are moving in the opposite direction to the fluid particles, computation can be carried out initially, but after some finite time it also breaks down. No difficulty has been encountered in all our computations if h is restricted to $0 \leq h < 1$.

As shown in [1], a good choice for h is to preserve the grid angles in the solution process which marches in λ, i.e.,

$$\frac{\partial}{\partial\lambda}\left[\frac{\nabla\xi}{|\nabla\xi|}\cdot\frac{\nabla\eta}{|\nabla\eta|}\right] = 0. \tag{51}$$

Since

$$\begin{aligned}\nabla\xi &= (M,-L)/\triangle,\\ \nabla\eta &= (-B,A)/\triangle,\end{aligned} \tag{52}$$

condition (51) becomes

$$\frac{\partial}{\partial\lambda}\left[\frac{AL+BM}{\sqrt{A^2+B^2}\sqrt{L^2+M^2}}\right]=0. \tag{53}$$

By making use of the last four equations of (8), it is easy to show that (53) is equivalent to

$$S^2J\frac{\partial h}{\partial\xi}+T^2I\frac{\partial h}{\partial\eta}=\left[S^2\left(B\frac{\partial u}{\partial\xi}-A\frac{\partial v}{\partial\xi}\right)-T^2\left(M\frac{\partial u}{\partial\eta}-L\frac{\partial v}{\partial\eta}\right)\right]h, \tag{54}$$

where

$$S^2=L^2+M^2,\quad T^2=A^2+B^2. \tag{55}$$

A consequence of determining h from (54) is that if the grid is orthogonal at $\lambda=0$ it will remain so for subsequent λ. Orthogonal grids are known to possess many desirable properties over nonorthogonal grids, e.g., attaining higher accuracy than nonorthogonal grids. Computationally, (54) is to be solved at every time step after the flow variables $\mathbf{Q}=(\zeta,u,v)^T$, and the geometric variables $\mathbf{K}=(A,B,L,M)^T$ are found. It is thus a first order linear partial differential equation for $h(\xi,\eta;\lambda)$ with λ appearing as a parameter. To find solution h in the range

$$0\le h\le 1 \tag{56}$$

we note that (54) is linear and homogeneous, and therefore it possesses two properties: (a) positive solution $h>0$ always exists, and (b) if h is a solution to (54) so is h/C, C being any constant. Making use of property (a), we let $g=\ln(hq)$ to get

$$\begin{aligned}S^2(A\sin\theta-B\cos\theta)\frac{\partial g}{\partial\xi}+T^2(M\cos\theta-L\sin\theta)\frac{\partial g}{\partial\eta}\\ =S^2\left(B\frac{\partial\cos\theta}{\partial\xi}-A\frac{\partial\sin\theta}{\partial\xi}\right)-T^2\left(M\frac{\partial\cos\theta}{\partial\eta}-L\frac{\partial\sin\theta}{\partial\eta}\right),\end{aligned} \tag{57}$$

where $q=\sqrt{u^2+v^2}$ and θ is the flow angle: $u=q\cos\theta,\ v=q\sin\theta$. Now, if g_1 is any solution to (57), then $h=e^{g_1}/qC$ is a solution to (54) satisfying condition (56), provided we choose C equal to the maximum of e^{g_1}/q over the whole flow field being computed. The reason to work with $\ln(hq)$ instead of $\ln h$ is that from our experience with steady flow [16], hq is continuous across slip lines, hence working with hq can minimize the numerical errors.

Numerically, (57) is solved easily by the method of characteristics if their slopes do not change sign; otherwise, it is solved by iteration.

4.3. Solution strategies. As the system of shallow water equations (8) written in unified coordinates is in conservation form, any well-established shock-capturing method can be used to solve it. We shall use the Godunov method with the MUSCL update to higher resolution to solve system (8). The computation will be done entirely in the $\lambda-\xi-\eta$ space. A physical cell in the $x-y$ plane marching along the pseudoparticle's pathline corresponds to a rectangular cell in the $\xi - \eta$ plane marching in the λ direction in the computational space $\lambda - \xi - \eta$. The superscript k refers to the marching time step number, and the subscripts i and j refer to the cell index number on a time plane $\lambda =$ const. The time step $\Delta\lambda^k = \lambda^{k+1} - \lambda^k$ is uniform for all i and j, but it is always chosen to satisfy the CFL stability condition. The grid divides the computational domain into cubic control volumes, or cells, which in ξ and η direction are centered at $(\lambda^k, \xi_i, \eta_j)$ and have widths $\Delta\xi_i = \xi_{i+1/2} - \xi_{i-1/2}$ and $\Delta\eta_j = \eta_{j+1/2} - \eta_{j-1/2}$ (for all k). Unless otherwise stated we shall use uniform cell width $\Delta\xi_i$ for all i and $\Delta\eta_j$ for all j.

In the physical space (t, x, y) a cuboid cell marching in (λ, ξ, η) space corresponds to a pseudoparticle marching along its path tube with step Δt $(\Delta t = \Delta\lambda)$. The pseudoparticle is bounded by four path surfaces $\xi = \xi_{i\pm1/2}$ and $\eta = \eta_{j\pm1/2}$ around it. Initially, any curvilinear coordinate grid on the $x - y$ plane may be used as the $\xi - \eta$ coordinate grid, and the initial geometric variables $\mathbf{K} = (A, B, L, M)^T$ can be determined from (2) as part of the initial conditions. A stationary solid wall is always a path surface of the fluids and hence also of the pseudofluids; it is therefore a coordinate surface of the unified coordinates.

Applying the divergence theorem to (8) over the cuboid cell (i, j, k) results in

$$\mathbf{E}_{i,j}^{k+1} = \mathbf{E}_{i,j}^{k} - \frac{\Delta\lambda^k}{\Delta\xi_i}\left(\mathbf{F}_{i+1/2,j}^{k+1/2} - \mathbf{F}_{i-1/2,j}^{k+1/2}\right) - \frac{\Delta\lambda^k}{\Delta\eta_j}\left(\mathbf{G}_{i,j+1/2}^{k+1/2} - \mathbf{G}_{i,j-1/2}^{k+1/2}\right), \tag{58}$$

$$i = 1, 2, \ldots, m, \quad j = 1, 2, \ldots, n, \tag{59}$$

where the notation for the cell average of any quantity f is

$$f_{i,j}^k = \frac{1}{\Delta\xi_i\Delta\eta_j}\int_{\xi_{i-1/2}}^{\xi_{i+1/2}}\int_{\eta_{j-1/2}}^{\eta_{j+1/2}} f(\lambda^k, \xi, \eta)d\xi\ d\eta, \tag{60}$$

and the notation for the λ average of f is

$$f_{i+1/2,j}^{k+1/2} = \frac{1}{\Delta\lambda^k}\int_{\lambda^k}^{\lambda^{k+1}} f(\lambda, \xi_{i+1/2}, \eta_j)d\lambda, \tag{61}$$

$$f_{i,j+1/2}^{k+1/2} = \frac{1}{\Delta\lambda^k}\int_{\lambda^k}^{\lambda^{k+1}} f(\lambda, \xi_i, \eta_{j+1/2})d\lambda. \tag{62}$$

According to Godunov's idea, the cell-interface fluxes $\mathbf{F}_{i+1/2,j}^{k+1/2}$ and $\mathbf{G}_{i,j+1/2}^{k+1/2}$ for the cell (i, j) are to be obtained from the self-similar solution of a local two-dimensional Riemann problem formed by the averaged constant state $\mathbf{E}_{i,j}$ of the cell (i, j) and those of its adjacent cells. Unfortunately, such a solution to (8) is unavailable at the present time. Indeed, even a two-dimensional Riemann solution to the simpler system (1), which is a special case of (8) when $h = 0$, is not yet available. On the other hand, it is known that a monotone difference scheme to a general conservation form converges to the physically relevant entropy-satisfying solution. In particular, Crandall and Majda [17] proved convergence for dimensional splitting algorithms when each step is approximated by a monotone difference scheme (such as the Godunov scheme) for a scalar conservation law in multidimension.

In view of the above, we shall numerically solve (8) using a Godunov-type scheme based on the dimensional splitting approximation to reduce the two-dimensional flow problem to two one-dimensional flow problems.

The dimensional splitting technique for finding an approximate solution to the Riemann problem in multidimensional flow is now well established and widely used. This technique renders the solution of a multiple spatial-dimensional problem to a sequential solution of several one-dimensional problems. The Godunov splitting and the Strang splitting [18] are frequently used in practical applications. We shall use the Strang splitting in this paper. Let $\mathcal{L}^{\xi}_{\Delta\lambda}$ represent the exact solution operator for the one-dimensional equation in the $\lambda-\xi$ plane and $\mathcal{L}^{\eta}_{\Delta\lambda}$ similarly defined, then according to Strang splitting

$$\mathbf{E}^{k+1} = \mathcal{L}^{\xi}_{\frac{\Delta\lambda}{2}} \mathcal{L}^{\eta}_{\Delta\lambda} \mathcal{L}^{\xi}_{\frac{\Delta\lambda}{2}} \mathbf{E}^{k}, \tag{63}$$

where $\Delta\lambda = \lambda^{k+1} - \lambda^{k}$.

The solution operator $\mathcal{L}^{\xi}_{\Delta\lambda}$ for the Riemann problem with variable coefficients in the governing equations in $\lambda - \xi$ plane will now be given in details.

4.4. The $\boldsymbol{\xi}$-Split Riemann problem. With the use of dimensional splitting, the solution of the original two-dimensional equation system is replaced by the sequential computation involving two one-dimensional equation systems. We consider in detail the resulting Riemann problem in the $\lambda - \xi$ plane; the resulting Riemann problem in the $\lambda - \eta$ plane can be discussed similarly.

In the $\mathcal{L}^{\xi}$ operator, it is assumed that $\frac{\partial}{\partial \eta} = 0$. Hence (8) becomes

$$\frac{\partial \mathbf{E}}{\partial \lambda} + \frac{\partial \mathbf{F}}{\partial \xi} = 0, \tag{64}$$

where

$$\mathbf{E} = \begin{bmatrix} \zeta\triangle \\ \zeta\triangle u \\ \zeta\triangle v \\ A \\ B \\ L \\ M \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} \zeta(1-h)I \\ \zeta(1-h)Iu + \frac{1}{2}g\zeta^2 M \\ \zeta(1-h)Iv - \frac{1}{2}g\zeta^2 L \\ -hu \\ -hv \\ 0 \\ 0 \end{bmatrix} \tag{65}$$

with

$$\triangle = AM - BL, \qquad I = uM - vL. \tag{66}$$

We rewrite (64) by using the component of velocity $\mathbf{q}$ in the direction $\mathbf{n}$ normal to, and $\mathbf{t}$ tangential to, the plane $\xi =$ const., i.e.,

$$\mathbf{n} = \frac{\nabla\xi}{|\nabla\xi|} = (M, -L)/S, \qquad \mathbf{t} = (L, M)/S, \tag{67}$$

$$\omega = \mathbf{q}\cdot\mathbf{n} = (uM - vL)/S, \tag{68}$$

$$\tau = \mathbf{q}\cdot\mathbf{t} = (uL + vM)/S, \tag{69}$$

where

$$S = (L^2 + M^2)^{1/2}. \tag{70}$$

Hereafter we shall abandon the last two equations of system (64), keeping in mind that $L = L(\xi)$ and $M = M(\xi)$ are given.

Equations (64) now become

$$\frac{\partial \mathbf{E}}{\partial \lambda} + \frac{\partial \mathbf{F}}{\partial \xi} = \mathbf{S}_2, \tag{71}$$

where

$$\mathbf{E} = \begin{bmatrix} \zeta\triangle \\ \zeta\triangle\omega \\ \zeta\triangle\tau \\ A \\ B \end{bmatrix}, \quad \mathbf{F} = S \begin{bmatrix} \zeta(1-h)\omega \\ \zeta(1-h)\omega^2 + \frac{1}{2}g\zeta^2 \\ \zeta(1-h)\omega\tau \\ -h(M\omega + L\tau)/S^2 \\ h(L\omega - M\tau)/S^2 \end{bmatrix}, \quad \mathbf{S}_2 = \frac{\partial \psi}{\partial \xi} S \begin{bmatrix} 0 \\ \zeta(1-h)\omega\tau \\ \zeta(1-h)\omega^2 + \frac{1}{2}g\zeta^2 \\ 0 \\ 0 \end{bmatrix}, \tag{72}$$

where

$$\tan \psi = M/L. \tag{73}$$

Equations (71) in the $\lambda - \xi$ plane resulting from dimensional splitting of the two-dimensional equations (8) are more complicated than the genuinely one-dimensional equations (12), and the solution to their Riemann problem is explained below.

The ξ-split Riemann problem is thus

$$\begin{cases} \dfrac{\partial \mathbf{E}}{\partial \lambda} + \dfrac{\partial \mathbf{F}}{\partial \xi} = \mathbf{S}_2, & \lambda > 0, \\ \mathbf{E}(0,\xi) = \begin{cases} \mathbf{E}_l, & \xi < 0, \\ \mathbf{E}_r, & \xi > 0, \end{cases} \end{cases} \tag{74}$$

where $\mathbf{E}_l$ and $\mathbf{E}_r$ are constant, and our purpose is to find the flux $\mathbf{F}$ on $\xi = 0$ to be used in the Godunov scheme to update the conserved quantities $\mathbf{E}$. At time level λ^k (taken to be equal to 0) h_r and h_l are the values of h at the cell $(i+1, j)$ and (i, j). They are *assumed* constant for $0 \le \lambda < \triangle\lambda$, i.e.,

$$\frac{\partial h}{\partial \lambda} = 0 \qquad (0 \le \lambda < \triangle\lambda), \tag{75}$$

and this is consistent with the h-equation (54). However, h changes its value at $\lambda = \triangle\lambda$ as given by (54), whose coefficients are evaluated at $\triangle\lambda$.

Now we first find all possible solutions to (71) for $\xi > 0$ and $\xi < 0$ separately and then use them to construct the solution to the Riemann problem (74).

Case (1). $\boldsymbol{\xi > 0}$.

Since $L = L_r =$ const. and $M = M_r =$ const., hence $S = S_r =$ const., $\psi = \psi_r =$ const., therefore

$$\mathbf{S}_2 = 0 \tag{76}$$

and

$$\begin{cases} \dfrac{\partial \mathbf{E}}{\partial \lambda} + \dfrac{\partial \mathbf{F}}{\partial \xi} = 0, & \lambda > 0, \quad \xi > 0, \\ \mathbf{E}(0,\xi) = \mathbf{E}_r, & \xi > 0, \end{cases} \tag{77}$$

where

$$\mathbf{E} = \begin{bmatrix} \zeta\triangle \\ \zeta\triangle\omega \\ \zeta\triangle\tau \\ A \\ B \end{bmatrix}, \quad \mathbf{F} = S \begin{bmatrix} \zeta(1-h)\omega \\ \zeta(1-h)\omega^2 + \frac{1}{2}g\zeta^2 \\ \zeta(1-h)\omega\tau \\ -h(M\omega + L\tau)/S^2 \\ h(L\omega - M\tau)/S^2 \end{bmatrix} \tag{78}$$

with $h = h_r = \text{const.}$ Similarity solution to (77) exists in the form

$$\mathbf{E} = \mathbf{E}(\mu), \qquad \mu = \frac{\xi}{\lambda}. \tag{79}$$

Possible solutions to (77) are

(i) a constant state $\mathbf{E} = \text{const.}$;
(ii) a centered expansion wave;
(iii) a shock;
(iv) a contact (slip) line.

Let us first find the eigenvalues and the corresponding right eigenvectors of the system (77).

For a smooth solution, (77) can be written as

$$\mathbf{A}\frac{\partial \mathbf{U}}{\partial \lambda} + \mathbf{B}\frac{\partial \mathbf{U}}{\partial \xi} = \mathbf{0}, \tag{80}$$

where $\mathbf{U} = (\zeta, \omega, \tau, A, B)^T$, $\mathbf{A} = \frac{\partial \mathbf{E}}{\partial \mathbf{U}}$, and $\mathbf{B} = \frac{\partial \mathbf{F}}{\partial \mathbf{U}}$.

In order to obtain the eigenvalues σ we need to find the determinant of the matrix $(\sigma\mathbf{A} - \mathbf{B})$. Direct computation gives

$$\det(\sigma\mathbf{A} - \mathbf{B}) = \sigma^2 m\zeta^2(m^2 - S^2\zeta g), \quad m = \sigma\triangle - S(1-h)\omega. \tag{81}$$

Case (1a). $h \neq 1$. From the vanishing of this determinant we get our eigenvalues.

$$\sigma_1 = 0 \qquad (\textit{multiplicity } 2), \tag{82}$$

$$\sigma_2 = \frac{S}{\triangle}(1-h)\omega, \tag{83}$$

$$\sigma_\pm = \frac{S}{\triangle}\left\{(1-h)\omega \pm \sqrt{g\zeta}\right\}. \tag{84}$$

We emphasize again that $S = S_r = \text{const.}$ and $h = h_r = \text{const.}$, $h \neq 1$. The corresponding set of right eigenvectors is

$$\mathbf{r}_1 = (0,0,0,0,1)^T, \tag{85}$$

$$\mathbf{r}_2 = (0,0,0,1,0)^T, \tag{86}$$

$$\mathbf{r}_3 = \left(0,0,1,-\frac{hL}{S\sigma_2},-\frac{hM}{S\sigma_2}\right)^T, \tag{87}$$

$$\mathbf{r}_\pm = \left(1, \pm\sqrt{\frac{g}{\zeta}}, 0, \mp\frac{hM}{S\sigma_\pm}\sqrt{\frac{g}{\zeta}}, \pm\frac{hL}{S\sigma_\pm}\sqrt{\frac{g}{\zeta}}\right)^T. \tag{88}$$

Since the eigenvectors (85)–(88) are linearly independent, system (77) is hyperbolic. To classify the characteristic fields, we see that

$$\nabla\sigma_1 \cdot \mathbf{r}_{1,2} = 0 \tag{89}$$

and

$$\nabla\sigma_2 \cdot \mathbf{r}_3 = 0, \tag{90}$$

implying that characteristic fields corresponding to the $\sigma_{1,2}$ and σ_3 are linearly degenerate. On the other hand,

$$\nabla\sigma_\pm = \left(\pm\frac{S}{2\triangle}\sqrt{\frac{g}{\zeta}}, \frac{S(1-h)}{\triangle}, 0, -\frac{\sigma_\pm M}{\triangle}, \frac{\sigma_\pm L}{\triangle}\right) \tag{91}$$

and

$$\nabla\sigma_\pm \cdot \mathbf{r}_\pm = \pm\frac{3S}{2\triangle}\sqrt{\frac{g}{\zeta}} \neq 0. \tag{92}$$

Therefore the $\sigma_\pm$ characteristic families are genuinely nonlinear.

Case (1b). $h = 1$ (Lagrangian case). In this case the eigenvalues are

$$\sigma_1 = 0 \qquad (\textit{multiplicity } 3), \tag{93}$$

$$\sigma_\pm = \pm\frac{S}{\triangle}\sqrt{g\zeta}. \tag{94}$$

The eigenvectors associated with $\sigma_\pm$ are

$$\mathbf{r}_\pm = \left(1, \pm\sqrt{\frac{g}{\zeta}}, 0, -\frac{M\triangle}{S^2\zeta}, \frac{L\triangle}{S^2\zeta}\right)^T. \tag{95}$$

Associated with $\sigma_1 = 0$ (multiplicity 3),

$$\operatorname{rank}(\sigma\mathbf{A} - \mathbf{B})\Big|_{\sigma=\sigma_1} = 3;$$

hence there exist the two, and only two, following linearly independent right eigenvectors:

$$\mathbf{r}_1 = (0,0,0,1,0)^T, \tag{96}$$

$$\mathbf{r}_2 = (0,0,0,0,1)^T. \tag{97}$$

We therefore arrive at the conclusion that *system* (77) *resulting from dimensional splitting of the two-dimensional equations in Lagrangian coordinates is weakly hyperbolic*, lacking one eigenvector, although all eigenvalues are real. This is in direct contrast to the genuinely one-dimensional flow case, (12), which is hyperbolic for all values of h, including the Lagrangian case ($h = 1$).

We shall now give solutions to the elementary waves in details: the expansion wave, the shock wave, and the slip line. These solutions will be used in constructing the Riemann solution to (74).

4.4.1. Expansion wave. The expansion wave is a smooth solution from the $\sigma_\pm$ characteristic fields which can be derived from the following system of ODEs:

$$\frac{d\omega}{d\zeta} = \pm\sqrt{\frac{g}{\zeta}}, \tag{98}$$

$$\frac{d\tau}{d\zeta} = 0, \tag{99}$$

$$\frac{dA}{d\zeta} = \mp\frac{hM}{S\sigma_\pm}\sqrt{\frac{g}{\zeta}}, \tag{100}$$

$$\frac{dB}{d\zeta} = \pm\frac{hL}{S\sigma_\pm}\sqrt{\frac{g}{\zeta}}. \tag{101}$$

The solution for (ω, τ) relates the flow state $\mathbf{Q} = (\zeta, \omega, \tau)^T$ in the expansion fan to the initial state $\mathbf{Q_0} = (\zeta_0, \omega_0, \tau_0)^T$ upstream of the fan through the following expressions:

$$\omega = \omega_0 \mp 2\left(\sqrt{g\zeta_0} - \sqrt{g\zeta}\right), \tag{102}$$

$$\tau = \tau_0. \tag{103}$$

Note that on crossing the expansion fan, these relations are independent of $\mathbf{K}_r$ and h_r.

To find the solution inside the expansion fan, we consider the characteristic ray through the origin $(0,0)$ and a general point (λ, ξ) inside the fan. The slope of the characteristic is

$$\frac{d\xi}{d\lambda} = \frac{\xi}{\lambda} = \sigma_\pm = \frac{S}{\triangle}\left\{(1-h)\omega \pm \sqrt{g\zeta}\right\}. \tag{104}$$

Using the above expression and the equation for ω in (102), we get

$$\zeta = \frac{1}{g(3-2h)^2}\left[(1-h)\left(\omega_0 \mp 2\sqrt{g\zeta_0}\right) - \frac{\triangle}{S}\frac{\xi}{\lambda}\right]^2, \tag{105}$$

where $\triangle = AM - BL$ is found from (100) and (101) to be a function of ζ

$$\triangle = \triangle_0\left(\frac{\sqrt{\zeta_0} - C_\pm}{\sqrt{\zeta} - C_\pm}\right)^{\frac{2h}{3-2h}}, \qquad C_\pm = \frac{1-h}{3-2h}\left(2\sqrt{\zeta_0} \mp \frac{\omega_0}{\sqrt{g}}\right). \tag{106}$$

Equation (105) together with (106) thus defines an implicit function $\zeta(\mu)$, $\mu = \frac{\xi}{\lambda}$. Equation (106) reduces to the one-dimensional case, (22), when $L = 0$ and $M = 1$. The expressions for ω and τ in terms of μ can be obtained simply via (102)–(103). Like ζ, they depend on $(\mathbf{K}_r, h_r)$, but at $\mu = \frac{\xi}{\lambda} = 0$ they depend only on h_r. The variations of A and B across an expansion fan can also be obtained from (100) and (101), but they are not needed in calculating the flux. (The flux function $\mathbf{F}$ does not involve A or B; it involves only L and M.)

4.4.2. Shock waves. From the Rankine–Hugoniot jump conditions of the system (77), we get

$$s[\zeta\triangle] = [\zeta(1-h)\omega S], \tag{107}$$

$$s[\zeta\triangle\omega] = \left[\left(\zeta(1-h)\omega^2 + \frac{g\zeta^2}{2}\right)S\right], \tag{108}$$

$$s[\zeta\triangle\tau] = [\zeta(1-h)\omega\tau S], \tag{109}$$

$$s[A] = -\left[\frac{h}{S}(M\omega + L\tau)\right], \tag{110}$$

$$s[B] = \left[\frac{h}{S}(L\omega - M\tau)\right], \tag{111}$$

where [.] denotes the jump across the discontinuity whose speed is denoted by $s = \frac{d\xi}{d\lambda}$. The shock jump relations after some algebraic manipulations can be expressed as follows:

$$s = \frac{S}{\triangle_0}\left[(1-h)\omega_0 \pm \sqrt{\frac{g}{2}\frac{\zeta}{\zeta_0}(\zeta+\zeta_0)}\right], \tag{112}$$

$$\omega = \omega_0 \pm \sqrt{\frac{g(\zeta+\zeta_0)(\zeta-\zeta_0)^2}{2\zeta\zeta_0}}, \tag{113}$$

$$\tau = \tau_0. \tag{114}$$

We note that the relations of the flow variables (ζ, ω, τ) across the shock are independent of $\mathbf{K}_r$ and h_r, but the slope of the shock wave s is dependent on $\mathbf{K}_r$ and h_r. However, this dependence is not needed in finding the water height ζ, and hence (ω, τ) also, at $\mu = 0$ provided only that $s > 0$. (If $s < 0$, the shock wave will appear in the quadrant $(\xi < 0,\, \lambda > 0)$).

Note that the jumps of A and B across a shock may also be obtained from (110) and (111), but they are not needed in calculating the flux $\mathbf{F}$ at $\mu = 0$.

4.4.3. Slip lines. For the slip line corresponding to the speed $s = \sigma_3 = \frac{S}{\triangle_0}(1-h)\omega_0 > 0$ we find, from Rankine–Hugoniot jump conditions (107)–(111),

$$\zeta = \zeta_0, \tag{115}$$

$$\omega = \omega_0, \tag{116}$$

but τ and A, B may jump arbitrarily.

We note again that, except the speed s, the relations (115)–(116) across a slip line are independent of $(\mathbf{K}_r, h_r)$. Although s depends on $(\mathbf{K}_r, h_r)$, the dependence is not needed in calculating (ζ, ω, τ) and the flux $\mathbf{F}$ at $\mu = 0$, provided $s > 0$. (If $s < 0$, the slip line appears in the quadrant $(\xi < 0,\, \lambda > 0)$.)

The flow across the slip line corresponding to $s = \sigma_{1,2} = 0$ cannot be discussed within the quadrant $(\xi > 0,\, \lambda > 0)$ alone.

Case (2). $\boldsymbol{\xi < 0}$.

The solution in the quadrant $(\xi < 0,\, \lambda > 0)$ can be obtained similarly.

Case (3). Riemann solution in $\boldsymbol{-\infty < \xi < +\infty}$.

Now, after obtaining all possible solutions for $\xi > 0$ and $\xi < 0$ separately the question is how to construct the solution to the Riemann problem for $\lambda > 0$, $-\infty < \xi < +\infty$. We note that at $\xi = 0$, (a) $\mathbf{S}_2$ in (71) is a delta function; (b) the coefficients in $\mathbf{E}$ and $\mathbf{F}$ jump discontinuously. These are the difficulties one would face with within the Eulerian system using curvilinear coordinates rather than Cartesian coordinates.

The σ_1-field is linearly degenerate and it corresponds to $s = 0$ in (107)–(111). These are five equations relating three jumps of ζ, ω, and τ and, therefore, in general have no solution, except when $h_r = h_l$, $L_r = L_l$, and $M_r = M_l$. In the latter case, there is a unique solution: the trivial solution $[\zeta] = [\omega] = [\tau] = 0$, i.e., a continuous solution.

To avoid the difficulty of nonexistence of the solution to the Rankine–Hugoniot relations (107)–(111), we replace both h_l and h_r by their average, i.e., $h_l = h_r = 0.5(h_l + h_r) = \overline{h}$, and similarly replace L_l and L_r, M_l, and M_r by their averages $\overline{M}$ and $\overline{L}$, respectively. Consequently, the Rankine–Hugoniot relations are satisfied, and the flow is continuous across $\mu = \frac{\xi}{\lambda} = 0$. We note from previous discussions that these replacements do not alter the relations of the flow variables (ζ, ω, τ) across the elementary waves (the expansion fan, the shock, and the slip line), as they do not depend on $(\mathbf{K}, h)$. It should be pointed out that the replacements of L_l and L_r by $\overline{L}$ and M_l and M_r by $\overline{M}$ are a fictitious one—they are used only to ensure the existence of the solution to (107)–(111)—but these average values are never used in the computation. On the other hand, the replacement of h_l and h_r by $\overline{h}$ is a real one: it is used in (105) when the line $\mu = \frac{\xi}{\lambda} = 0$ is inside the expansion fan which is, however, a rare case.

The Riemann solution for $-\infty < \xi < +\infty$ can now be constructed in the usual way as if the slip line corresponding to $s = \sigma_{1,2} = 0$ did not exist: shock (or expansion fan), slip line, and expansion fan (or shock), separated by uniform flow regions.

We note that in solving the split Riemann problems we do not need to use explicitly the complete set of right eigenvectors. The eigenvectors that are used are those corresponding to the genuinely nonlinear characteristic families; these eigenvectors exist even in the case $h = 1$ (Lagrangian coordinates). The missing eigenvectors in Lagrangian coordinates correspond to the linearly degenerated characteristic families and are not employed in constructing solutions to the Riemann problems. This also explains why our computations with $h = 1$ encounter no difficulty and produce results identical to that for $h = 0.999$. We present results for $h = 0.999$, rather than for $h = 1.0$, because the theoretical base for the existence and uniqueness of the solution to the Riemann problem for $h = 1.0$ is not certain.

5. Test examples.

In this section the unified coordinates approach is tested numerically on several examples. The flat bottom case is considered with zero roughness coefficients m and n. In all cases the effects of h on the computational robustness and accuracy are discussed.

Example 1. The first problem is purely a one-dimensional two-dam break problem. In a long channel three different heights of still water are separated by two dams, of which one is located at $x = 0.8$ and the other at $x = 1.2$ (Figure 1(a)). At $t = 0$ the dam located at $x = 0.8$ is broken instantly and completely, resulting in an expansion wave moving upstream and a bore (shock) rushing downstream (Figure 1(b), (c), (d)). The bore (shock) then reaches the second dam ($x = 1.2$) at some time later, triggering it to break completely and resulting in a stronger bore (shock) moving downstream and another expansion wave moving upstream (Figure 1(e), (f), (g)). It is difficult for conventional shock-capturing schemes to accurately compute this problem due to the interaction of the first bore (resulting from the rapture of the first dam) with the second dam. Most conventional shock-capturing schemes [19], [20], [21] smear the first bore, thus giving only its approximate location (Figures 4 and 5)). Consequently, it is impossible computationally to determine the time of breaking of the second dam. In our computation the shock-adaptive Godunov method [13], [14], which gives infinite bore resolution, is applied and this difficulty is avoided. The exact time when the second dam breaks can be found by using formula (29) for the speed of the shock (bore), and it is found to be (with eight significant figures of accuracy) 0.24292472. In our computation the second dam breaks at $t = 0.2375$.

Figures 1 and 2 show the evolution of water height and velocity with time. We note that after the breaking of the second dam, the water velocity increases significantly (Figure 2(d), (e), (f)). This combination of the strong bore moving with high velocity can potentially be destructive. Comparisons between exact solutions and computed ones at the time $t = 0.1$ (after the first dam broke) and the time $t = 0.3375$ (0.1 time units after the second dam broke) are presented in Figure 3. Figure 4 presents computed results for water height and velocity at $t = 0.22$ using ordinary Godunov scheme with the MUSCL update. We note that smeared shock reaches the second dam prematurely. It is seen in Figure 5 that with the ordinary shock-capturing method the shock is smeared, though its average location is correct, as expected. It is also seen that the shock-adaptive Godunov scheme perfectly resolves the bore, giving its exact location. In all computations we use $h = 0$ (Eulerian).

Example 2. This example, known as the Salzman problem in gas dynamics, is presented to show the role an irregular grid plays in producing spurious flow. The problem consists of a rectangular channel (Figure 6) whose walls form reflective boundaries. The initial data are

$$\begin{cases} \zeta = 4.0, & u = 7.4246, \qquad v = 0.0, x < 0, \\ \zeta = 1.0, & u = 0.0, \qquad\quad\; v = 0.0, 0 \le x \le 1. \end{cases}$$

They are chosen in such a way that the discontinuity, initially located at $x = 0.0$, will result in a plane shock propagating at the speed equal to 9.899 in the x direction. The initial grid has 10 uniform cells in the y-direction ($0.0 \le y \le 0.1$) and 100 nonuniform cells in the x-direction ($0 \le x \le 1$). The coordinates of the cell centers in Figure 6 are

$$\begin{aligned} x_{i,j} &= (i-1) * \triangle + (11-j) * \triangle * \sin\frac{\pi(i-1)}{100}, \\ y_{i,j} &= (j-1) * \triangle + 0.5 * \triangle, \\ &\quad i = 1, \dots, 101 \qquad j = 1, \dots, 10, \end{aligned}$$

where $\triangle = 1/100$. We shall perform computation for $h = 0.999$ and grid-angle preserving h. Figure 7 presents resulting grid (7(a)) and water height (7(b)) for the case $h = 0.999$ at the time $t = 0.06$, which is several time steps before the computation breaks down. As we can see, this grid is highly deformed near the position $x = 0.5$ behind the shock. Looking at the velocity vectors (Figure 7(c)) we see that there is a presence of spurious flow: the y-component of velocity (especially near $x = 0.5$) and hence vorticity which, in turn, affects the grid and the water height (Figure 7(a), (b)).

Similar results of spurious flow production were reported, for example, by Dukowicz and Meltz in [22], where it was found that Lagrangian coordinates do not preserve the one-dimensionality of a plane shock but produce spurious vorticity. They successfully introduced a technique to filter out the spurious vorticity, while retaining the real one, if present. In another approach, described by Caramana and Shashkov in [23], the spurious vorticity is eliminated by proper use of subzonal Lagrangian masses and associated densities and pressures; these subzonal pressures give rise to forces that resist these spurious motions. We have found that the spurious flow can be avoided automatically by using unified coordinates with grid-angle preserving h. The results are presented in Figure 8. Here we note that although the vertical component of velocity is still present (Figure 8(c), especially near the shock, it is very small and is not magnified during the computation. These results (Figure 8) are almost identical to those of gas dynamics case in [22], which requires a special technique to filter out the spurious vorticity.

Example 3. The next example is a two-dimensional steady Riemann problem generated by two uniform parallel flows as

$$(\zeta, F, \theta) = \begin{cases} (1, 4, 0), & y > 0.5, \\ (2, 2.4, 0), & y < 0.5, \end{cases}$$

where F is the Froude number and θ the flow angle, $\theta = \tan^{-1}(v/u)$. The flow contains a shock wave, a slip line, and an expansion wave (Figure 9). The slip line is sensitive to the dissipative property of the numerical methods. In [24], [25] the problem was solved numerically using a generalized Lagrangian method which perfectly resolves slip lines. However, that method is valid only for steady flow, whereas the method in this paper is valid for unsteady flow as well. Since the analytical solution for the problem is available, it is an excellent benchmark problem for the verification of numerical methods. In the computation, the steady flow is achieved by time marching until the flow structure and the variables do not change with time. A grid of 60×100 with $\Delta\xi = \Delta\eta = 0.01$ is employed in the computation. Initially, a grid with $\Delta x = \Delta y = 0.01$ in the physical plane is laid over a domain of $\{0 \leq x \leq 0.6,\ 0 \leq y \leq 1\}$. The initial data are given at each cell according to its position in $y > 0.5$ or $y < 0.5$, representing cell-averaged values. The physical domain will change with time according to the pseudoparticle's velocity $h\mathbf{q}$ if h is not zero. If we follow the computational cells (pseudoparticles), they will move out the initial physical domain, and it would be difficult to have a steady state of flow in the original physical domain. To avoid this, a special technique called "motionless viewing window" is applied as in the classical Lagrangian method. Accordingly, the column of cells which have moved out of the original physical domain to the right are deleted, while a new column of cells are added at the input flow boundary on the left.

In Figures 10–12 we show computed Froude number distributions using our unified code for $h = 0$, $h = 0.5$, and $h = 0.999$, compared with the exact solutions. The poor resolution of the slip line seen in Figure 10 is a common feature of any method based on Eulerian coordinates, as a result of Godunov averaging across slip lines which, in general, do not coincide with the (Eulerian) coordinate lines. A comparison of Figures 10–12 also shows that the slip line resolution improves with increasing h from $h = 0$ to $h = 0.999$, as expected.

Figure 13 shows the computed Froude number using the grid-angle preserving h as determined by (57), which is solved at each time step using the method of characteristics. We see that its slip line resolution is almost as sharp as that for $h = 0.999$, and it is much better resolved than those for $h = 0$ and $h = 0.5$.

All the computations started with the Eulerian grid (Figure 14). The flow-generated grids, i.e., the lines joining the cell centers, at steady state are shown in Figures 15–17. We note that (a) the grid using grid-angle preserving h is everywhere orthogonal, (b) the grids for $h = 0.5$ and $h = 0.999$ are severely deformed near the slip line, and such grid deformation, although it doesn't bring any troubles in the present example, can cause inaccuracy in other steady flows [1] as well as for the unsteady flows; see Example 2 above and the following examples.

Example 4. The next example is the "implosion/explosion" problem, so called by analogy with gas dynamics. It is an unsteady problem in a two-dimensional container. Initially, two regions of still water are separated by a cylindrical wall (radius 0.2) centered in the 1×1 square domain shown in Figure 18. The depth of the water is 0.1 within the cylinder and 1 outside. At $t = 0$ the wall is removed and the resulting flow is investigated. Initially, a uniform rectangular grid 100×100 with $\Delta\xi = \Delta\eta = \Delta x = \Delta y = 0.01$ is given (Figure 19). We test this example with $h = 0$, $h = 0.5$, $h = 0.999$, and the grid-angle preserving h. When $h = 0.999$, the code can run only until $t = 0.04$; soon afterwards it breaks down. This is because the computational cells literally move with the fluid particles and for large h become severely deformed. If we reduce h, say, $h = 0.5$, the code can run longer until $t = 0.08$ when it, too, breaks down. This shows that smaller h can delay the severe cell deformation but cannot remove it. With the grid-angle preserving method, which keeps the grid regular, the code can run for a very long time without any indication of severe grid deformation. Figures 20–22 give the grids at $t = 0.04$ for different cases. We see that irregular grids prevail when h is constant and a regular grid prevails when the grid angles are preserved (Figure 22).

The surfaces of water height for different times are presented in Figure 23. The physical behavior of the flow is clearly captured in these pictures. After the wall is removed, the cylindrical shock moves inside the lower level region, and the expansion wave propagates towards the walls (Figure 23(a), (b), (c)). At around time $t = 0.0565$ the shock collapses, resulting in a spectacular rise of water level near the center (Figure 23(d)). In subsequent times the water height decreases in the central region now manifesting in cylindrical shock wave propagating towards the walls (Figure 23(e), (f), (g), (h)). This process of the water in the central region going up and down repeats itself but with more moderate height and depth. We see in Figure 23(b), (c), (e) that the column of water does not have perfect circular shape, as one would expect to have, but has wavy shape. It was shown in [26] that this behavior is a consequence of using a rectangular grid.

Example 5. Finally, we test our code on the two-dimensional dam breaking problem. This test case was computed in, for example, [27], [28].

Two levels of still water in the 1.4 × 1 basin are separated by a dam at the position $x = 0.7$ (Figure 24). The initial height ratio is 10 with values of $\zeta_l = 1$ and $\zeta_r = 0.1$ on two sides of an idealized dam that has been represented as a mathematical discontinuity in water. At time $t = 0$ water is released into the lower level side through a middle third of the dam, forming a wave that propagates while spreading laterally. At the same time, an expansion wave spreads into the reservoir.

A 140 × 100 rectangular grid was chosen in this case. We have performed the computation for $h = 0.999$ and for grid-angle preserving h. Figure 25 represents the flow-generated grid in the case $h = 0.999$ at time $t = 0.04$. The grid is severely distorted by the same reason as in the previous example, and soon after that time the computation breaks down. The computation using grid-angle preserving h is stable, producing results for much larger time. In Figure 26 we present a computed grid which is fairly uniform everywhere even at time $t = 0.15$. The surface elevation of the water height is given in Figure 27, describing the flow in details which are similar to those in [23], [24].

6. Conclusions. In this paper we have successfully adopted the uniform coordinates of Hui, Li, and Li [1] for the shallow water equations. It has been tested on a large number of problems and found that with the free function h in the unified coordinates chosen to preserve grid angles, the unified coordinate system is superior to both Eulerian and Lagrangian systems in that (a) it resolves slip lines as sharply as the Lagrangian system, especially for steady flows; (b) it avoids the severe grid deformation of the Lagrangian system which causes inaccuracy and breakdown of computation; (c) it also automatically avoids spurious flow produced by the Lagrangian system. Additionally, it was found that the two-dimensional shallow water equations in Lagrangian coordinates are only weakly hyperbolic with possible defects in numerical computation.

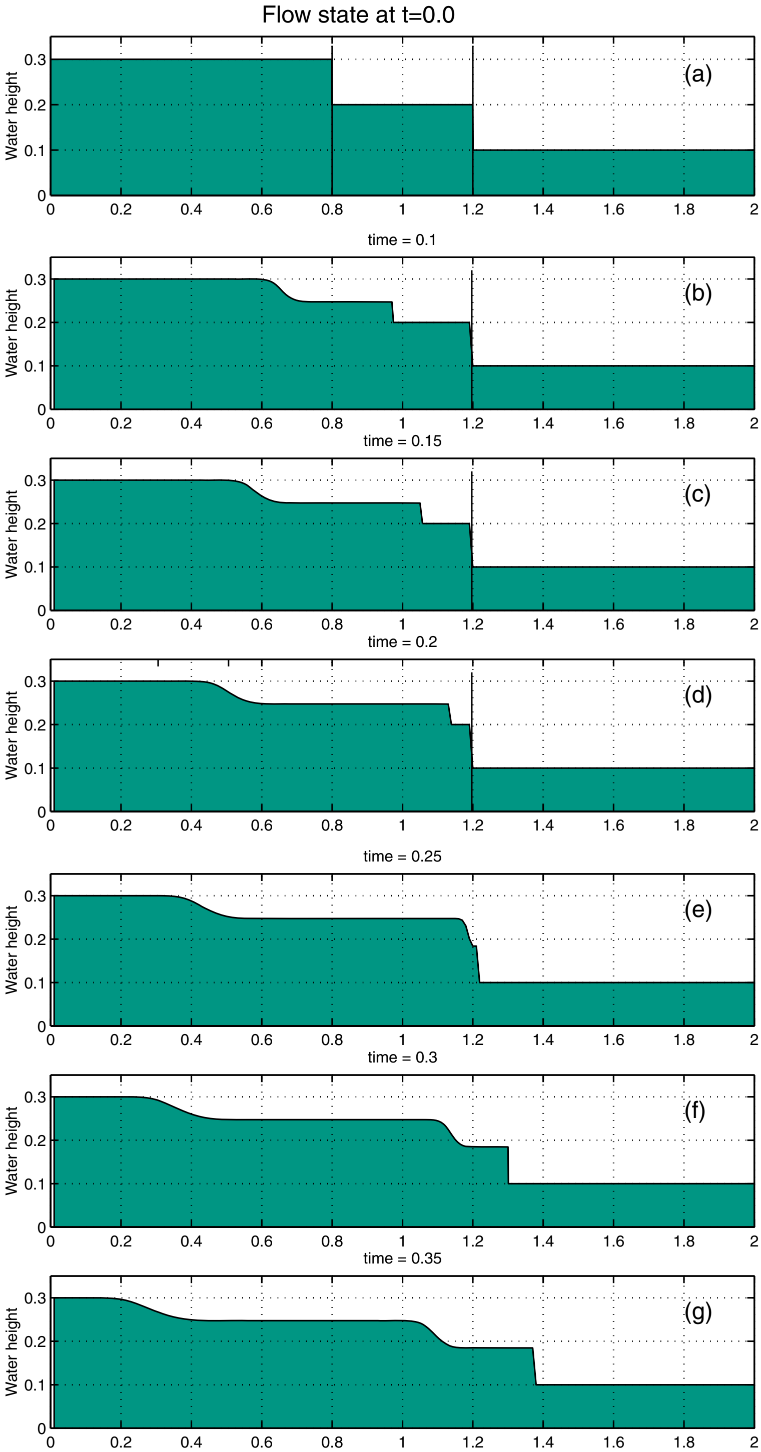


FIG. 1. *Evolution of water height in the two-dam problem. Shock-adaptive Godunov scheme.* $h = 0$ *(Eulerian).* (a) $t = 0.0$; (b) $t = 0.1$; (c) $t = 0.15$; (d) $t = 0.2$; (e) $t = 0.25$; (f) $t = 0.3$; (g) $t = 0.35$.

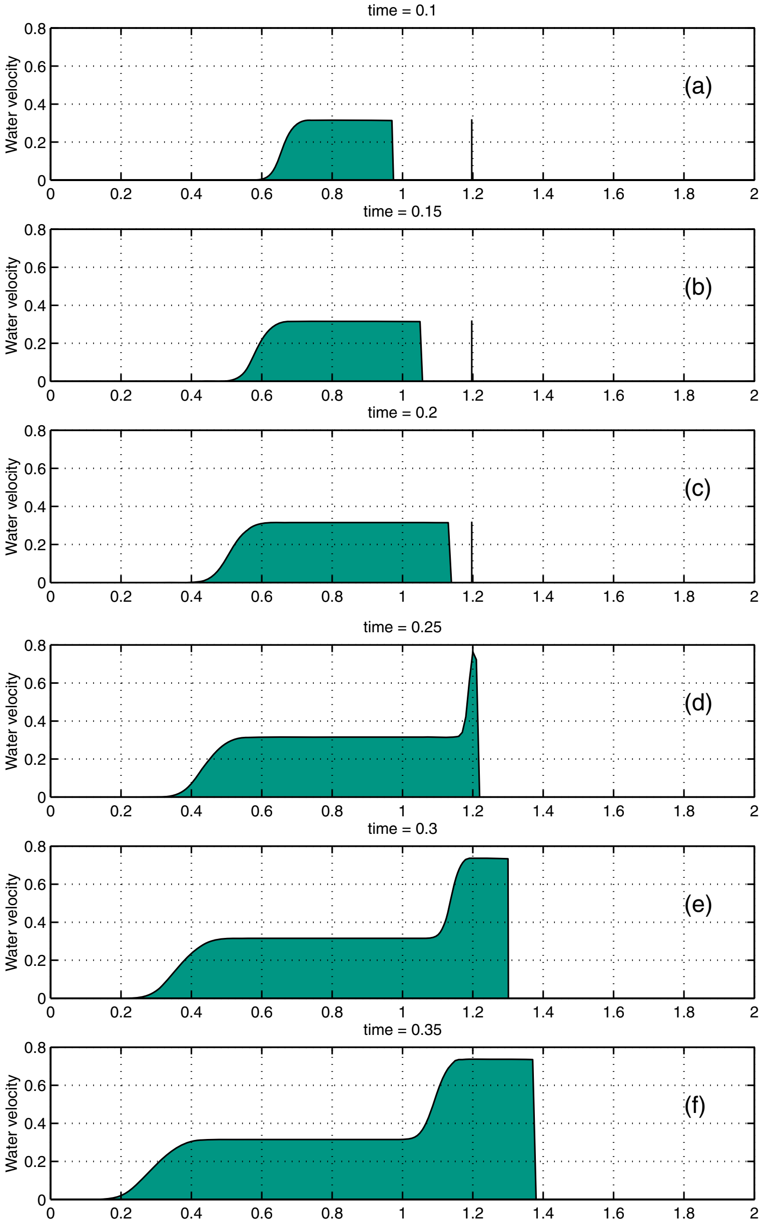


FIG. 2. *Evolution of particle velocity in the two-dam problem. Shock-adaptive Godunov scheme.* $h = 0$ *(Eulerian).* (a) $t = 0.1$; (b) $t = 0.15$; (c) $t = 0.2$; (d) $t = 0.25$; (e) $t = 0.3$; (f) $t = 0.35$.

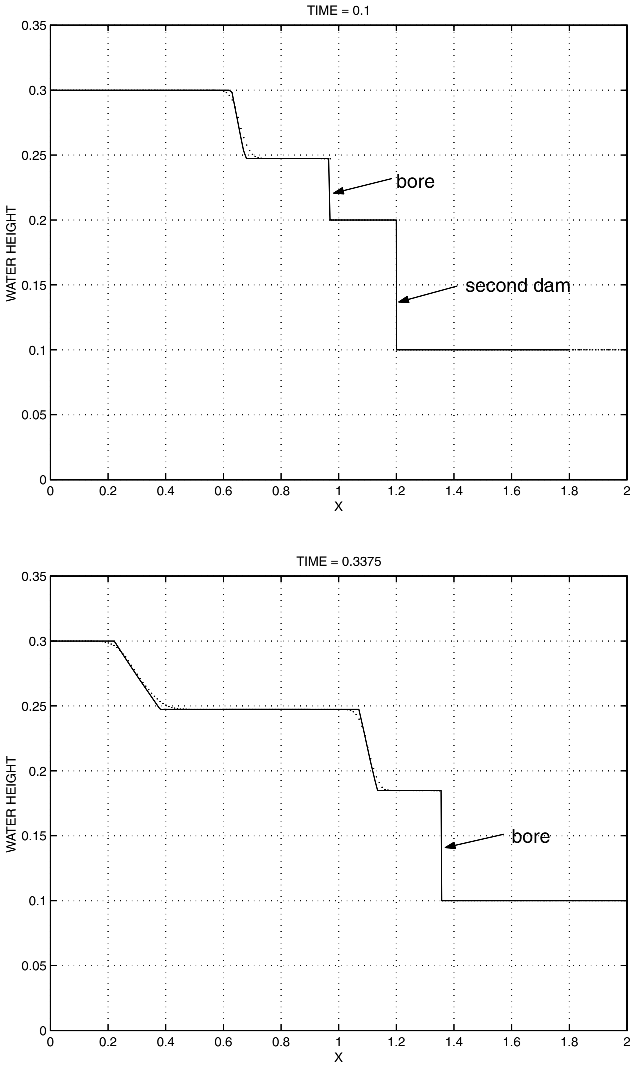


FIG. 3. *Comparison of exact (solid line) and computed (dots) solutions before (upper plot) and after (lower plot) the collapse of the second dam. Shock-adaptive Godunov scheme.* $h = 0$ *(Eulerian).*

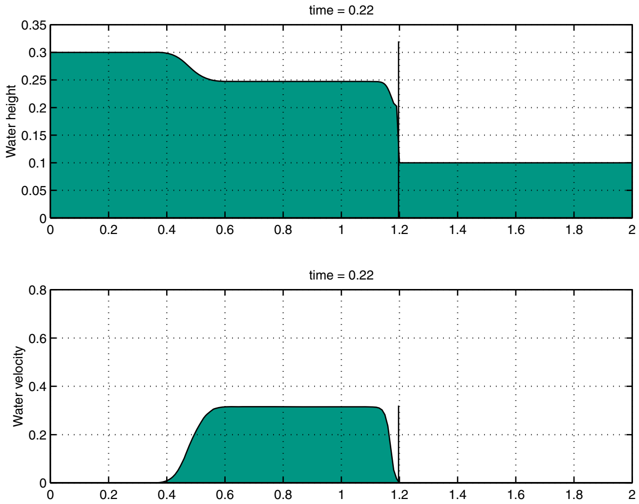


FIG. 4. *Water height and particle velocity at $t = 0.22$. Godunov scheme. $h = 0$ (Eulerian).*

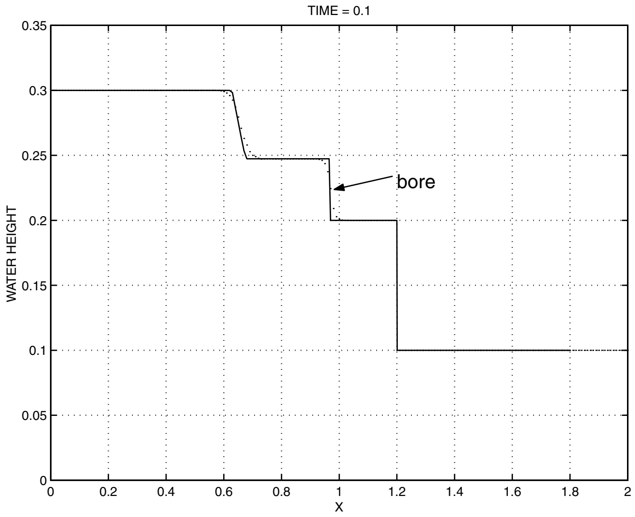


FIG. 5. *Comparison of exact (solid line) and computed (dots) solutions before the collapse of the second dam. Godunov scheme. $h = 0$ (Eulerian).*

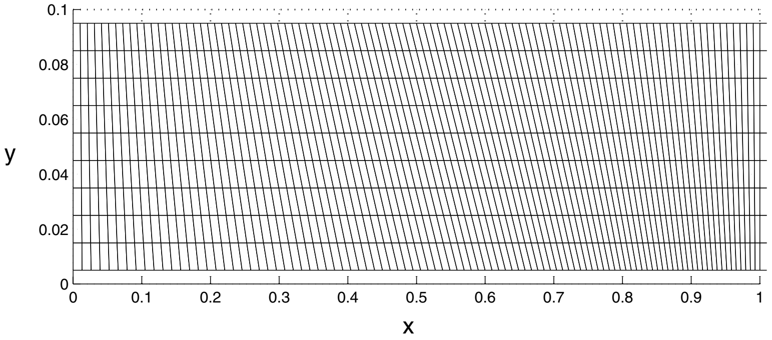


FIG. 6. *Initial grid for the Salzman test problem.*

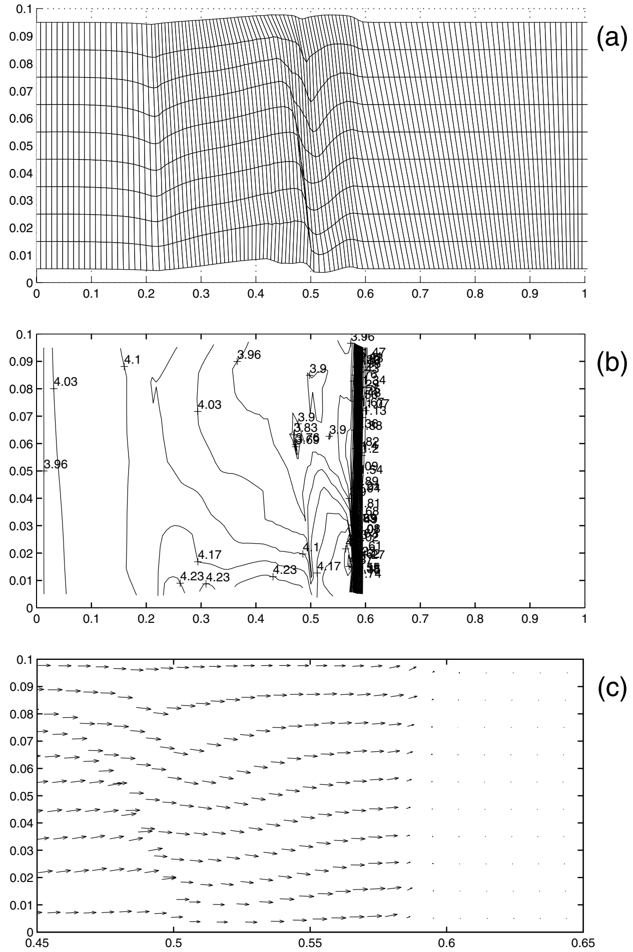


FIG. 7. *Computed solution to Salzman problem.* $t = 0.06$, $h = 0.999$. (a) *flow-generated grid;* (b) *contours of water height;* (c) *velocity distribution.*

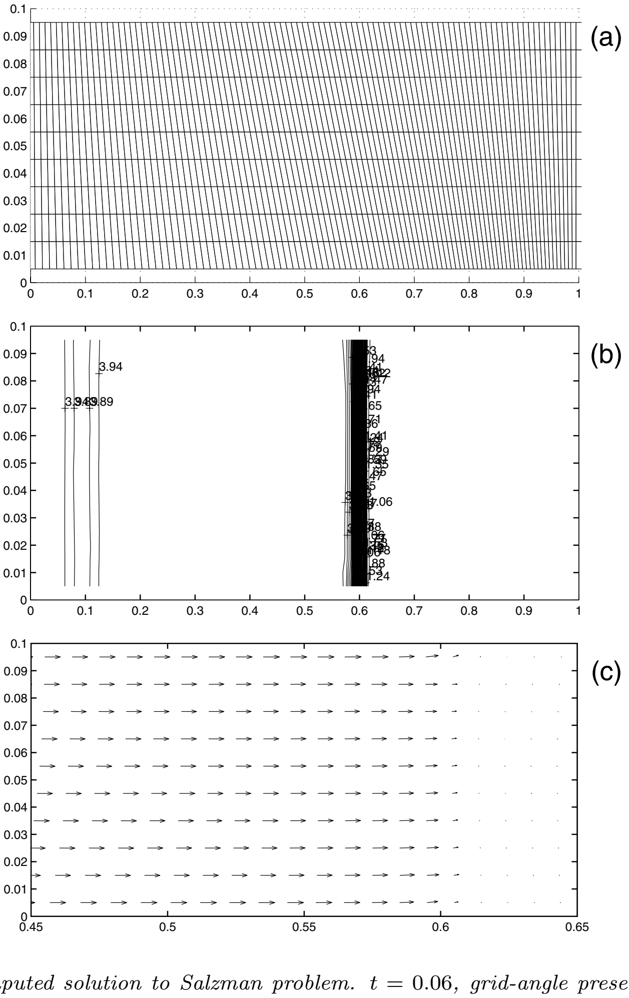


FIG. 8. *Computed solution to Salzman problem. $t = 0.06$, grid-angle preserving h.* (a) *flow-generated grid;* (b) *contours of water height;* (c) *velocity distribution.*

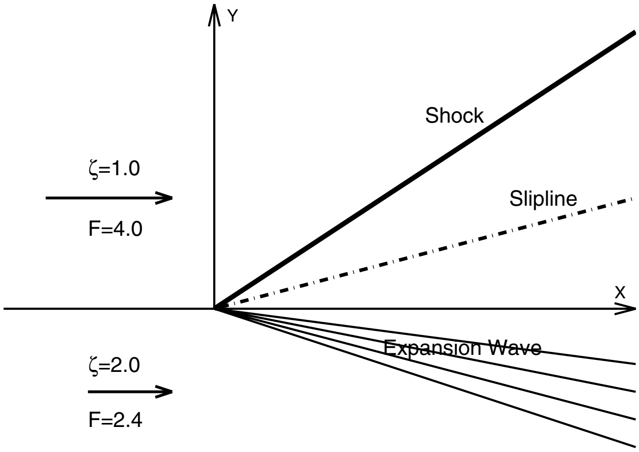


FIG. 9. *Sketch of a steady Riemann problem.*

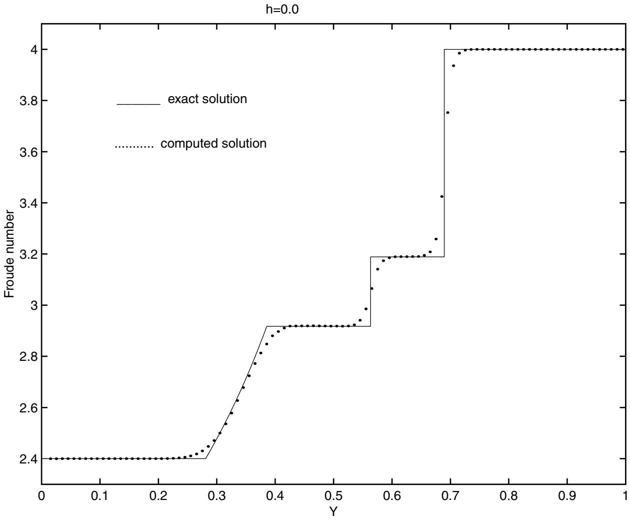


FIG. 10. *Froude number distribution in a steady Riemann problem computed by the present unified code; $h = 0$ (Eulerian).*

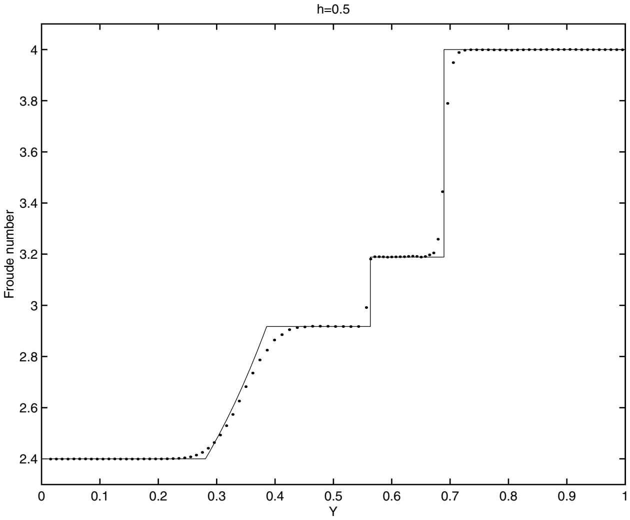


FIG. 11. *Froude number distribution in a steady Riemann problem computed by the present unified code;* $h = 0.5$.

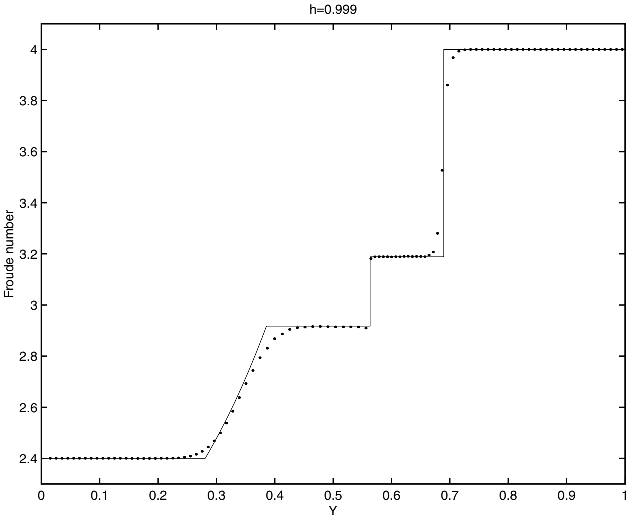


FIG. 12. *Froude number distribution in a steady Riemann problem computed by the present unified code;* $h = 0.999$.

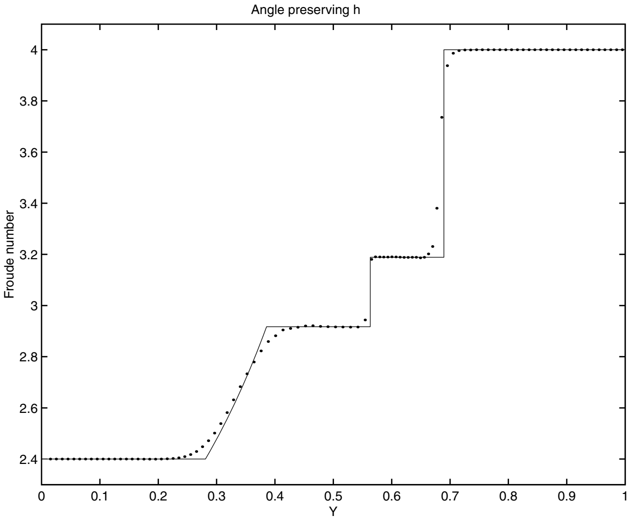


FIG. 13. *Froude number distribution in a steady Riemann problem computed by the present unified code with h chosen to preserve grid angles.*

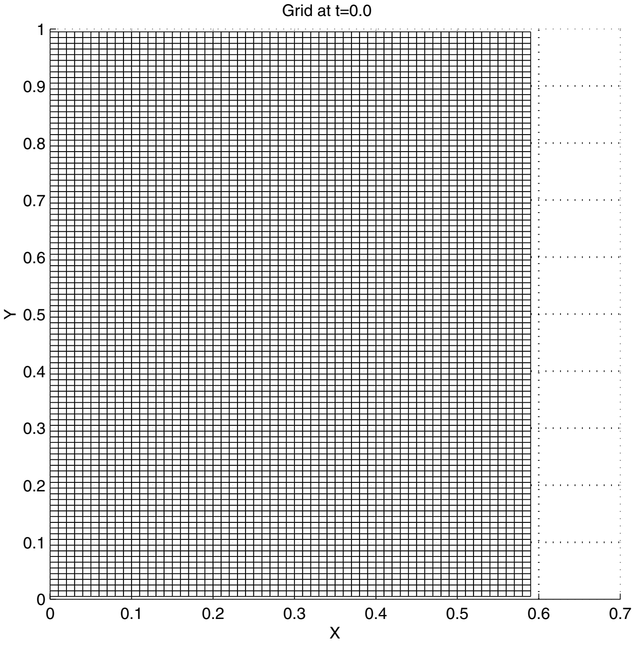


FIG. 14. *Eulerian ($h = 0$) grid, also used as initial grid for all cases in the steady Riemann problem.*

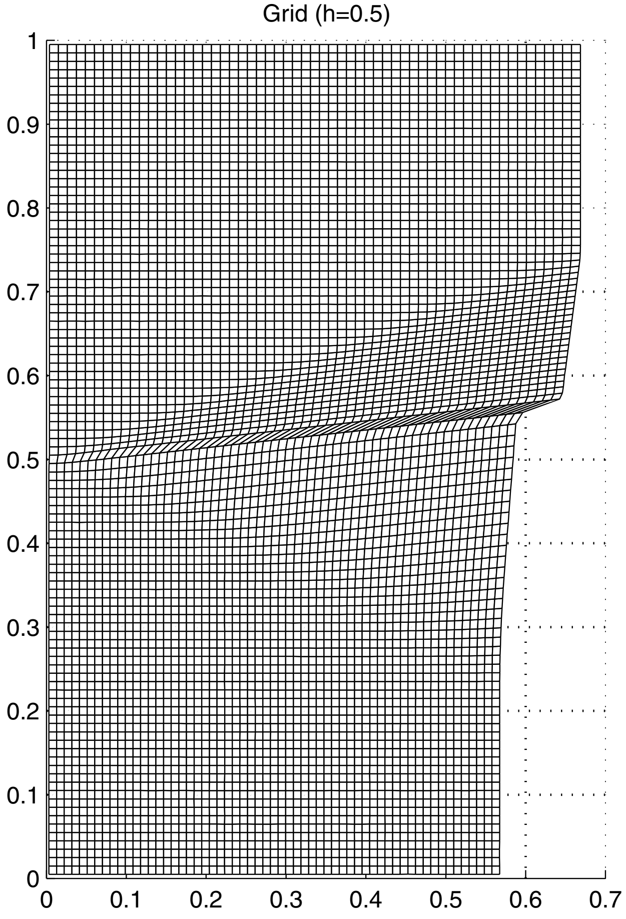


Fig. 15. *Flow-generated grid in steady Riemann problem;* $h = 0.5$.

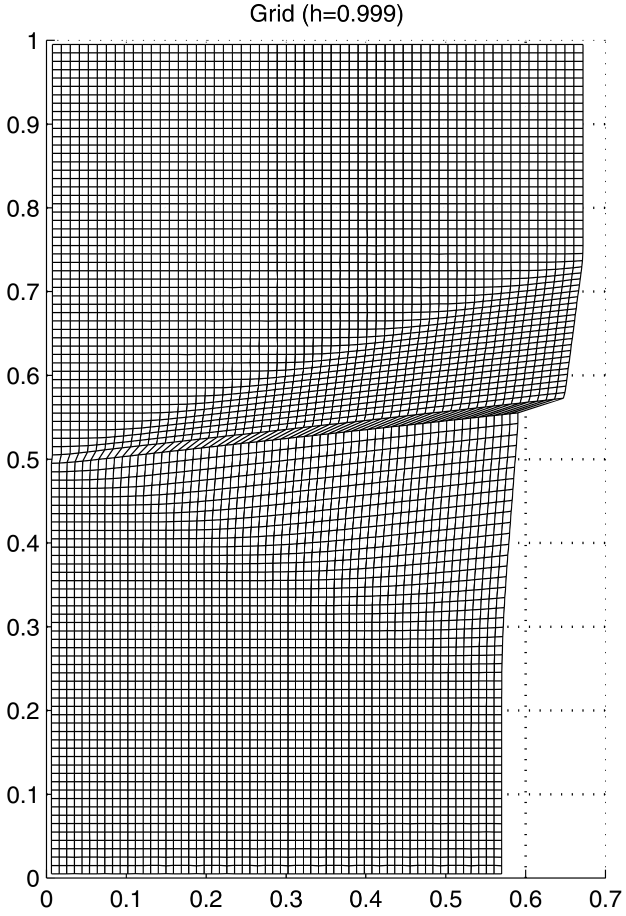


Fig. 16. *Flow-generated grid in steady Riemann problem;* $h = 0.999$.

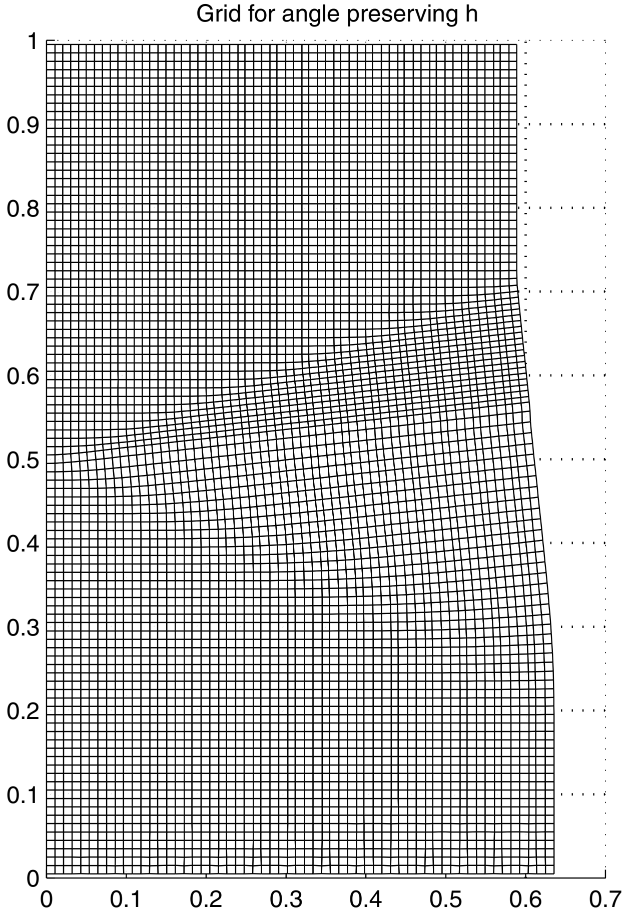


FIG. 17. *Flow-generated grid in steady Riemann problem; h chosen to preserve grid angles.*

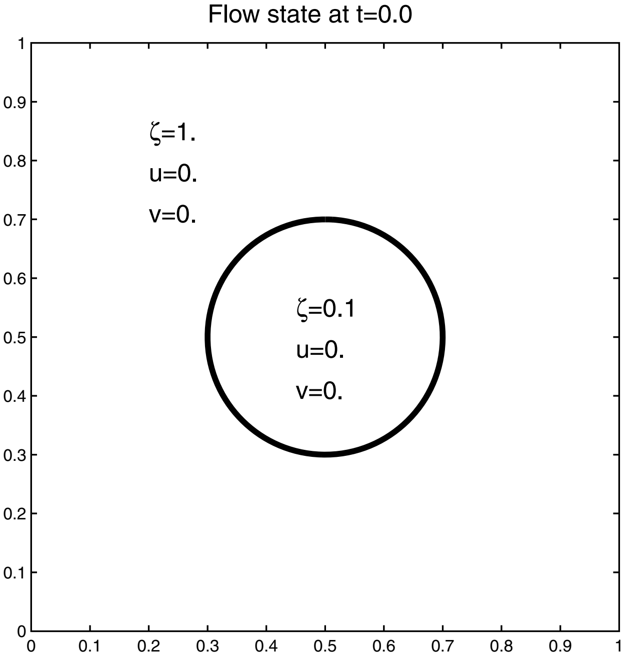


Fig. 18. *The initial state for the "implosion/explosion" problem.*

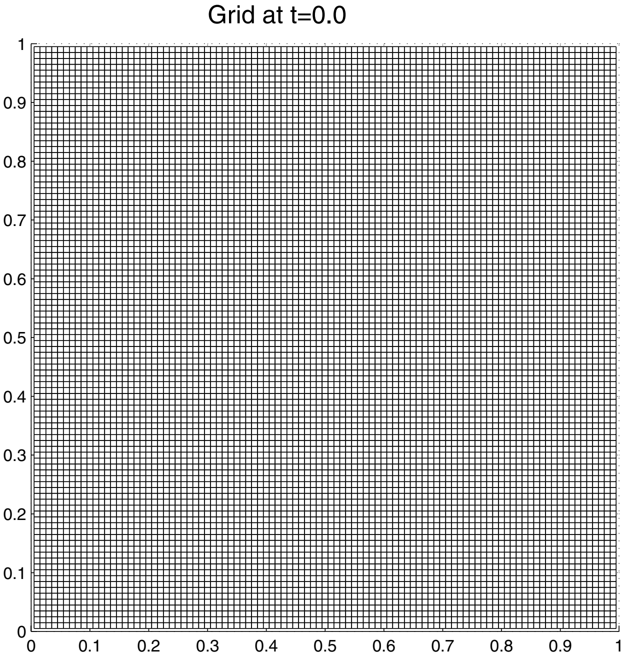


FIG. 19. *The initial grid for the "implosion/explosion" problem.*

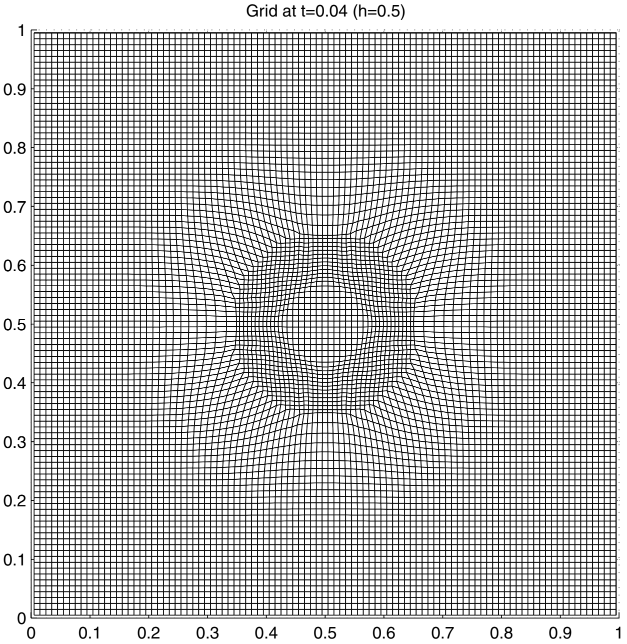


FIG. 20. *Flow-generated grid at* $t = 0.04$ *in an "implosion/explosion" problem;* $h = 0.5$*.*

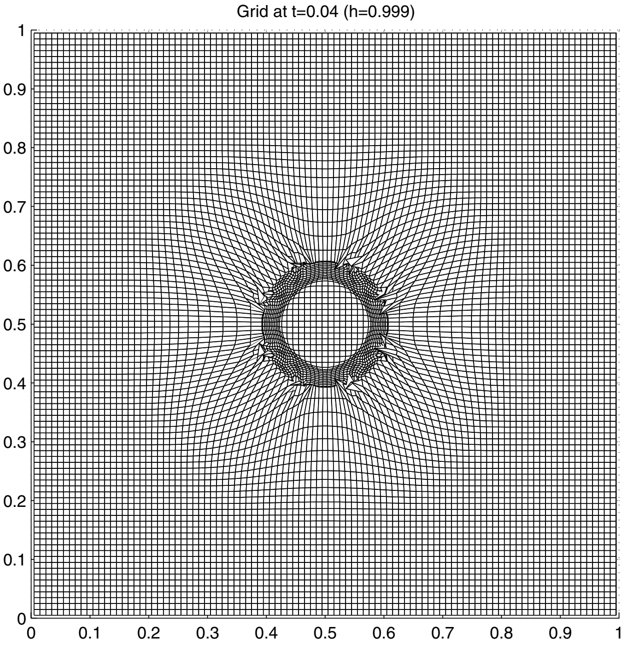


FIG. 21. *Flow-generated grid at $t = 0.04$ in an "implosion/explosion" problem; $h = 0.999$.*

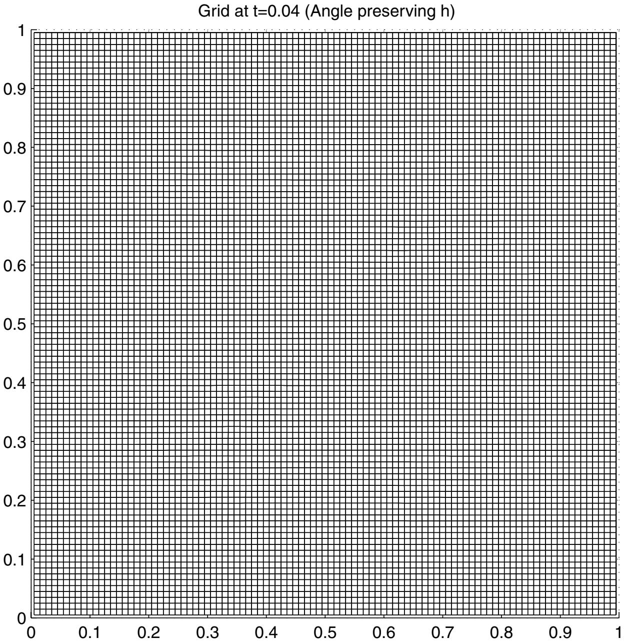


FIG. 22. *Flow-generated grid at $t = 0.04$ in an "implosion/explosion" problem; grid-angle preserving h.*

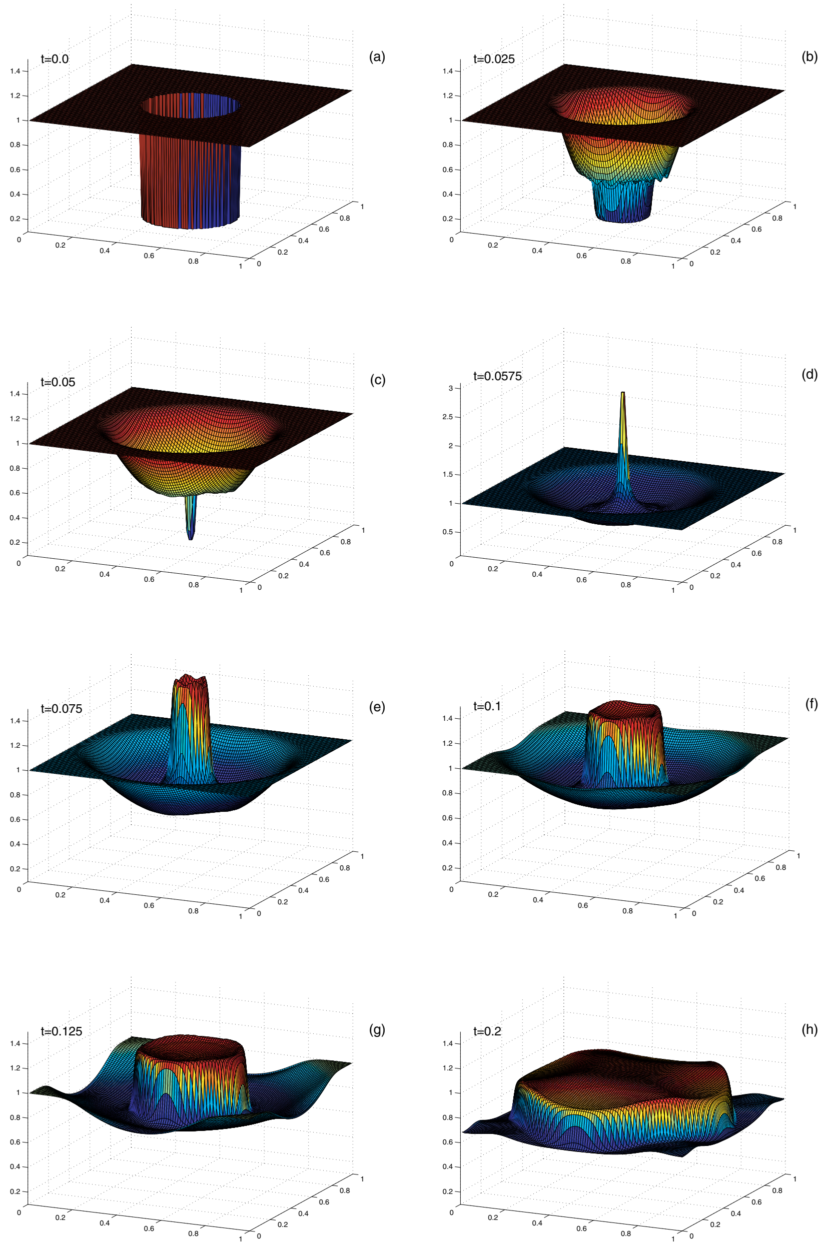


FIG. 23. *Evolution of water height with time in an "implosion/explosion" problem; grid-angle preserving h.* (a) $t = 0.0$; (b) $t = 0.025$; (c) $t = 0.05$; (d) $t = 0.0575$; (e) $t = 0.075$; (f) $t = 0.1$; (g) $t = 0.125$; (h) $t = 0.2$.

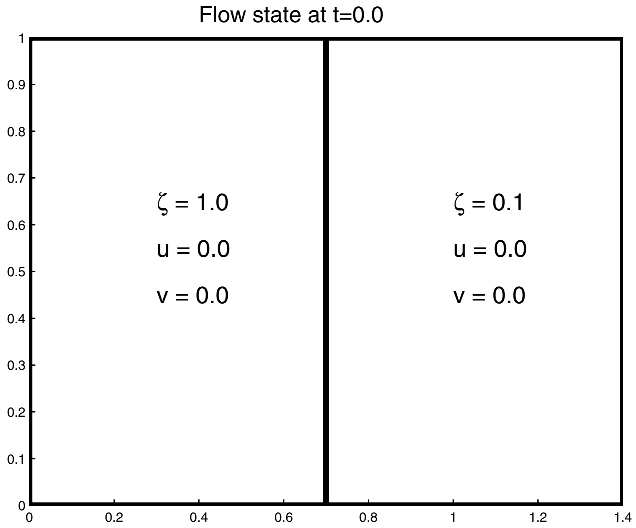


FIG. 24. *The initial state for the two-dimensional dam break problem.*

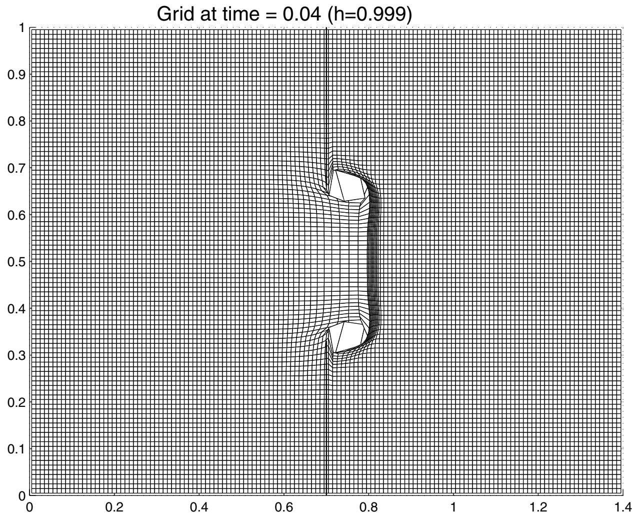


FIG. 25. *Flow-generated grid at* $t = 0.04$ *in a two-dimensional dam break problem;* $h = 0.999$.

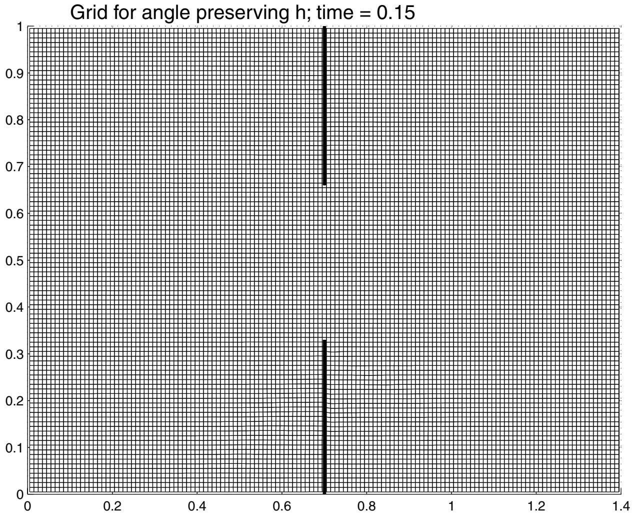


FIG. 26. *Flow-generated grid at $t = 0.15$ in a two-dimensional dam break problem; grid-angle preserving h.*

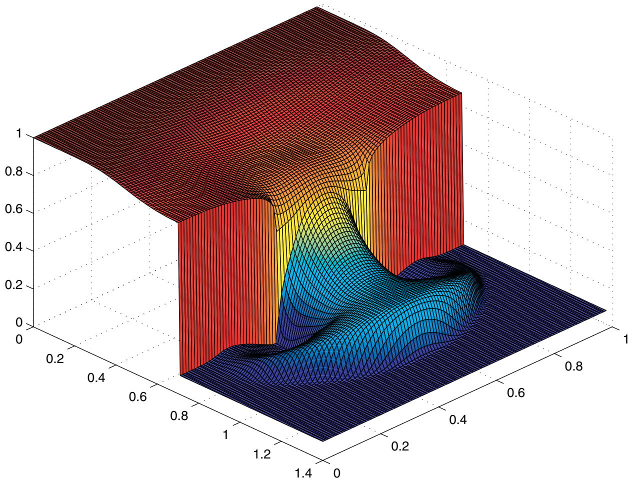


FIG. 27. *Water height at $t = 0.15$ after breaking of the dam in a two-dimensional dam break problem; grid-angle preserving h.*

REFERENCES

[1] W. H. HUI, P. Y. LI, AND Z. W. LI, *A unified coordinate system for solving the two-dimensional Euler equations*, J. Comput. Phys., 153 (1999), pp. 596–637.

[2] A. HARTEN, *The artificial compression method for computation of shocks and contact discontinuities* III: *Self-adjusting hybrid schemes*, Math. Comp., 32 (1978), pp. 363–389.

[3] A. HARTEN, *ENO schemes with subcell resolution*, J. Comput. Phys., 83 (1989), pp. 148–184.

[4] C. W. SHU AND S. OSHER, *Efficient implementation of essentially non-oscillatory shock-capturing schemes* II, J. Comput. Phys., 83 (1989), pp. 32–78.

[5] R. FEDKIW, T. ASLAM, B. MERRIMAN, AND S. OSHER, *A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method)*, J. Comput. Phys., 152 (1999),

pp. 457–492.

[6] R. Fedkiw, T. Aslam, and S. Xu, *The ghost fluid method for deflagration and detonation discontinuities*, J. Comput. Phys., 154 (1999), pp. 393–427.

[7] D. J. Benson, *Computational methods in Lagrangian and Eulerian hydrocodes*, Comput. Methods Appl. Mech. Engrg., 99 (1992), pp. 235–394.

[8] C. W. Hirt, A. A. Amsden, and J. L. Cook, *An arbitrary Lagrangian-Eulerian computing method for all flow speeds*, J. Comput. Phys., 14 (1974), pp. 227–253.

[9] W. Pracht, *Calculating three-dimensional fluid flows at all speeds with an Eulerian-Lagrangian computing mesh*, J. Comput. Phys., 17 (1975), pp. 132–159.

[10] L. G. Margolin, *Introduction to "an arbitrary Lagrangian-Eulerian computing method for all flow speeds,"* J. Comput. Phys., 135 (1997), pp. 198–202.

[11] M. S. Hall, *A comparison of first and second order rezoned and Lagrangian Godunov solutions*, J. Comput. Phys., 90 (1990), pp. 458–485.

[12] W. H. Hui and S. Koudriakov, *The role of coordinates in the computation of discontinuities in one-dimensional flow*, Comput. Fluid Dyn. J., 8 (2000), pp. 495–510.

[13] W. H. Hui and C. Y. Loh, *A new Lagrangian method for steady supersonic flow computation, part* III: *Strong shocks*, J. Comput. Phys., 103 (1992), pp. 465–471.

[14] C. Y. Lepage and W. H. Hui, *A shock-adaptive Godunov scheme based on the generalized Lagrangian Formulation*, J. Comput. Phys., 122 (1995), pp. 291–299.

[15] H. O. Kreiss and J. Lorenz, *Initial-Boundary Value Problems and the Navier-Stokes Equations*, Academic Press, San Diego, 1989.

[16] W. H. Hui and D. L. Chu, *Optimal grid for the steady Euler equations*, Comput. Fluid Dyn. J., 4 (1996), pp. 403–426.

[17] M. G. Crandall and A. Majda, *Monotone difference approximations for scalar conservation laws*, Math. Comp., 34 (1980), pp. 1–21.

[18] G. Strang, *On the construction and comparison of difference schemes*, SIAM J. Numer. Anal., 5 (1968), pp. 506–517.

[19] E. F. Toro, *Riemann problems and the WAF method for solving the two-dimensional shallow water equations*, Philos. Trans. Roy. Soc. London Ser. A, 338 (1992), pp. 43–68.

[20] P. Glaister, *Approximate Riemann solutions of the shallow water equations*, J. Hydr. Research, 26 (1988), pp. 293–306.

[21] F. Alcrudo, P. Garcia-Navarro, and J.-M. Saviron, *Flux difference splitting for* 1*D open channel flow equations*, Internat. J. Numer. Methods Fluids, 14 (1992), pp. 1009–1018.

[22] J. K. Dukowicz and B. J. A. Meltz, *Vorticity errors in multidimensional Lagrangian codes*, J. Comput. Phys., 99 (1992), pp. 115–134.

[23] E. J. Caramana and M. J. Shashkov, *Elimination of artificial grid distortion and hourglass-type motions by means of Lagrangian subzonal masses and pressures*, J. Comput. Phys., 142 (1998), pp. 521–561.

[24] J. Y. Yang, S. H. Chang, and W. H. Hui, *A new Lagrangian method for steady supercritical shallow flow computation*, Comput. Methods Appl. Mech. Engrg., 104 (1993), pp. 333–355.

[25] J. Y. Yang, C. A. Hsu, and W. H. Hui, *A generalized Lagrangian method for solving the steady shallow water equations*, Math. Comput. Simulation, 35 (1993), pp. 43–61.

[26] F. Alcrudo and P. Garcia-Navarro, *A high-resolution Godunov-type scheme in finite volumes for the* 2*D shallow-water equations*, Internat. J. Numer. Methods Fluids, 16 (1993), pp. 489–505.

[27] M. E. Hubbard, *Multidimensional slope limiters for MUSCL-type finite volume schemes on unstructured grids*, J. Comput. Phys., 155 (1999), pp. 54–74.

[28] K. Xu, *Unsplitting BGK-type schemes for the shallow water equations*, Internat. J. Modern Phys. C, 10 (1999), pp. 505–516.

SIAM J. Sci. Comput.
Vol. 23, No. 5, pp. 1655–1682

A PARTICLE-PARTITION OF UNITY METHOD–PART II: EFFICIENT COVER CONSTRUCTION AND RELIABLE INTEGRATION[*]

MICHAEL GRIEBEL[†] AND MARC ALEXANDER SCHWEITZER[†]

Abstract. In this paper we present a meshfree discretization technique based only on a set of irregularly spaced points $x_i \in \mathbb{R}^d$ and the partition of unity approach. In this sequel to [M. Griebel and M. A. Schweitzer, *SIAM J. Sci. Comput.*, 22 (2000), pp. 853–890] we focus on the cover construction and its interplay with the integration problem arising in a Galerkin discretization. We present a hierarchical cover construction algorithm and a reliable decomposition quadrature scheme. Here, we decompose the integration domains into disjoint cells on which we employ local sparse grid quadrature rules to improve computational efficiency. The use of these two schemes already reduces the operation count for the assembly of the stiffness matrix significantly. Now the overall computational costs are dominated by the number of the integration cells. We present a regularized version of the hierarchical cover construction algorithm which reduces the number of integration cells even further and subsequently improves the computational efficiency. In fact, the computational costs during the integration of the nonzeros of the stiffness matrix are comparable to that of a finite element method, yet the presented method is completely independent of a mesh. Moreover, our method is applicable to general domains and allows for the construction of approximations of any order and regularity.

Key words. meshfree method, gridless discretization, partition of unity method, Galerkin method, sparse grids, numerical integration, mesh generation

AMS subject classifications. 65C20, 65N30, 65D30, 65N50

PII. S1064827501391588

1. Introduction.

Meshfree methods (MMs) are promising approaches to overcome the problem of mesh generation, which is still the most time-consuming part of any finite element (FE) simulation. MMs are based only on a (finite) collection of independent points within the domain of interest; i.e., there are no fixed connections between any two points as in a conventional mesh. These points can now be used as collocation nodes [1, 9, 10, 11, 19] for the construction of approximate densities [21, 22, 23] or even for the construction of trial and test spaces for a Galerkin method [2, 3, 4, 8, 13].

The shape functions of a meshfree Galerkin method are in general more complex than FE shape functions. In a MM the shape functions are (in general) piecewise rational functions, whereas in a finite element method (FEM) they are piecewise polynomials. This is due to the fact that the construction of a meshfree shape function is based only on independent points instead of a mesh. Therefore, the integration of meshfree shape functions is more complicated than the integration of FE shape functions. Hence, the assembly of the stiffness matrix and right-hand side vector in a meshfree Galerkin method is far more expensive than in the FEM. Meshfree Galerkin methods therefore could not be applied to real world problems up until now and are considered to be in an experimental state only.

In this paper we present a numerical method based only on a set of irregularly spaced points and the partition of unity (PU) approach [2]. In a partition of unity

[*] Received by the editors July 2, 2001; accepted for publication (in revised form) September 7, 2001; published electronically January 30, 2002.
http://www.siam.org/journals/sisc/23-5/39158.html

[†] Institut für Angewandte Mathematik, Universität Bonn, Wegelerstr. 6, D-53115 Bonn, Germany (griebel@iam.uni-bonn.de, schweitz@iam.uni-bonn.de).

method (PUM), we define a global approximation u_{PU} simply as a weighted sum of local approximations u_i,

$$u_{\mathrm{PU}}(x) := \sum_{i=1}^{N} \varphi_i(x) u_i(x).$$

These local approximations u_i are completely independent of each other; i.e., the local supports $\omega_i := \mathrm{supp}(u_i)$, the local basis $\{\psi_i^k\}$, and the order of approximation p_i for every single $u_i := \sum u_i^k \psi_i^k$ can be chosen independently of all other u_j. Here, the functions φ_i form a PU. They are used to splice[1] the local approximations u_i together in such a way that the global approximation u_{PU} benefits from the local approximation orders p_i, yet it still fulfills global regularity conditions; see [13]. For a general partial differential equation (PDE) $Lu = f$ the fully assembled approximation functions $\varphi_i \psi_i^k$ have to be used within the Galerkin procedure. Hence, for the approximation of a PDE we have to integrate the product functions $\varphi_i \psi_i^k$ in the assembly of the stiffness matrix. This integration is one major issue of concern with PUMs. The dominant factor for the approximation quality of the method is certainly the local basis function ψ_i^k, but, other than with the p-version of the FEM, this high order function is *not* the cause of concern during the integration—the PU functions φ_i are.

The algebraic structure of the functions φ_i is (in general) more complex than that of FE shape functions, since the φ_i have to repair any spatial irregularity induced by the overlaps of the ω_i. Hence, the construction of a PU with a simple algebraic structure is the most crucial step in a PUM. Therefore, the design and implementation of a PUM for general covers $C_\Omega := \{\omega_i \,|\, i = 1, \dots, N\}$ including a fast, yet reliable quadrature scheme is quite involved.

One approach to utilize at least some of the PUM benefits is the so-called generalized finite element method (GFEM) [7, 30]. Here, the construction of the PU $\{\varphi_i\}$ is left to an h-version FEM. On top of this PU, local basis functions ψ_k^i can still be selected with all the freedom the PU approach allows for, e.g., ψ_k^i which are adapted to known local behavior of the solution. However, the dependence on the h-mesh construction for the PU is of course a major drawback of the GFEM. Furthermore, one needs to supply appropriate quadrature schemes for the reliable integration of these general local basis functions ψ_k^i independent of the facts that the cover is a mesh and that the PU is piecewise linear.

Truly meshfree Galerkin methods [8, 13] have to be concerned with the construction of a cover from a given set of points $P = \{x_i \in \Omega\}$ and hence have to cope with geometric searching and sorting problems. To tackle these problems an algorithm for finding a set covering a single point based on a tree concept was proposed in [17]. However, the cover C_Ω was still assumed to be given.

In this paper we present a general cover construction algorithm based only on a set of irregularly spaced points $P = \{x_i \in \Omega\}$. We partition the domain into overlapping d-rectangular patches ω_i which we assign to the given points x_i to cover the complete domain. We use d-binary trees (binary trees, quadtrees, octrees) for the construction of these patches ω_i. While the data structures used here are similar to those used

[1] Due to this splicing property of the PU one is tempted to construct the global solution u_{PU} by solving only local problems for the u_i on the support patches ω_i. Here, one would use the local basis functions ψ_i^k as trial and test functions and disregard the partition of unity functions φ_i during the construction of the global solution u_{PU}. For the mass matrix problem—and related interpolation problems—this is a proper approach which directly leads to a system of linear equations already in block diagonal form without any lumping.

in [17], we are interested in the *construction of a cover* C_Ω with desirable features. For instance, the subsets $N_x \subset C_\Omega$ of cover patches ω_k which cover a point $x \in \Omega$, i.e., $N_x = \{\omega_k \in C_\Omega \,|\, x \in \omega_k\}$, should be small. Although a cover C_Ω generated by this general algorithm is minimal in the sense that $\mathrm{card}(N_x) \ll \mathrm{card}(C_\Omega)$ is very small for all $x \in \Omega$, the resulting PU functions φ_i are (in general) still more complex than in the GFEM. The piecewise character of the constructed functions φ_i, however, can be further significantly simplified by making slight changes to the general cover construction. With this refined algorithm we construct covers for general domains that stay close to k-irregular grids. This construction not only minimizes the number of patches $\mathrm{card}(N_x)$ which cover a single point $x \in \Omega$ but also leads to PU functions φ_i with simpler algebraic structure. A k-irregular grid is completely sufficient for a PUM, since the φ_i will repair the jump within the spatial resolution and ensure the global regularity conditions imposed on the approximation u_{PU}. At the same time, the cost of the assembly of the stiffness matrix and right-hand side vector is significantly reduced, since the simpler algebraic structure of the φ_i allows for cheaper quadrature rules.

Furthermore, we introduce a numerical quadrature scheme for the fully assembled approximation functions $\varphi_i \psi_i^k$ which further reduces the integration costs. Here, we decompose the integration domain to resolve the algebraic structure of the PU functions φ_i; i.e., we decompose the integration domains into disjoint cells on which the integrands are smooth functions. We then use a sparse grid quadrature rule [12] with a dynamic stopping criterion locally on the cells. Hence, the number of integration points on each cell is minimal with respect to accuracy. It turns out that overall, the number of operations needed by our method during the assembly of the stiffness matrix is comparable to that of a FEM. Yet at the same time it is truly a MM; i.e., it is completely independent of a mesh. Hence, our PUM is a very flexible and efficient numerical discretization technique, and it is a strong competitor for conventional FEMs. With the proposed method the treatment of real world problems using meshfree Galerkin methods might now be in reach.

The remainder of the paper is organized as follows. In section 2 we give a short review of the construction of PU spaces for meshfree Galerkin methods. We then present in section 3 a general hierarchical cover construction algorithm based only on a set of irregularly spaced points which allows for a fast neighbor search. Here, we make use of d-binary trees (quadtrees, octrees, etc.) to assign parts of the domain to each of the given points to cover the complete domain. The Galerkin discretization of a PDE using PUM shape functions is given in section 4. In section 5 we introduce an appropriate numerical quadrature scheme for PUM shape functions. The scheme is based on a decomposition approach to resolve the piecewise character of the PU functions φ_i. On the cells of this decomposition we employ local sparse grid quadrature rules with a dynamic stopping criterion. This reduces the computational costs on each cell substantially, yet still ensures a reliable accuracy of the integration. In section 6 we present a refinement of the general cover construction algorithm given in section 3; a similar cover construction algorithm was recently proposed in [18]. It also accounts for the geometric neighboring relations of the PU functions φ_i, i.e., the neighboring relations of the cover patches ω_i, which have a significant effect on the number of integration cells. With this improved algorithm the number of integration cells and the corresponding computational effort during the integration of the stiffness matrix entries is significantly reduced. Numerical results for elliptic problems in two and three dimensions are given in section 7.

2. Construction of trial and test spaces. In the following, we give a short recap of how to construct PU spaces for a meshfree Galerkin method; see [13] for details. The starting point for any MM is a collection of N independent points $P := \{x_i \in \mathbb{R}^d \,|\, x_i \in \overline{\Omega}, i = 1, \dots, N\}$. In the PU approach we need to construct a PU $\{\varphi_i\}$ on the domain of interest Ω to define an approximate solution

$$u_{\mathrm{PU}}(x) := \sum_{i=1}^{N} \varphi_i(x) u_i(x), \tag{2.1}$$

where the union of the supports $\operatorname{supp}(\varphi_i) = \overline{\omega_i}$ covers the domain $\overline{\Omega} \subset \bigcup_{i=1}^{N} \omega_i$ and $u_i \in V_i^{p_i}(\omega_i)$ is some locally defined approximation of order p_i to u on ω_i. Given a cover $C_\Omega = \{\omega_i \,|\, i = 1, \dots, N\}$ we can then define such a PU and local approximations u_i by using *Shepard functions* as φ_i and local approximation spaces $V_i^{p_i}$ on the patches ω_i.

A naive approach toward the construction of such a cover C_Ω would be the design of patches $\tilde{\omega}_i$ in such a way that every given point $x_i \in \tilde{\omega}_j$ for some $j \neq i$. However, this procedure (in general) does not lead to a cover of the complete domain Ω, i.e., $\bigcup_j \tilde{\omega}_j \not\supset \overline{\Omega}$, since the points $x_i \in P$ may not be uniformly distributed in the domain Ω. In [13] the following algorithm was proposed which tackles this problem by using a set $\tilde{P} = P \cup Q$ of original points $P = \{x_i\}$ and additional (user supplied) points $Q = \{\xi_k\}$. Note that the additional points ξ_k are introduced to guarantee that the patches ω_i completely cover the entire domain Ω. However, they are *not* used to construct additional cover patches; i.e., the algorithm constructs a cover patch ω_i only for $x_i \in P$ not for $\xi \in \tilde{P} \setminus P$.

Algorithm 1 (direct cover construction).

1. Given: the domain $\Omega \subset \mathbb{R}^d$, a scalar $\alpha \geq 1$, the set of points $P = \{x_i \in \mathbb{R}^d \,|\, x_i \in \overline{\Omega}, i = 1, \dots, N\}$ for the PU construction and a set of points $Q = \{\xi_i \in \mathbb{R}^d \,|\, \xi_i \in \overline{\Omega}\}$ to resolve the domain Ω.
2. Set $\tilde{P} = P \cup Q$.
3. For all $x_k \in P$: Set $\omega_k := \bigotimes_{i=1}^{d} [x_k^i - h_k^i, x_k^i + h_k^i]$ with $h_k^i = 0$ for all $i = 1, \dots, d$.
4. For all $y \in \tilde{P}$:
   - (a) Set $R = 0$. Evaluate the set $S_{y,R}$ of all points $x_k \in P$ that fall within a searching square B_R which is centered in y and whose side length is equal to $2R$. If $S_{y,R} = \emptyset$ (or $S_{y,R} = \{y\}$ if $y = x_k$), increase the size of the searching square, i.e., R, and try again.
   - (b) Compute the distances $d_{y,k} := \|y - x_k\|$ for all $x_k \in S_{y,R}$ with $x_k \neq y$.
   - (c) Determine a point $x_l \neq y$ with $d_{y,l} = \min_k d_{y,k}$.
   - (d) If $y \notin \omega_l$ increase h_l^i for all $i = 1, \dots, d$ appropriately such that $y \in \omega_j$ holds.
5. For all $x_k \in P$: Set $\omega_k = \alpha\omega_k$, i.e., $h_k^i = \alpha h_k^i$ for all $i = 1, \dots, d$.

Now that we have found a cover C_Ω of the domain Ω, we construct a PU $\{\varphi_i\}$ by defining weight functions W_i on the cover patches ω_i. From these weight functions W_i we can easily generate a PU by Shepard's method; i.e., we define

$$\varphi_i(x) = \frac{W_i(x)}{\sum W_k(x)}. \tag{2.2}$$

Since the cover patches ω_i constructed by Algorithm 1 are d-rectangular, i.e., they are products of intervals, the most natural choice for a weight function W_i is a product of one-dimensional functions; i.e., $W_i(x) = \prod_{l=1}^{d} W_i^l(x^l) = \prod_{l=1}^{d} \mathcal{W}(\frac{x - x_i^l + h_i^l}{2h_i^l})$, with $\mathrm{supp}(\mathcal{W}) = [0,1]$, such that $\mathrm{supp}(W_i) = \overline{\omega_i}$. It is sufficient for this construction to choose a one-dimensional weight function $\mathcal{W}$ which is nonnegative. Throughout this paper we use normed B-splines [27] as the generating weight function $\mathcal{W}$.

In general, a PU $\{\varphi_i\}$ can of course recover only the constant function on the domain Ω. Hence, the consistency error in the L^2-norm of a discretization with the $\{\varphi_i\}$ would be of first order only. Therefore, we need to improve the approximation quality to use the method for the discretization of a PDE. To this end, we multiply the PU functions φ_i locally with polynomials. Since we use d-rectangular patches ω_i only, a local tensor product space is the most natural choice. Throughout this paper, we use complete Legendre polynomials[2] as local approximation spaces $V_i^{p_i}$; i.e., we choose $V_i^{p_i} = \mathrm{span}\,(\{\psi_i^k \,|\, \psi_i^k = \prod_{l=1}^{d} \mathcal{L}_i^{k_l}, \sum_{l=1}^{d} k_l \leq p_i\})$, where $\mathcal{L}_i^{k_l}$ is the one-dimensional Legendre polynomial of degree k_l on the interval $[x_i^l - h_i^l, x_i^l + h_i^l]$.

The selection of optimal local approximation orders p_i and basis functions ψ_i^k are by nature problem-dependent. The regularity of the analytical solution space, e.g., $H^k(\Omega)$ or $L^p(\Omega)$, and information about the analytical solution u itself may provide some insight. This and other adaptivity related issues are the subject of future research and will not be treated in this paper.

Following the construction given above, we can construct approximate solutions u_{PU} of any order and regularity without additional constraints on the cover C_Ω. The resulting shape functions $\varphi_i\psi_i^k$, however, have some surprising properties.

1. The PU functions φ_i are (in general) noninterpolatory. Furthermore, there are more degrees of freedom in a PUM space than there are points $x_i \in P$ due to the use of (multidimensional) local approximation spaces $V_i^{p_i}$.
2. The regularity of the shape functions $\varphi_i\psi_i^k$ is independent of the number of degrees of freedom. The shape functions inherit the regularity of the PU functions φ_i (if we assume that the local approximation spaces $V_i^{p_i}$ are at least of the same regularity). Therefore, we can increase the regularity of an approximation u_{PU} by changing the B-spline used in (2.2) independent of the local approximation spaces $V_i^{p_i}$. Note that this is different from FEMs. In the FEM the global regularity of an approximation is given by the element regularity which on the other hand is implemented by constraints imposed on the local degrees of freedom. Hence, a higher regularity may be achieved only by increasing the number of degrees of freedom of an element.
3. The PUM shape functions are piecewise rational functions due to the use of piecewise polynomial weights in (2.2).

The cover C_Ω itself influences the computational work of the method significantly. For one, the neighbor relations $\omega_i \cap \omega_j \neq \emptyset$ of the cover C_Ω already define the sparsity pattern of the stiffness matrix. Second, the evaluation of a single PU function φ_i (see (2.2)) involves the evaluation of the weights of all neighboring patches $N_i := \{\omega_j \in C_\Omega \,|\, \omega_i \cap \omega_j \neq \emptyset\}$. Hence, it is necessary to control the number of neighbors $\mathrm{card}(N_i)$ to limit the computational work during the assembly of the stiffness matrix. Furthermore, the smoothness of a PU function φ_i is strongly dependent on the amount of overlap $\omega_i \cap \omega_j$ of the neighboring patches $\omega_j \in N_i$; see [13] for details.

[2] Note that due to their product structure the shape functions $\varphi_i\psi_i^k$ of our PUM do not inherit orthogonality properties of the local basis functions ψ_i^k. Hence, the chosen Legendre polynomials $\mathcal{L}^k$ will not lead to a diagonal matrix for the mass matrix problem, as well as the integrated Legendre polynomials $\mathcal{L}_I^k$ will not lead to a diagonal matrix for the Poisson problem.

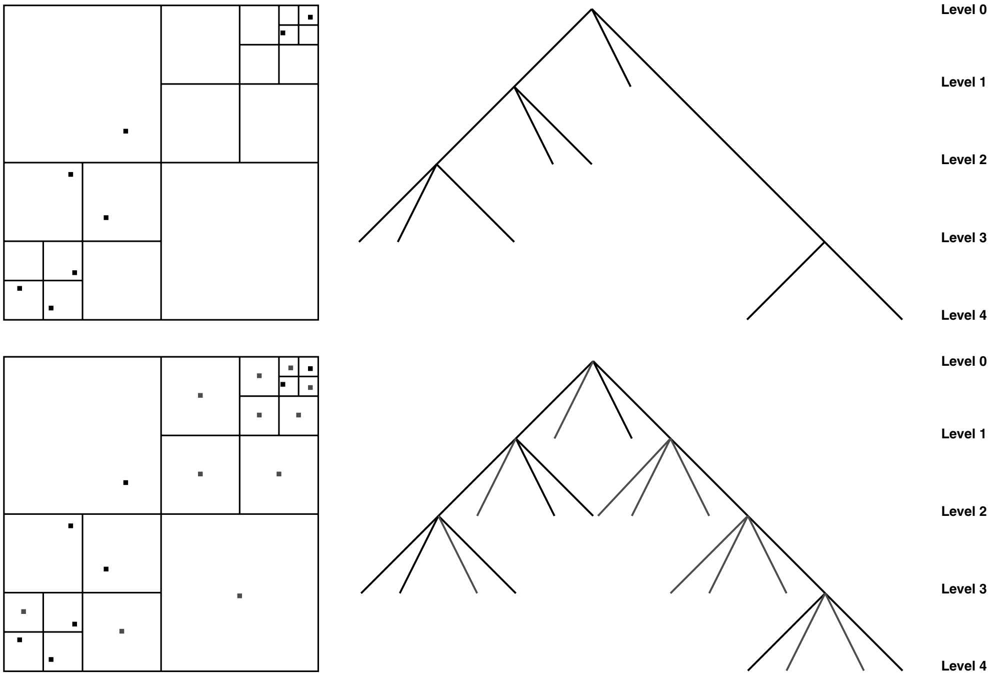


FIG. 3.1. *Hierarchical cover construction with Algorithm* 2 *in two dimensions. The initial cell decomposition induced by P (upper left) and its corresponding tree representation (upper right) after step* 2 *of Algorithm* 2*. The final cell decomposition (lower left) and its tree representation after the completion of Algorithm* 2*.*

3. Hierarchical cover construction.

Therefore, we need to construct a cover C_Ω which minimizes the number of neighbors $\text{card}(N_i)$ for each patch ω_i but ensures significantly large overlaps $\omega_i \cap \omega_j$ to allow for the use of a cheaper quadrature scheme for each nonzero entry of the stiffness matrix.

With Algorithm 1 for the cover construction the control over the neighborhoods N_i is somewhat limited. Hence, it is very difficult to limit the density of the stiffness matrix. Even more problematic though is the fact that Algorithm 1 needs an additional input[3] Q besides the set of points P and the domain $\Omega \subset \mathbb{R}^d$ to ensure that the complete domain is indeed covered by C_Ω. This of course makes Algorithm 1 significantly less useful, especially in time-dependent settings.

In the following we propose a new algorithm which employs a decomposition approach for the domain Ω to assign sets $\omega_i \subset \mathbb{R}^d$ to the points $x_i \in P$ in such a way that they cover the domain $\Omega \subset \bigcup \omega_i$. This hierarchical algorithm does not need an additional input Q.

ALGORITHM 2 (hierarchical cover construction).

1. Given: the domain $\Omega \subset \mathbb{R}^d$, a bounding box $R_\Omega = \bigotimes_{i=1}^d [l_\Omega^i, u_\Omega^i] \supset \overline{\Omega}$, the initial point set $P = \{x_j \in \mathbb{R}^d \,|\, x_j \in \overline{\Omega}\}$, and a parameter $k \in \mathbb{N}$.
2. Build a d-binary tree (quadtree, octree) over R_Ω such that per leaf L at most one $x_i \in P$ lies within the associated cell $C_L := \bigotimes_{i=1}^d [l_L^i, u_L^i]$, and the difference of the levels of two adjacent cells is at most k; see Figure 3.1.

[3] The selection of the set Q within Algorithm 1 is quite involved to ensure the sparsity of the stiffness matrix and to cover the complete domain at the same time if the points $x_i \in P$ are nonuniformly distributed.

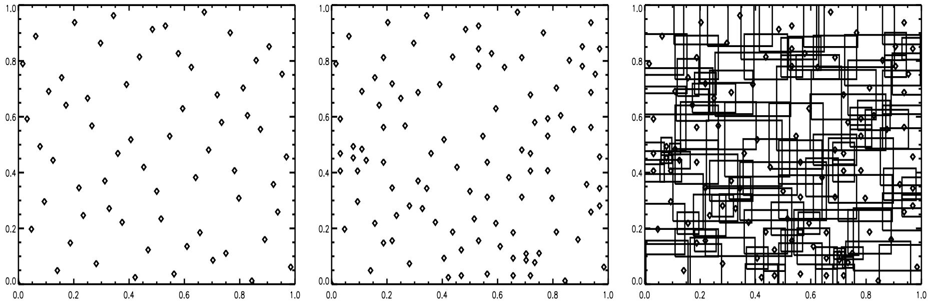


Fig. 3.2. *$P = Halton_0^{63}(2,3)$ point set in $R_\Omega = \Omega = [0,1]^2$ (left), P with* $\mathrm{card}(P) = 106$ *(center) after Algorithm* 2 *with $k = \infty$ and $x_L^i = l_L^i + \frac{1}{2}(u_L^i - l_L^i)$ for generated points x_L, and the generated cover C_Ω (right) with $h_L^i = \frac{5}{4}\max\{u_L^i - x_L^i, x_L^i - l_L^i\}$.*

3. For all cells $C_L = \bigotimes_{i=1}^d [l_L^i, u_L^i]$ with $C_L \cap \Omega \neq \emptyset$:
   - (a) If there is an $x_j \in P$ with $x_j \in C_L$, set $x_L = x_j$. Else, set $x_L =$ any element of C_L, e.g., the center $x_L^i = l_L^i + \frac{1}{2}(u_L^i - l_L^i)$ of the cell C_L.
   - (b) Set $P = P \cup \{x_L\}$.
   - (c) Set $\omega_L = R_L = \bigotimes_{i=1}^d [x_L^i - h_L^i, x_L^i + h_L^i] \supset C_L$, where $h_L^i = \frac{\alpha}{2}\max\{u_L^i - x_L^i, x_L^i - l_L^i\}$ with $\alpha > 1$.

With Algorithm 2 we not only ensure the covering property $\Omega \subset \bigcup \omega_i$ without additional input data, but we also control the neighborhoods N_i, i.e., the nonzeros of the stiffness matrix, and ensure the smoothness of the functions φ_i. The neighborhoods N_i of the cover patches ω_i constructed by Algorithm 2 are small, yet the amount of overlap of any two neighboring patches is of significant size. Certainly, these features do come at a price we have to pay: the algorithm automatically introduces additional points x_{N+k} into the set P; see Figures 3.1 and 3.2.[4] This increases the number of unknowns, i.e., the number of rows of the stiffness matrix, and seemingly the overall computational cost. However, as it turns out, the number of nonzeros of a stiffness matrix based on our algorithm is comparable to the number of nonzeros of a stiffness matrix based on Algorithm 1 for uniformly distributed points P (see Table 6.1, where the initial point set P for the cover construction is a Halton[5] point set). And it is significantly less for highly irregular point sets P (see Table 6.2). Furthermore, the proposed algorithm enables the user to control the amounts of overlap $\omega_i \cap \omega_j$ completely by the choice of the parameter k in step 2 (which corresponds to the local imbalance of the tree), the choice of $x_L \in C_L$ in step 3(a), and the choice of α in step 3(c). Hence, this construction leads to smoother PU functions φ_i and allows for the use of cheaper quadrature schemes (compared with Algorithm 1) during the assembly of the stiffness matrix, although the functions φ_i are still more complex

[4] The additional points are necessary to ensure the shape regularity of the tree cells and patches. A similar tree-based algorithm for the construction of shape-regular triangulations with an almost-minimal number of vertices was proposed in [5]. Analogously, the presented algorithm may have a similar almost-optimal property: If m is the minimal number of shape-regular d-rectangles required to cover the given point set in such way that all d-rectangles contain at most a single point, then the cover C_Ω constructed by the presented algorithm is of size $\mathrm{card}(C_\Omega) = O(m)$.

[5] Halton sequences are quasi Monte Carlo sequences, which are used in sampling and numerical integration. Consider $n \in \mathbb{N}_0$ given as $\sum_j n_j p^j = n$ for some prime p. We can define the transformation H_p from $\mathbb{N}_0$ to $[0,1]$ with $n \mapsto H_p(n) = \sum_j n_j p^{-j-1}$. Then, the (p,q) Halton sequence with N points is defined as $\mathrm{Halton}_0^N(q,p) := \{(H_p(n), H_q(n)) \,|\, n = 0, \ldots, N\}$.

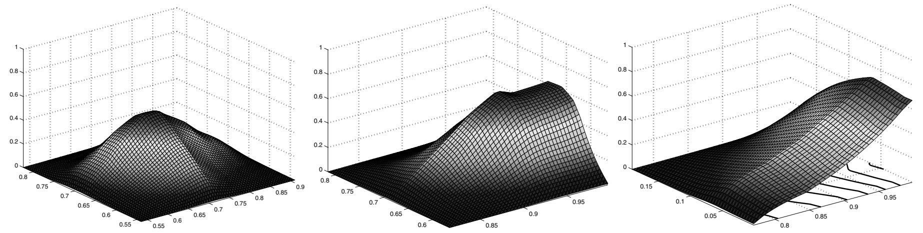


FIG. 3.3. *The PU function φ_i on $\Omega \cap \omega_i$ generated by Algorithm* 2 *with the input data from Figure* 3.2 *for an interior point (left), a boundary point (center), and a corner point (right) using linear B-splines.*

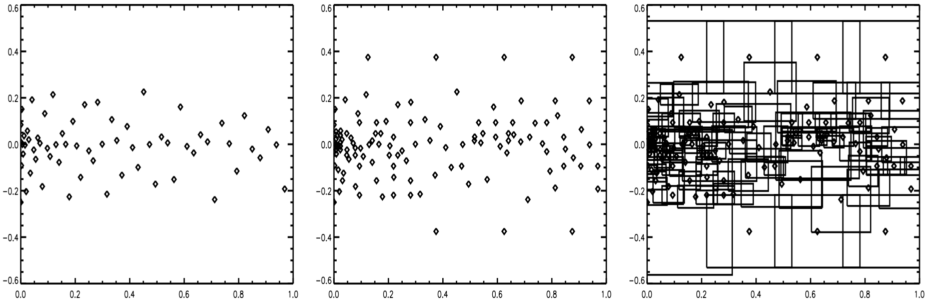


FIG. 3.4. *P is a Halton$_0^{63}(2,3)$ point set in $R_\Omega = \Omega = [0,1] \times [-0.5, 0.5]$ graded by $(x,y) \mapsto (x^2, \pm y^2)$ (left), P with* $\text{card}(P) = 121$ *(center) after Algorithm* 2 *with $k = \infty$ and $x_L^i = l_L^i + \frac{1}{2}(u_L^i - l_L^i)$ for generated points x_L, and the generated cover C_Ω (right) with $h_L^i = \frac{5}{4}\max\{u_L^i - x_L^i, x_L^i - l_L^i\}$.*

than FE shape functions (see Figure 3.3). In summary, the proposed algorithm is applicable to general domains Ω and any initial distribution of points P without an additional input Q, yet it also reduces the computational costs during the assembly of the stiffness matrix and right-hand side vector.

Note that we capture the domain Ω by the cells C_L and subsequently by the $\omega_L = R_L$ only. This, however, does *not* limit the domain or boundary resolution of our method. At this stage in the construction process, we are interested only in generating a cover C_Ω of the domain Ω. Only during the integration of the stiffness matrix and right-hand side vector entries do we need to restrict the evaluation of the associated shape functions $\varphi_L \psi_L^k$ to the computational domain Ω, i.e., to the integration domain $\omega_L \cap \Omega$. We postpone the issue of domain and boundary resolution to section 5. Note further that due to steps 3(a) and 3(c) we generally produce d-rectangular cover patches ω_i independent of the shape of the bounding box R_Ω; see Figures 3.2 and 3.4. Note also that the construction allows for the fast evaluation of a single PU function φ_i (see (2.2)) due to the efficient neighbor search in the hierarchical tree data structure. Note finally that the introduction of a hierarchical cover induces a hierarchy for the associated function space[6] which we may exploit in the design of fast multilevel solvers for the linear equations arising from a PUM discretization. This issue, however, is the subject of future research.

[6] For covers from Algorithm 1 the introduction of such a hierarchy would at least be artificial. Since in Algorithm 1 the selection of the supports ω_i is independent of a hierarchical ordering on the points, this hierarchy on the points would not lead to a hierarchy for the supports of the shape functions in a natural way.

4. Galerkin method with the PUM space.

We want to solve elliptic boundary value problems of the type

$$\begin{aligned} Lu &= f \quad \text{in } \Omega \subset \mathbb{R}^d\,, \\ Bu &= g \quad \text{on } \partial\Omega\,, \end{aligned} \tag{4.1}$$

where L is a symmetric partial differential operator of second order and B expresses suitable boundary conditions. The implementation of Neumann boundary conditions with our PUM is straightforward and similar to their treatment within the FEM. The realization of essential boundary conditions with MMs is more involved than with a FEM due to the noninterpolatory character of the meshfree shape functions. There are several different approaches to the implementation of essential boundary conditions with meshfree approximations; see [13, 16, 27]. Throughout this paper we use Lagrangian multipliers to enforce essential boundary conditions; see [13, 27].

In the following let $a(\cdot,\cdot)$ be the continuous and elliptic bilinear form induced by L on $H^1(\Omega)$. We discretize the PDE using Galerkin's method. Then, we have to compute the stiffness matrix

$$A = (a_{ij})\,, \text{ with } a_{ij} = a(\varphi_j\psi_j^l, \varphi_i\psi_i^k) \in \mathbb{R}^{\dim(V_i^{p_i})\times\dim(V_j^{p_j})}$$

and the right-hand side vector

$$\hat{f} = (f_i) \text{ with } f_i = \langle f, \varphi_i\psi_i^k\rangle_{L^2} = \int_\Omega f\varphi_i\psi_i^k \in \mathbb{R}^{\dim(V_i^{p_i})}\,.$$

If we restrict ourselves for reasons of simplicity to the case $L = -\Delta$ we have to compute the integrals $\int_\Omega \varphi_i\psi_i^k f$ for the right-hand side and the integrals $\int_\Omega \nabla(\varphi_i\psi_i^k)\nabla(\varphi_j\psi_j^l)$ for the stiffness matrix. Recall that φ_i is defined by (2.2), i.e.,

$$\varphi_i(x) = \frac{W_i(x)}{\sum W_k(x)}.$$

Now we carry out the differentiation in $\int_\Omega \nabla(\varphi_i\psi_i^k)\nabla(\varphi_j\psi_j^l)$. With the notation $\mathcal{S} := \sum W_k$, $\mathcal{T} := \sum \nabla W_k$, and $\mathcal{G}_i := \nabla W_i\mathcal{S} - W_i\mathcal{T}$ we end up with the integrals

$$\begin{aligned} a(\varphi_j\psi_j^l, \varphi_i\psi_i^k) = \int_\Omega \mathcal{S}^{-4}\mathcal{G}_i\psi_i^k\mathcal{G}_j\psi_j^l \quad &+ \int_\Omega \mathcal{S}^{-2}W_i\nabla\psi_i^kW_j\nabla\psi_j^l \\ + \int_\Omega \mathcal{S}^{-3}\mathcal{G}_i\psi_i^kW_j\nabla\psi_j^l \quad &+ \int_\Omega \mathcal{S}^{-3}W_i\nabla\psi_i^k\mathcal{G}_j\psi_j^l \end{aligned} \tag{4.2}$$

for the stiffness matrix and the integrals

$$\langle f, \varphi_i\psi_i^k\rangle_{L^2} = \int_\Omega \mathcal{S}^{-1}W_i\psi_i^kf \tag{4.3}$$

for the right-hand side. Due to the fact that we use piecewise polynomial weights W_i for the Shepard construction (2.2) and that the support patches ω_i overlap each other, the functions $\mathcal{T}$ and $\mathcal{G}_i$ may have quite a number of jumps of significant size. Therefore, the integrals (4.2) and (4.3) should not be computed by a simple quadrature scheme which does not respect these discontinuities and the algebraic structure of the shape functions. Instead, we need to decompose the integration domain in such a way that the piecewise character of the integrands is resolved.

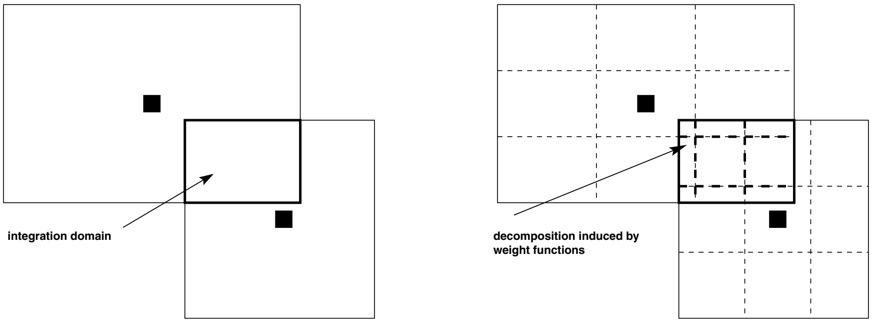


FIG. 5.1. *Integration domain $\Omega_{ij} = \omega_i \cap \omega_j$ (left). The decomposition $E_{\omega_{ij}}$ of the integration domain ω_{ij} via the subdivision induced by the weight functions W_i and W_j (right). Here, the weights are tensor products of quadratic B-splines.*

5. Decomposition sparse grid quadrature scheme.

Let us assume that the PU is given by an h-mesh construction as we have in the GFEM. Then, we know how to resolve the piecewise character of the integrands: we subdivide the integration domains with the help of the geometric elements of the h-mesh. However, with our general PUM we do not have a mesh or geometric elements. However, we have support patches ω_i and weight functions W_i which define the PU functions φ_i by (2.2). From this information only, we have to find an appropriate subdivision of the support patches ω_i and subsequently the integration domains. Furthermore, we have to cope with rational integrands on the cells of such a subdivision in our general PUM. The foundation for the proposed quadrature scheme is a decomposition approach which was first presented in [27]. Here, we give a short review of the construction principles for the decomposition $D_{\omega_{ij}} := \{D^n_{\omega_{ij}}\}$ of the integration domains $\omega_{ij} := \omega_i \cap \omega_j \cap \Omega$.

The integration domains ω_{ij} may be decomposed into disjoint cells $D^n_{\omega_{ij}}$ by exploiting the tensor product structure of the cover patches ω_i and the weight functions W_i used during the construction (2.2) of the PU $\{\varphi_i\}$. This decomposition of an integration domain ω_{ij} can efficiently be computed by splitting ω_{ij} via its caps $\omega_{ij} \cap \omega_k$ with the neighboring cover patches $\omega_k \in N_{ij} := N_i \cap N_j$ using a second tree data structure.

Consider the integration domain $\omega_{ij} = \omega_i \cap \omega_j \subset \Omega$. The intersection ω_{ij} of two cover patches ω_i, ω_j which are tensor products of intervals is also a tensor product of intervals; see Figure 5.1 (left). Moreover, the employed weight functions W_k are tensor products of normed B-splines of order l; i.e., they are piecewise polynomials of degree l. Therefore, the weight function W_k induces a subdivision of the respective cover patch ω_k into $(l+1)^d$ subpatches $\{\omega^q_k\}$ on which $W_k\big|_{\omega^q_k}$ is polynomial. Furthermore, these subpatches $\{\omega^q_k\}$ are also tensor products of intervals. With the help of these subpatches $\{\omega^q_i\}$, $\{\omega^q_j\}$ we can define a first decomposition $E_{\omega_{ij}} = \{E^n_{\omega_{ij}}\}$ of ω_{ij}; see Figure 5.1 (right). On the cells $E^n_{\omega_{ij}}$ of this decomposition we have that $W_i\big|_{E^n_{\omega_{ij}}}$ and $W_j\big|_{E^n_{\omega_{ij}}}$ are polynomials of degree l, but all other $W_k\big|_{E^n_{\omega_{ij}}}$ may still be piecewise polynomial only. Therefore, we further refine the decomposition $E_{\omega_{ij}}$ by subdividing the cells $E^n_{\omega_{ij}}$ with the help of the $\{\omega^q_k\}$ subpatches for all $\omega_k \in N_{ij}$; see Figure 5.2. The resulting decomposition $D_{\omega_{ij}} = \{D^n_{\omega_{ij}}\}$ consists of d-rectangular cells $D^n_{\omega_{ij}}$ on which all weight functions $W_k|_{D^n_{\omega_{ij}}}$ are polynomials of degree l. The number of cells $\mathrm{card}(D_{\omega_{ij}})$ of the decomposition $D_{\omega_{ij}} = \{D^n_{\omega_{ij}}\}$ depends on the polynomial degree l

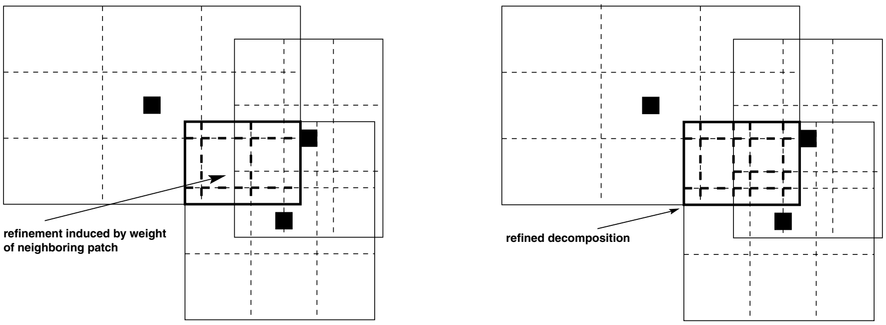


FIG. 5.2. *Refinement of the decomposition $E_{\omega_{ij}}$ of the integration domain ω_{ij} via the subdivision induced by the weight function W_k (tensor product of quadratic B-splines) of one neighboring point x_k (left). The resulting decomposition $D_{\omega_{ij}}$ after the refinement step for the neighboring weight function W_k (right).*

of the weight functions W_k used during the Shepard construction (2.2) for the PU, the number of neighbors $\text{card}(N_i)$, and their geometric location.

Since all weights W_k are polynomial on the cells $D^n_{\omega_{ij}}$, the functions $\mathcal{T}$ and $\mathcal{G}_i$ are nonsingular rational functions on $D^n_{\omega_{ij}}$. Hence, any standard quadrature rule for smooth functions is applicable for the numerical integration of the weak form on the cells $D^n_{\omega_{ij}}$ (if we assume that the local basis functions ψ^k_i and ψ^l_j are smooth on ω_{ij}). Independent of the local quadrature rule used on $D^n_{\omega_{ij}}$ we can utilize the product structure of the shape functions $\varphi_i\psi^k_i$ to reduce the computational costs of an evaluation of the weak form at a quadrature point. Here, we simultaneously evaluate the complete block $a_{ij} = a(\varphi_j\psi^l_j, \varphi_i\psi^k_i) \in \mathbb{R}^{\dim(V^{p_i}_i)\times\dim(V^{p_j}_j)}$ of the stiffness matrix rather than evaluating every single $(a_{ij})_{kl} = a(\varphi_j\psi^l_j, \varphi_i\psi^k_i) \in \mathbb{R}$ for fixed k and l. Besides the reduction in the number of evaluations of φ_i and φ_j, this also allows for a hierarchical evaluation of the local basis functions ψ^k_i and ψ^l_j (which is available for the chosen Legendre polynomials) which reduces the computational costs of an evaluation of the weak form significantly (esp. for higher order approximations).

For the selection of a quadrature rule on the cells $D^n_{\omega_{ij}}$ we can now assume the smoothness of the integrands. However, the quadrature rule still has to be applicable to general situations (general covers, weights and local basis functions ψ^k_i, etc.). Hence, we have to find a fast converging, cheap quadrature rule on $D^n_{\omega_{ij}}$ which allows for a reliable dynamic stopping criterion for a wide range of integrands.

So-called sparse grid quadrature [12] rules are multidimensional interpolatory rules with a substantially smaller number of integration nodes compared with a tensor product rule. They are defined as special products of one-dimensional interpolatory quadrature rules. Although the number of evaluations of the integrand is significantly less for a sparse grid quadrature rule, the order of the achieved error is comparable to that of a full tensor product rule. Here, we state only the fundamental construction principles and error bounds (see [12]) and the references cited therein for further details.

Consider a sequence of nested one-dimensional quadrature rules for univariate functions $\{Q^1_l \,|\, Q^1_l f := \sum_{i=1}^{n^1_l} w_{li} f(x_{li}),\ n^1_l = O(2^l),\ f : \mathbb{R} \to \mathbb{R}\}$ with weights w_{li}, nodes x_{li}, and error bound $|Q^1_l f - \int f| = O(2^{-lr})$, where f is assumed to be r-times continuously differentiable. This bound holds, for example, for the Clenshaw–Curtis

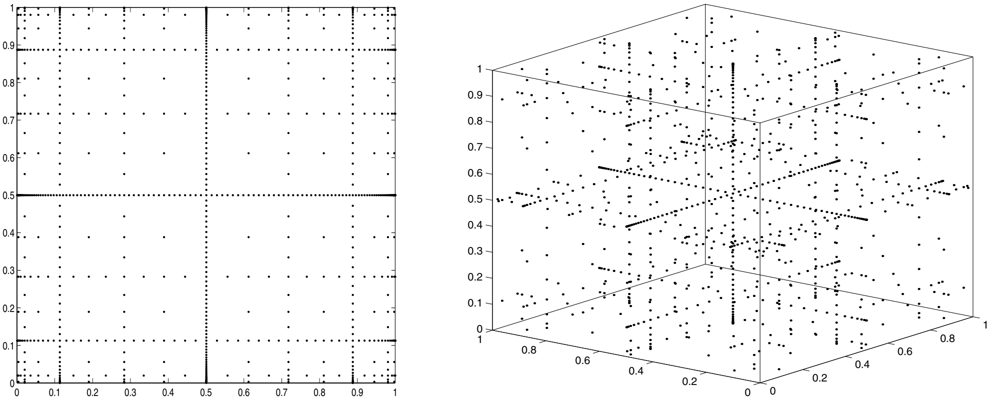


FIG. 5.3. *Quadrature nodes of sparse grid Gauß–Patterson rules, level* $l = 6$ *with* 769 *nodes in two dimensions (left) and level* $l = 5$ *with* 1023 *nodes in three dimensions (right).*

and Gauß–Patterson rules. With the help of the difference quadrature rules Δ_k^1,

$$\Delta_k^1 f := (Q_k^1 - Q_{k-1}^1) f \quad \text{with} \quad Q_0^1 f := 0,$$

we can define the sparse grid quadrature rule Q_l^d on level l in d-dimensions as

$$Q_l^d f := \sum_{\sum_{i=1}^d k_i \leq l+d-1} (\Delta_{k_1}^1 \otimes \cdots \otimes \Delta_{k_d}^1) f$$

with $f : \mathbb{R}^d \to \mathbb{R}$, $l \in \mathbb{N}$, and $k \in \mathbb{N}^d$. Due to the restriction $\sum_{i=1}^d k_i \leq l + d - 1$ in the summation, the number n_l^d of quadrature points x_i^d of the resulting sparse grid quadrature rule Q_l^d is

$$n_l^d = O(2^l l^{d-1})$$

only. Hence, the number of function evaluations for a sparse grid quadrature rule is dramatically less (see Figure 5.3) than for a full tensor product rule where the integrand has to be evaluated at $O(2^{ld})$ quadrature points. This reduction of the computational costs, however, does not compromise the approximation quality significantly for smooth functions. When f is assumed to be r-times continuously differentiable the estimate

$$|Q_l^d f - \textstyle\int f| = O(2^{-lr} l^{(d-1)(r+1)})$$

holds.

In summary, sparse grid quadrature rules are not only cheaper to evaluate (esp. in higher dimensions) compared with tensor product rules, but rather their overall efficiency with respect to accuracy is significantly better. In [12] the fast convergence of sparse grid quadrature rules based on Gauß–Patterson rules (see Figure 5.3) is shown for a wide variety of function classes. In fact, sparse grid quadrature rules based on Gauß–Patterson rules converge exponentially for smooth integrands. Since the integrands we are interested in are smooth on the cells $D_{\omega_{ij}}^n$ of the constructed decomposition $D_{\omega_{ij}}$ we use Gauß–Patterson sparse grid rules for the numerical integration of the stiffness matrix entries.

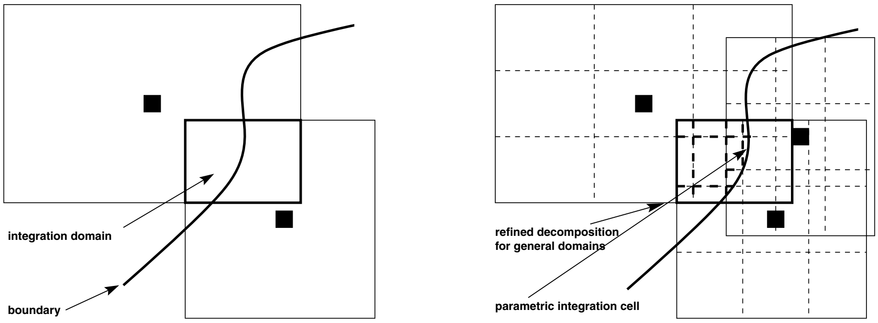


FIG. 5.4. *Integration domain* $\Omega_{ij} = \omega_i \cap \omega_j \cap \Omega$ *for general domains* Ω *(left). The decomposition* $D_{\omega_{ij}}$ *of a general integration domain* ω_{ij} *via the subdivision induced by the weight functions* W_i, W_j, *and a neighboring patch (right); compare Figures* 5.1 *and* 5.2. *Here, the weights are tensor products of quadratic B-splines.*

To ensure a reliable accuracy of our quadrature scheme, we use a simple three level dynamic stopping criterion [24]. The quadrature on a cell $D^n_{\omega_{ij}}$ is stopped if

$$|Q^d_{l-1}f - Q^d_{l-2}f| \le c_1\epsilon_a + c_2\epsilon_r|Q^d_{l-1}f| \quad \text{and} \quad |Q^d_l f - Q^d_{l-1}f| \le \epsilon_a + c_3\epsilon_r|Q^d_l f|$$

hold for the integrand $(a_{ij})_{kl}$ of minimal order $k = l = 0$ as well as for the integrand of maximal order $k = \dim(V_i^{p_i})-1$, $l = \dim(V_j^{p_j})-1$. Here, c_1, c_2, and c_3 are nonnegative constants, and ϵ_a and ϵ_r are user supplied absolute and relative tolerances. These tolerances which determine the accuracy of the integration, however, have to be chosen with respect to the approximation space. Here, the diameters $\operatorname{diam}(\omega_i)$, $\operatorname{diam}(\omega_j)$ of the cover patches ω_i, ω_j, the number of integration cells $\operatorname{card}(D_{\omega_{ij}})$, their respective diameters $\operatorname{diam}(D^n_{\omega_{ij}})$, and the local approximation orders p_i, p_j have to be considered. An automatic selection of the tolerances ϵ_a and ϵ_r which minimizes the computational work but at the same time does not compromise the accuracy of the discretization [29, Chapter 4] is the subject of future research.

Note that the decomposition approach given above is not restricted to domains Ω which are unions of d-rectangles but rather applicable to general domains. For integration domains $\omega_i \cap \omega_j \cap \Omega \neq \omega_i \cap \omega_j$ we apply the construction to the fictitious integration domain $\tilde{\omega}_{ij} := \omega_i \cap \omega_j$. From this decomposition $D_{\tilde{\omega}_{ij}} = \{D^n_{\tilde{\omega}_{ij}}\}$ we then select the subset $\hat{D}_{\omega_{ij}} := \{\hat{D}^n_{\omega_{ij}} \in D_{\tilde{\omega}_{ij}} \,|\, \hat{D}^n_{\omega_{ij}} \cap \Omega \neq \emptyset\}$ of all cells which overlap the actual integration domain $\omega_{ij} = \tilde{\omega}_{ij} \cap \Omega$. This intermediate cover $\hat{D}_{\omega_{ij}}$ consists of d-rectangular cells $\hat{D}^n_{\omega_{ij}} \cap \Omega \neq \emptyset$ which cover the integration domain ω_{ij}; see Figure 5.4. Now the remaining task is to resolve (with the necessary accuracy) the boundary $\partial\Omega$ of the domain which runs through some of the cells $\hat{D}^n_{\omega_{ij}}$. We assume that a representation for the boundary $\partial\Omega$ is given as part of the computational domain Ω. That is, we assume the domain Ω and its boundary $\partial\Omega$ are given as a collection of mappings $\mathcal{R}^m_\Omega : [-1,1]^d \to \overline{\Omega} \subset \mathbb{R}^d$ from a reference cell into the physical space $\overline{\Omega} \subset \mathbb{R}^d$ with $\bigcup \mathcal{R}^m([-1,1]^d) = \overline{\Omega}$. These mappings $\mathcal{R}^m_\Omega$ may be coming from a CAD system; i.e., the mappings themselves are only an approximation to the true domain Ω, or may be coming from a given analytical representation of the domain Ω.

With the help of these mappings $\mathcal{R}^m_\Omega$ we can compute the parametric integration cell $D^n_{\omega_{ij}} := \hat{D}^n_{\omega_{ij}} \cap \Omega$ which lies within the integration domain ω_{ij}; see Figure 5.4. The final decomposition $D_{\omega_{ij}} = \{D^n_{\omega_{ij}} \,|\, D^n_{\omega_{ij}} := \hat{D}^n_{\omega_{ij}} \cap \Omega\}$ for general domains Ω therefore consists of d-rectangular cells $D^n_{\omega_{ij}} = \hat{D}^n_{\omega_{ij}}$ if $\hat{D}^n_{\omega_{ij}} \subset \Omega$ and parametric cells $D^n_{\omega_{ij}}$ if the corresponding d-rectangular cell $\hat{D}^n_{\omega_{ij}}$ overlaps the boundary $\partial\Omega$.

Note that the mappings $\mathcal{R}^m_\Omega$ do *not* influence the shape functions of our PUN method. While in a FEM the supports of the shape function have to align with the boundary $\partial\Omega$ of the domain, in our PUM the supports of the shape functions have to cover only the domain Ω and its boundary $\partial\Omega$. The mappings of the domain representation $\mathcal{R}^m_\Omega$ are merely used to implement integration domains ω_{ij} with complicated geometry.

Overall, the numerical quadrature of the stiffness matrix entries is completed in three steps:

1. First we compute the decomposition $D_{\tilde{\omega}_{ij}}$ for the domain $\tilde{\omega}_{ij} = \omega_i \cap \omega_j$; see Figures 5.1 and 5.2. With the help of the domain representation mappings $\mathcal{R}^m_\Omega$ we then select the integration cells $\hat{D}^n_{\omega_{ij}} = D^n_{\tilde{\omega}_{ij}} \cap \Omega \neq \emptyset$ which overlap the computational domain Ω. Furthermore, we use the mappings $\mathcal{R}^m_\Omega$ to construct the final decomposition $D_{\omega_{ij}} = \{D^n_{\omega_{ij}} \mid D^n_{\omega_{ij}} := \hat{D}^n_{\omega_{ij}} \cap \Omega\}$; see Figure 5.4.
2. For each of these integration cells $D^n_{\omega_{ij}}$ we then compute the Jacobian $J_{T^n_{\omega_{ij}}}$ of the mapping $T^n_{\omega_{ij}} : [-1,1]^d \to D^n_{\omega_{ij}}$ from the reference integration domain $[-1,1]^d$ onto the integration cell $D^n_{\omega_{ij}}$ in the configuration space.
3. Finally, we evaluate the entries of the stiffness matrix and right-hand side vector by computing the integrals (4.2) and (4.3) on the integration cells $D^n_{\omega_{ij}}$. That is, we compute them on the reference integration domain $[-1,1]^d$ using the transformations $T^n_{\omega_{ij}}$ and the respective Jacobians $J_{T^n_{\omega_{ij}}}$:

$$\int_{D^n_{\omega_{ij}}} \mathcal{F} = \int_{[-1,1]^d} \mathcal{F} \circ T^n_{\omega_{ij}} |J_{T^n_{\omega_{ij}}}|.$$

   The Jacobian $J_{T^n_{\omega_{ij}}}$ for a simple d-rectangular integration cell is of course constant, and this transformation does not increase the costs of the numerical integration. However, for a parametric cell the transformation $T^n_{\omega_{ij}}$ involves the mappings $\mathcal{R}^m_\Omega$ of the domain representation. Therefore, the Jacobian may well be space-dependent and has to be evaluated at every integration node of the quadrature rule.[7]

   Again, the error during the numerical quadrature has to be controlled by the selection of ϵ_a and ϵ_r to ensure that the order of approximation is not compromised by the integration error.

The overall computational costs of the proposed quadrature scheme depends on the number of cells $\text{card}(D_{\omega_{ij}})$ of the decomposition, i.e., on the order l of the weight functions, the geometric location of the neighbors $\omega_j \in N_i$, their number $\text{card}(N_i)$, and the local quadrature rule used on the cells $D^n_{\omega_{ij}}$. Due to the use of the sparse grid rules on the cells, the computational costs are significantly reduced compared with tensor product rules.

6. Hierarchical regular cover construction.

A further reduction of the computational effort necessary during the assembly of the stiffness matrix can be achieved

[7] Note that in general the transformations $T^n_{\omega_{ij}}$ may lead to (locally) nonsmooth integrands where an isotropic sparse grid quadrature scheme may not be well-suited on all integration cells $D^n_{\omega_{ij}}$. Here, a further decomposition of the integration cells $D^n_{\omega_{ij}}$ where the integrands are nonsmooth or even adaptive quadrature rules (sparse grid or other) should be employed on such integration cells $D^n_{\omega_{ij}}$.

Table 6.1

The number $\sum \operatorname{card}(N_i)$ *of neighbors for covers generated by Algorithm* 1 *with* $\alpha = 1.5$ *using rectangular patches* ($A_{1,R}$)*, and using square patches* ($A_{1,S}$)*. The number of cover patches* $\operatorname{card}(P)$ *after the cover construction and the number of neighbors for Algorithm* 2 *with* $k = \infty$, $x^i_L = l^i_L + \frac{1}{2}(u^i_L - l^i_L)$ *for generated points* x_L *and* $h^i_L = \frac{5}{4}\max\{u^i_L - x^i_L, x^i_L - l^i_L\}$ (A_2*; see also Figure* 3.2)*, and for Algorithm* 3 *with* $k = \infty$ *and* $\alpha_l = 2$ (A_3*; see also Figure* 6.1)*. The initial point set* P *for all algorithms was Halton*$^{N-1}_0(2, 3)$.

N	$A_{1,R}$	$A_{1,S}$	card(P)	A_2	A_3
1024	18840	21530	1729	20023	21751
4096	79102	91902	6364	73908	78588
16384	325730	377388	27673	326919	360053
65536	1267300	1431210	101314	1245108	1259134

Table 6.2

The number $\sum \operatorname{card}(N_i)$ *of neighbors for covers generated by Algorithm* 1 *with* $\alpha = 1.5$ *using rectangular patches* ($A_{1,R}$)*, and using square patches* ($A_{1,S}$)*. The number of cover patches* $\operatorname{card}(P)$ *after the cover construction and the number of neighbors for Algorithm* 2 *with* $k = \infty$, $x^i_L = l^i_L + \frac{1}{2}(u^i_L - l^i_L)$ *for generated points* x_L *and* $h^i_L = \frac{5}{4}\max\{u^i_L - x^i_L, x^i_L - l^i_L\}$ (A_2*; see also Figure* 3.4)*, and for Algorithm* 3 *with* $k = \infty$ *and* $\alpha_l = 2$ (A_3*; see also Figure* 6.3)*. The initial point set* P *for all algorithms was a graded Halton*$^{N-1}_0(2, 3)$.

N	$A_{1,R}$	$A_{1,S}$	card(P)	A_2	A_3
1024	33878	39114	1897	23193	26203
4096	127950	149460	7501	92235	104985
16384	507010	591794	30040	369754	416292

only by reducing the number of cells of the decomposition[8] $D_{\omega_{ij}}$. This can be attained by the alignment of the cover patches ω_k and their subdivisions $\{\omega^q_k\}$.

Taking into account that we limit ourselves to the use of tensor product B-splines as weight functions for Shepard's construction (2.2) we can align the cover patches to simplify the algebraic structure of the resulting partition of unity functions φ_i. Here, we eliminate some of the flexibility in step 3 of Algorithm 2 for the choices of x_L and α. This, however, does not lead to a significantly larger number of neighbors. Hence, the number of nonzeros of the stiffness matrix stays (almost) constant; see Tables 6.1 and 6.2. However, the number of integration cells $\operatorname{card}(D_{\omega_{ij}})$ is substantially reduced by this modification; see Tables 6.4 and 6.5.

Recall that we split the integration domain ω_{ij} into several cells by its caps $\omega_{ij} \cap \omega^q_k$ with the cells ω^q_k of the subdivision induced by the weight W_k on $\omega_k \in N_{ij}$ during the construction of the decomposition $D_{\omega_{ij}}$. Hence, we align these caps $\omega_{ij} \cap \omega^q_k$, which subsequently induce at least one integration cell $D^n_{\omega_{ij}}$, if we align the neighboring cover patches ω_k with respect to their subdivisions $\{\omega^q_k\}$. Therefore, many of the $\omega_{ij} \cap \omega^q_k$ will lead to the same integration cell $D^n_{\omega_{ij}}$, and the overall number of integration cells $\operatorname{card}(D_{\omega_{ij}})$ will be reduced significantly; see Tables 6.4 and 6.5. This alignment of the cover patches ω_k and their subdivisions $\{\omega^q_k\}$ is achieved by the following algorithm.

[8] The decomposition itself, however, is minimal in the sense that it has a minimal number $\operatorname{card}(D_{\omega_{ij}})$ of integration cells necessary to resolve the piecewise character of the PU functions. In our construction (2.2) of the PU we have to allow for higher orders l of the B-spline weights to be able to construct global solutions u_{PU} with higher order regularity, i.e., $u_{\mathrm{PU}} \in \mathcal{C}^{l-1}$. Therefore, the remaining influences on the computational effort involved with the numerical integration of the stiffness matrix entries are the geometric neighboring relations of our cover patches ω_i.

Algorithm 3 (hierarchical regular cover construction).

1. Given: the domain $\Omega \subset \mathbb{R}^d$, a bounding box $R_\Omega = \bigotimes_{i=1}^d [l_\Omega^i, u_\Omega^i] \supset \overline{\Omega}$, the initial point set $P = \{x_j \in \mathbb{R}^d \,|\, x_j \in \overline{\Omega}\}$, and a parameter $k \in \mathbb{N}$.
2. Build a d-binary tree (quadtree, octree) over R_Ω such that per leaf L at most one $x_i \in P$ lies within the associated cell $C_L := \bigotimes_{i=1}^d [l_L^i, u_L^i]$, and the difference of the levels of two adjacent cells is at most k.
3. For all cells $C_L = \bigotimes_{i=1}^d [l_L^i, u_L^i]$ with $C_L \cap \Omega \neq \emptyset$:
   (a) Set $x_L^i = l_L^i + \frac{1}{2}(u_L^i - l_L^i)$.
   (b) If there is an $x_j \in P$ with $x_j \in C_L$, set $P = P \setminus \{x_j\}$.
   (c) Set $P = P \cup \{x_L\}$.
   (d) Set $\omega_L = R_L = \bigotimes_{i=1}^d [x_L^i - h_L^i, x_L^i + h_L^i] \supset C_L$, where $h_L^i = \frac{\alpha_l}{2}(u_L^i - l_L^i)$.

Here, the parameter α_l in the computation of the support size in step 3(d) is dependent only on the weight function used in (2.2), i.e., the order l of the B-spline. By construction the one-dimensional distances from a point $x_L \in P$ to its direct neighboring point $x_j \in P$, i.e., the point x_j corresponding to the sibling tree cell $C_j = \bigotimes_{i=1}^d [l_j^i, u_j^i]$, are $|x_L^i - x_j^i| = u_L^i - l_L^i = u_j^i - l_j^i$, where $C_L = \bigotimes_{i=1}^d [l_L^i, u_L^i]$ is the cell associated with x_L. Hence, if we choose α_l in such a way that condition (6.1) is fulfilled, we not only align the patch ω_L with its direct neighboring patch ω_j but rather also their corresponding subdivisions $\{\omega_L^q\}$ and $\{\omega_j^q\}$ induced by the weight functions W_L and W_j; see Figure 6.1. Moreover, this alignment of the patches does not increase the number of neighbors $\operatorname{card}(N_L)$. With the notation $h_l^i := \frac{\alpha_l}{l+1}(u_L^i - l_L^i)$ for the B-spline interval size, the condition reads

$$x_L^i + \frac{l+1}{2} h_l^i = x_j^i - \left(\frac{l+1}{2} - m\right) h_l^i = x_L^i + (u_L^i - l_L^i) - \left(\frac{l+1}{2} - m\right) h_l^i \tag{6.1}$$

for the ith coordinate with $i = 1, \dots, d$. Here, the parameter $m \in \mathbb{N}$ indicates the amount of overlap $\omega_L \cap \omega_j \sim \bigotimes_{i=1}^d m h_l^i$ for the neighbor $\omega_j \in N_L$. Any integer m with $1 \leq m \leq \frac{l+1}{2}$ leads to minimal neighborhoods N_L and minimal decompositions $D_{\omega_{Lj}}$; i.e., the number $\sum_L \operatorname{card}(N_L)$ of nonzero entries of the stiffness matrix and the number $\sum_{L,j} \operatorname{card}(D_{\omega_{Lj}})$ of integration cells are (almost) constant. Therefore, it is advisable to choose the largest such integer to control the gradients of the PU function φ_i. Solving (6.1) for α_l we have

$$\alpha_l = \frac{l+1}{l+1-m}.$$

With the choice of $l = 2n - 1$ and maximal $m = n$, this yields $\alpha_l = 2$; in general, we have $1 < \alpha_l \leq 2$. Due to this construction many of the points $x_i \in P$ are covered only by the corresponding ω_i. Therefore, we have $\varphi_i(x_j) = \delta_{ij}$ for many PU functions φ_i and points $x_j \in P$; see Figure 6.2. In fact, $\varphi_i(x) = 1$ holds not only for the point $x = x_i$ if we have $\alpha_l < 2$ but rather on a subpatch $\tilde{\omega}_i \subset \omega_i$ with $x_i \in \tilde{\omega}_i \sim \bigotimes_{i=1}^d h_l^i$, i.e., $\varphi_i|_{\tilde{\omega}_i} \equiv 1$; see Figure 6.2.

When we compare the covers C_Ω (Figures 3.2 and 6.1, Figures 3.4 and 6.3) and functions φ_i (Figures 3.3 and 6.2) generated by Algorithms 2 and 3, we clearly see the effect of the alignment of the cover patches.

Note that the change of the point set P in step 3(c) in Algorithm 3 is admissible due to the noninterpolatory character of the PUM shape functions $\varphi_i \psi_i^k$. We can interpret this change in the point set P as a change of the weight functions W_k used during the Shepard construction (2.2). So far the weight functions W_k and the cover

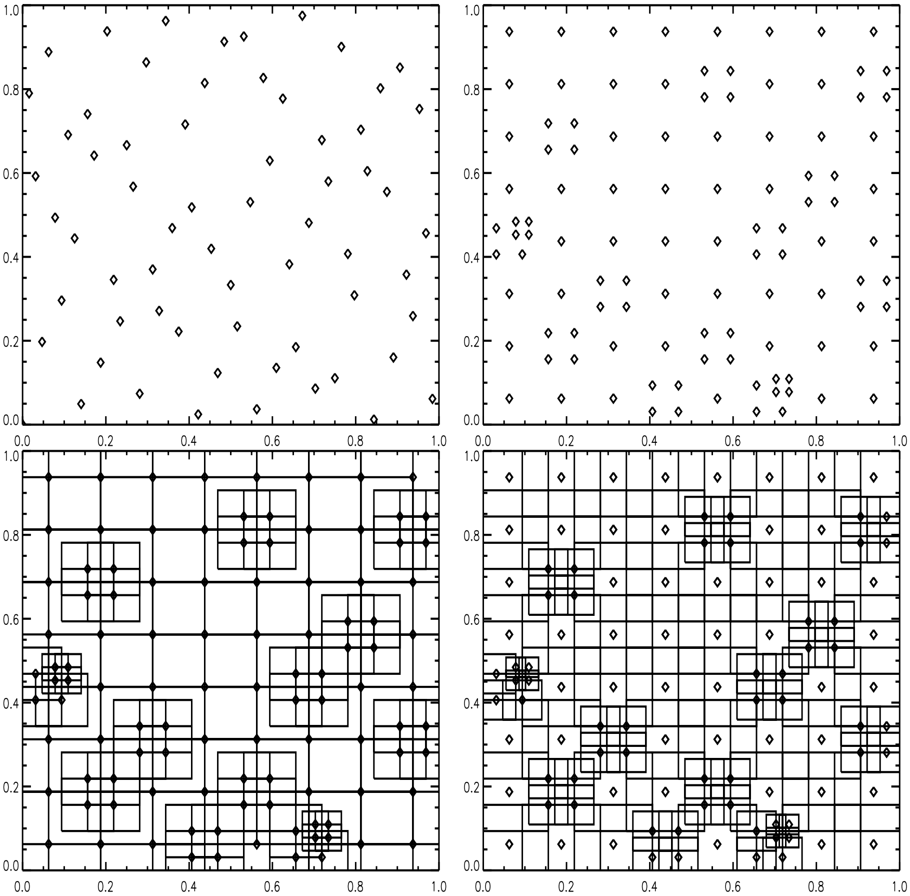


FIG. 6.1. $P = \mathit{Halton}_0^{63}(2,3)$ *point set in* $R_\Omega = \Omega = [0,1]^2$ *(upper left),* P *with* $\mathrm{card}(P) = 106$ *(upper right) after Algorithm* 3 *with* $k = \infty$*, and the generated cover* C_Ω *with* $\alpha_l = 2$ *(lower left) and* $\alpha_l = 1.5$ *(lower right).*

patches ω_k were assumed to be centered in the given point x_k (compare section 2), but this is of course not a necessary condition for the PUM to work. Therefore, we may view the construction given above as a more general approach toward assigning weight functions W_k to a given point $x_k \in P$. The weight functions W_k and cover patches ω_k are now centered in $l_L + \frac{1}{2}(u_L - l_L)$ rather than in the given point x_k. Note that the constructed point set P of newly introduced and shifted points x_k is only part of the implementation of the function space. The initial point set P of step 1 is still the set of all relevant points for the resolution of the solution and the geometry of the domain. Therefore, a copy $\tilde{P}$ of the initial point set P is stored separately, and the points $x_l \in \tilde{P}$ are used in time-dependent settings to generate covers for future time steps [13]. Hence by the introduction of general weight functions W_k as part of the cover construction, we can also write step 3 of Algorithm 3 equivalently as

3'. For all cells $C_L = \bigotimes_{i=1}^d [l_L^i, u_L^i]$ with $C_L \cap \Omega \neq \emptyset$:
- (a) Set $x_L^i = l_L^i + \frac{1}{2}(u_L^i - l_L^i)$.
- (b) If there is *no* $x_j \in P$ with $x_j \in C_L$, set $P = P \cup \{x_L\}$.
- (c) Set $\omega_L = R_L = \bigotimes_{i=1}^d [x_L^i - h_L^i, x_L^i + h_L^i] \supset C_L$, where $h_L^i = \frac{\alpha_l}{2}(u_L^i - l_L^i)$.
- (d) Set the associated weight function $W_L(x) := \Pi_{i=1}^d \mathcal{W}(\frac{x - x_L^i + h_L^i}{2h_L^i})$.

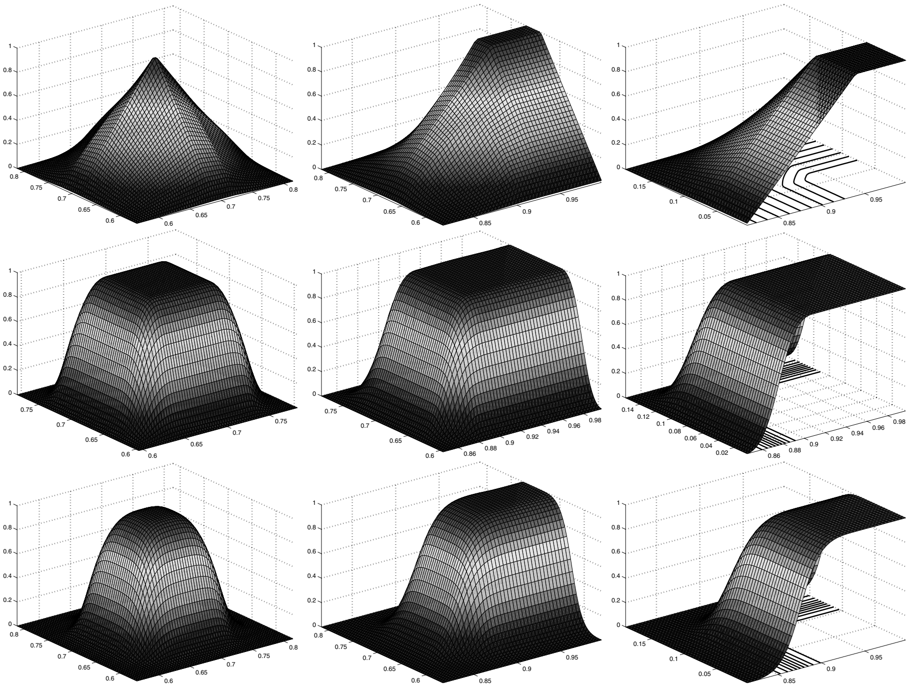


FIG. 6.2. *The PU function φ_i on $\Omega \cap \omega_i$ generated by Algorithm* 3 *with the input data (upper row $l = 1$, $\alpha_l = 2$, center row $l = 2$, $\alpha_l = 1.5$, lower row $l = 3$, $\alpha_l = 2$) from* 6.1 *for an interior point (left), a boundary point (center), and a corner point (right).*

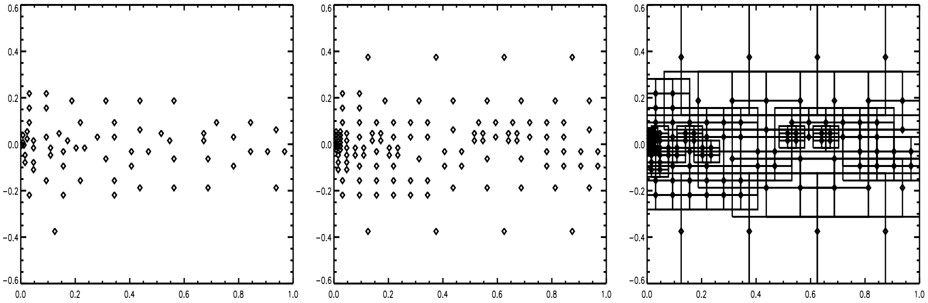


FIG. 6.3. *P is a Halton$_0^{63}(2,3)$ point set in $R_\Omega = \Omega = [0,1] \times [-0.5, 0.5]$ graded by $(x, y) \mapsto (x^2, \pm y^2)$ (left), P with* $\text{card}(P) = 121$ *(center) after Algorithm* 3 *with $k = \infty$, and the generated cover C_Ω with $\alpha_l = 2$ (right).*

This leaves the given points at their original location. Note also that the cover patches $\omega_L = R_L$ constructed with Algorithm 3 and the bounding box R_Ω always have the same aspect ratio; see Figure 6.1. If we apply the algorithm given above to $\Omega = R_\Omega = [0,1]^d$ with $k = \infty$ and $\alpha_l = 2$ to a uniformly distributed set of points P, we construct a uniform grid[9] (or at least an r-irregular grid with very small r depending only on the quality of the initial point set P; see Figure 6.1). Here, also the cells $D^n_{\omega_{ij}}$ of the decomposition $D_{\omega_{ij}}$ are (geometrically) identical to a bilinear finite element.

[9] The covers from Algorithm 3 with $k = 0$ will correspond to a uniform grid regardless of the initial point set P when we have the order $l = 2n - 1$ and maximal $m = n$.

Table 6.3

The average number s_{C_Ω} (6.3) of nonzero blocks a_{ij} per row of the stiffness matrix for the different cover construction algorithms in two (left) and three dimensions (right). Algorithm 2 with $k = \infty$, $x^i_L = l^i_L + \frac{1}{2}(u^i_L - l^i_L)$ for generated points x_L and $h^i_L = \frac{5}{4}\max\{u^i_L - x^i_L, x^i_L - l^i_L\}$ (A_2), and Algorithm 3 with $k = \infty$, $\alpha_l = 2$ ($A_{3,2}$) and $\alpha_l = 1.5$ ($A_{3,1.5}$) . The initial point set P was Halton$^{N-1}_0(2,3)$ in $[0,1]^2$ in two dimensions (left) and Halton$^{N-1}_0(2,3,5)$ in $[0,1]^3$ in three dimensions (right).

$d = 2$					$d = 3$				
N	card(P)	A_2	$A_{3,2}$	$A_{3,1.5}$	N	card(P)	A_2	$A_{3,2}$	$A_{3,1.5}$
1024	1729	11.58	12.58	8.51	1024	3543	26.81	30.90	21.23
4096	6364	11.61	12.35	8.48	8192	26699	29.16	34.44	22.41
16384	27673	11.81	13.01	8.65	65536	199417	31.53	34.80	23.30
65536	101314	12.29	12.43	8.56	524288	1694568	32.31	34.94	24.04

Table 6.4

The average number a_{C_Ω} (6.2) of integration cells per nonzero block a_{ij} of the stiffness matrix for the different cover construction algorithms. Algorithm 2 with $k = \infty$, $x^i_L = l^i_L + \frac{1}{2}(u^i_L - l^i_L)$ for generated points x_L and $h^i_L = \frac{5}{4}\max\{u^i_L - x^i_L, x^i_L - l^i_L\}$ (A^l_2), and Algorithm 3 with $k = \infty$ (A^l_{3,α_l}) using a linear, a quadratic and a cubic B-spline during the Shepard construction (2.2). The initial point set P was Halton$^{N-1}_0(2,3)$ in $[0,1]^2$.

N	card(P)	A^1_2	$A^1_{3,2}$	$A^1_{3,1.3}$	A^2_2	$A^2_{3,1.5}$	A^3_2	$A^3_{3,2}$
1024	1729	31.54	6.39	7.03	48.82	8.16	76.51	10.77
4096	6364	32.22	5.29	6.60	50.34	7.46	78.51	9.93
16384	27673	32.18	6.28	6.99	49.51	8.27	77.20	10.76
65536	101314	34.65	5.16	6.62	53.75	7.40	82.47	9.85

Furthermore, the PU $\{\varphi_i\}$ generated by (2.2) will again be piecewise linear for $l = 1$ just like their FE counterpart in the GFEM (see Figure 6.2). Hence, in this situation our method does reconstruct functions φ_i that are identical to bilinear finite element functions, and also our general decomposition algorithm will recover the corresponding geometric elements. Hence, the number of integrals to be evaluated here with our method or a FEM are the same. We give the average number

$$a_{C_\Omega} := \frac{\sum_{i=1, j\in N_i}^{\text{card}(P)} \text{card}(D_{\omega_{ij}})}{\sum_{i=1}^{\text{card}(P)} \text{card}(N_i)} \tag{6.2}$$

of integration cells per nonzero block a_{ij} of the stiffness matrix in Tables 6.4 and 6.5. Furthermore, we give the average number

$$s_{C_\Omega} := \frac{\sum_{i=1}^{\text{card}(P)} \text{card}(N_i)}{\text{card}(P)} \tag{6.3}$$

of nonzero blocks a_{ij} per block-row of the stiffness matrix in Table 6.3 which corresponds to the number of entries in a FE stencil. For a one-dimensional uniform grid the average a_{C_Ω} is $\frac{4}{3}$ for hat functions (at interior points, where we have $N_i = \{\omega_{i-1}, \omega_i, \omega_{i+1}\}$). This situation corresponds of course to the case $l = 1$, $m = 1$, $\alpha_l = 2$ in Algorithm 3. Due to the tensor product approach, we have $(\frac{4}{3})^d$ as the optimal ratio of integration cells to nonzero blocks for $l = 1$ in the d-dimensional case. We certainly cannot expect to meet this optimal ratio for an irregular cover.

From the averages displayed in Tables 6.4 and 6.5 for the two-dimensional and three-dimensional case we see that the averages a_{C_Ω} are (almost) independent of the

Table 6.5

The average number a_{C_Ω} (6.2) of integration cells per nonzero block a_{ij} of the stiffness matrix for the different cover construction algorithms. Algorithm 2 with $k = \infty$, $x^i_L = l^i_L + \frac{1}{2}(u^i_L - l^i_L)$ for generated points x_L and $h^i_L = \frac{5}{4}\max\{u^i_L - x^i_L, x^i_L - l^i_L\}$ (A^l_2), and Algorithm 3 with $k = \infty$ (A^l_{3,α_l}) using a linear, a quadratic and a cubic B-spline during the Shepard construction (2.2). The initial point set P was Halton$^{N-1}_0(2,3,5)$ in $[0,1]^3$.

N	card(P)	A^1_2	$A^1_{3,2}$	$A^1_{3,1.3}$	A^2_2	$A^2_{3,1.5}$	A^3_2	$A^3_{3,2}$
1024	3543	250.03	16.34	20.71	464.61	17.61	877.89	29.19
8192	26699	302.20	15.96	22.48	553.66	18.59	1048.01	29.41
65536	199417	363.77	12.09	20.63	668.04	16.59	563.76	24.63
524288	1694568	–	12.44	26.52	–	15.72	–	25.07

number of points N of the initial point set P for Algorithm 3 as well as for Algorithm 2. Furthermore, we clearly see the substantial reduction in the number of integration cells for Algorithm 3 compared with Algorithm 2. The average a_{C_Ω} for Algorithm 3 drops by more than a factor of $\frac{1}{6}$ in two dimensions and by more than a factor of $\frac{1}{18}$ in three dimensions compared with the average a_{C_Ω} for covers constructed by Algorithm 2. This significant improvement in the number of integration cells is of course due to the alignment of the patches we have with Algorithm 3 but not with Algorithm 2; see Figures 6.1 and 3.2, and Figures 6.3 and 3.4. Another advantage of Algorithm 3 over Algorithm 2 is the fact that due to the alignment of the weight subdivisions $\{\omega^q_k\}$ the aspect ratio and volume of every integration cell $D^n_{\omega_{ij}}$ is bounded. This is not the case for covers constructed with Algorithm 2. In fact, in the three-dimensional example of Table 6.5 our decomposition algorithm would have generated a number of integration cells with very large aspect ratios and almost vanishing volume in the case $N = 524288$ for a cover constructed with Algorithm 2.

The average a_{C_Ω} for covers from Algorithm 3 (with $l = 1$) is about three times the optimal ratio of $(\frac{4}{3})^d$. For one this factor can be explained by the sudden change in the spatial resolution of the cover, i.e., the level difference k of two neighboring patches, which leads to nonaligned subdivisions $\{\omega^q_k\}$ (see Figures 6.1 and 6.3), and therefore increases the number of integration cells. Moreover, however, the optimal ratio of $(\frac{4}{3})^d$ holds for interior patches only. For cover patches which overlap the boundary of the domain this ratio is 2 (or even 2.25 at corners) in two dimensions, since the subdivisions $\{\omega^q_k\}$ are aligned only with each other but *not* with the boundary $\partial\Omega$; see Figure 6.4. Hence, we expect the average number of integration cells a_{C_Ω} to decrease for larger N since the volume to surface ratio improves. This can be observed from the numbers $A^{1_1}_3$ displayed in Tables 6.4 and 6.5. A similar argument can be made in the case of a quadratic B-spline.[10]

When we use a cubic B-spline ($l = 3$) we construct smooth approximations $u_{\mathrm{PU}} \in \mathcal{C}^2$, and the optimal ratio for interior patches is $(\frac{8}{3})^d$; i.e., it is about 7 in two dimensions and 19 in three dimensions. The averages a_{C_Ω} given in Tables 6.4 and 6.5 in this case are about 10 in two dimensions and 26 in three dimensions; i.e., they are closer to their optimal ratio of $(\frac{8}{3})^d$ than a_{C_Ω} is to its optimum for the linear B-spline. This can be explained by the fact that for $l = 3$ the boundary effect mentioned

[10] In the case $l = 2$, we have an optimal ratio of $\frac{5}{3}$ in one dimension. However, due to the fact that the overlap $m = n$ for $l = 2n$ is smaller in relation to the overall number of cells card$(\{\omega^q_k\})$ (see Figure 6.1), the generalization of this optimum to higher dimensions is *not* given by $(\frac{5}{3})^d$; e.g., in two dimensions the optimum is $\frac{22}{9}$ only; see Figure 6.4.

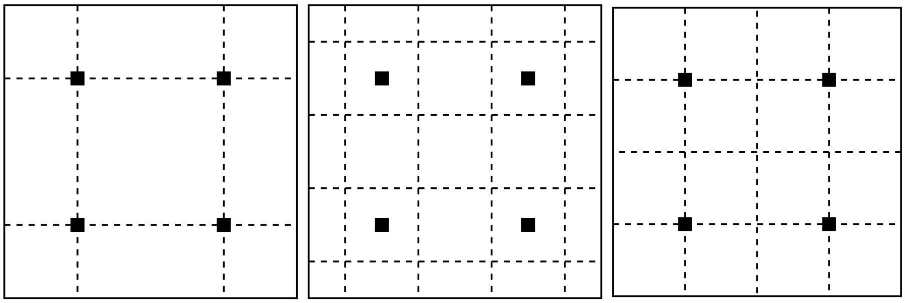


Fig. 6.4. *The integration cells induced by four neighboring linear B-splines (left), quadratic B-splines (center), and cubic B-splines (right) in two dimensions. The integration cells $D^n_{\omega_{ij}}$ align with the boundary for the cubic B-spline (right) but not for the linear (left) or quadratic (center) B-spline due to the overlap condition for the support patches ω_i.*

above does not exist; see Figure 6.4. Here, the cells $\{\omega^q_k\}$ induced by the cubic spline align with the boundary of $[0,1]^d$. Hence, only the irregularity of the cover causes an increase in the number of integration cells. Overall, the number of integration cells for an approximate solution $u_{\mathrm{PU}} \in \mathcal{C}^2$ is about twice the number of integration cells we have when we construct an approximate solution $u_{\mathrm{PU}} \in \mathcal{C}^0$.

So far we were concerned only with the computational cost during the integration and the influence the shape functions $\varphi_i\psi^k_i$ have on the computational efficiency of our PUM. Another important issue, however, is the stability of the basis of our PUM space. Here, we also have to address the question of whether the functions $\varphi_i\psi^k_i$ are indeed a basis. In the case of $l = 1$ and $\alpha_l = 2$ the alignment of the cover patches ω_i and their respective weight subdivisions $\{\omega^q_i\}$ leads to the reconstruction of the FE hat functions for the PU. Hence, our PUM reduces to the GFEM in this situation. It is well known [2, 3, 27, 30] that the GFEM (in general) generates linear-dependent shape functions $\varphi_i\psi^k_i$, the so-called nullity of the method. This is essentially due to the fact that in the GFEM the PU functions φ_i already reconstruct the linear polynomial.

Consider the one-dimensional situation, where we have one element and two nodes with their associated hat function as φ_i. Assume that we use linear polynomials as local approximations spaces V_i. The shape functions $\varphi_i\psi^k_i$ are (global) polynomials due to this construction. The number of shape functions is four, and the maximal polynomial degree is two. Since the quadratic polynomials in one dimension can be generated by three basis functions, we see that the GFEM shape functions are linear dependent.

With our approach, the φ_i reconstruct the linear polynomial only away from the boundary; close to the boundary we have $\varphi_i \equiv 1$. Therefore, the shape functions are not linear dependent. However, since the small boundary layer where $\varphi_i \equiv 1$ decreases with larger N the condition number κ of the mass matrix is dependent on N; i.e., the basis is no longer stable. A simple cure for this stability problem is to use $m < 1$ in (6.1) when we have $l = 1$; i.e., we limit ourselves to $1 < \alpha_l < 2$ when $l = 1$. With $\alpha_l < 2$ we can find a subpatch $\tilde{\omega}_i \subset \omega_i$, where $\varphi_i|_{\tilde{\omega}_i} \equiv 1$ for many i. Therefore, the PU functions φ_i no longer reconstruct the linear polynomial independent of N, and the resulting shape functions form a stable basis. We therefore allow for any value $1 < \alpha_l < 2$ in Algorithm 3 if $l = 1$. The number of integration cells increases somewhat due to this generalization. The patches ω_i are still aligned, but their respective weight subdivisions are not. The optimal ratio increases from $(\frac{4}{3})^d$ to 2^d. From the averages displayed in Tables 6.4 and 6.5 we see that a_{C_Ω} for the choice of $\alpha_l = 1.3$ is about a factor of 1.5 to 2 larger than the average is for the optimal choice of $\alpha_l = 2$. However, the number of integration cells is still substantially less compared with the covers from Algorithm 2.

A similar problem arises for higher order splines $l > 1$ only when $\alpha_l > 2$; e.g., we need $\alpha_l = 4$ with $l = 2$. Therefore, we can stay with our choices of $\alpha_l = 1.5$ if $l = 2$ and $\alpha_l = 2$ if $l = 3$.

7. Numerical experiments. In this section we present some results of our numerical experiments for an elliptic PDE using the h-version and the p-version of our PUM.

We apply our PUM to elliptic problems on the unit cell $\Omega = (0,1)^d$ in two and three dimensions. Here, we consider the Laplace equation

$$-\Delta u = f \tag{7.1}$$

with Dirichlet boundary conditions $u = g$ on $\partial\Omega$ and the equation

$$-\Delta u + u = f \tag{7.2}$$

of Helmholtz type with Neumann boundary conditions $\nabla u = g$ on $\partial\Omega$. Furthermore, we use the presented method to study heat conduction in lattice materials [20, 26] in three dimensions.

The local approximation spaces $V_i^{p_i}$ used in our numerical experiments are complete Legendre polynomials with $p_i = p$ for all patches ω_i. The weight functions W_i used in the Shepard construction (2.2) are linear splines ($l = 1$, $\alpha_l = 1.3$). We give the relative error

$$e = \frac{\|u - u_{\mathrm{PU}}\|}{\|u\|}$$

in the L^∞-, L^2-, and the energy-norm, which is computed with the help of the integration scheme presented in section 5. Moreover, we also give the convergence rates

$$\rho = \frac{\log\left(\frac{\|u-u_{\mathrm{PU},L}\|}{\|u-u_{\mathrm{PU},L-1}\|}\right)}{\log\left(\frac{\mathsf{dof}_L}{\mathsf{dof}_{L-1}}\right)},$$

where $\mathsf{dof} := \sum \dim(V_i^{p_i})$, with respect to two successive refinement levels L and $L-1$. These convergence rates ρ correspond to an algebraic error estimate

$$\|u - u_{\mathrm{PU},L}\| = O\left(\mathsf{dof}_L^{\rho}\right) \tag{7.3}$$

which is valid for the h-version of the PUM [2, 3]. We can relate these rates ρ to the common h^α notation by $\alpha = -\rho d$, since $N^{-\frac{1}{d}} \sim h$ for uniform point sets. Hence, the optimal convergence rates ρ for our PUM based on uniform point sets and linear polynomials are $\rho_2 = -1$ and $\rho_E = -\frac{1}{2}$ in two dimensions ($\rho_2 = -\frac{2}{3}$ and $\rho_E = -\frac{1}{3}$ in three dimensions). For the pointwise convergence[11] we have $\rho_2 < \rho_\infty < \rho_E$.

[11] In the FEM we have the estimate $\|u - u_h\|_\infty = O\left(h^2|\log h|^{\mu(d)}\right)$, where $\mu(2) = 1$ and $\mu(d) = \frac{d}{4} + 1$ for $d \geq 3$ [25, 28]. The L^∞-norm is usually approximated by the maximum over the nodal values, where we can observe a superconvergence of order h^2. For our approximation to the L^∞-norm, however, we do not use the points $x_i \in P$ but all quadrature points since the PUM shape functions $\varphi_i \psi_i^n$ are noninterpolatory and the Legendre polynomials ψ_i^n of odd degree vanish at x_i. Hence, we cannot expect to measure h^2 superconvergence in the L^∞-norm.

Table 7.1

Errors (e) and convergence rates (ρ^A) in different norms for problem (7.1) *in two dimensions with solution* (7.5).

N	card(P)	p	dof	e_∞	ρ_∞^A	e_2	ρ_2^A	e_E	ρ_E^A
16	28	1	84	5.739_{-2}	–	6.462_{-2}	–	2.676_{-1}	–
64	106	1	318	1.817_{-2}	-8.639_{-1}	1.726_{-2}	-9.917_{-1}	1.403_{-1}	-4.850_{-1}
256	406	1	1218	5.700_{-3}	-8.633_{-1}	4.405_{-3}	-1.017_{0}	6.963_{-2}	-5.217_{-1}
1024	1729	1	5187	1.926_{-3}	-7.488_{-1}	1.072_{-3}	-9.753_{-1}	3.462_{-2}	-4.823_{-1}
4096	6364	1	19092	1.101_{-3}	-4.291_{-1}	2.749_{-4}	-1.044_{0}	1.762_{-2}	-5.183_{-1}
16384	27673	1	83019	4.431_{-4}	-6.193_{-1}	6.749_{-5}	-9.555_{-1}	8.677_{-3}	-4.819_{-1}
65536	101314	1	303942	2.099_{-4}	-5.757_{-1}	1.716_{-5}	-1.055_{0}	4.374_{-3}	-5.278_{-1}

Table 7.2

Errors (e) and convergence rates (ρ) in different norms for problem (7.2) *in two dimensions with solution* (7.5).

N	card(P)	p	dof	e_∞	ρ_∞	e_2	ρ_2	e_E	ρ_E
64	106	1	318	2.942_{-2}	–	1.476_{-2}	–	1.341_{-1}	–
64	106	2	636	7.960_{-4}	-5.208_{0}	3.448_{-4}	-5.420_{0}	5.761_{-3}	-4.541_{0}
64	106	3	1060	1.465_{-5}	-7.821_{0}	1.046_{-5}	-6.843_{0}	1.986_{-4}	-6.592_{0}
64	106	4	1590	9.093_{-8}	-1.253_{1}	1.091_{-7}	-1.125_{1}	2.612_{-6}	-1.068_{1}
64	106	5	2226	8.003_{-9}	-7.223_{0}	2.640_{-9}	-1.106_{1}	7.121_{-8}	-1.071_{1}
64	106	6	2968	8.579_{-10}	-7.762_{0}	2.588_{-10}	-8.073_{0}	8.429_{-9}	-7.418_{0}
64	106	7	3816	3.710_{-10}	-3.336_{0}	6.482_{-11}	-5.509_{0}	2.481_{-9}	-4.866_{0}
64	106	8	4770	9.538_{-11}	-6.087_{0}	2.183_{-11}	-4.877_{0}	9.217_{-10}	-4.437_{0}
64	106	9	5830	4.137_{-11}	-4.163_{0}	8.451_{-12}	-4.729_{0}	4.100_{-10}	-4.037_{0}
64	106	10	6996	1.401_{-11}	-5.939_{0}	3.075_{-12}	-5.545_{0}	1.700_{-10}	-4.829_{0}
64	106	11	8268	8.220_{-12}	-3.192_{0}	1.535_{-12}	-4.159_{0}	8.807_{-11}	-3.937_{0}
64	106	12	9646	3.836_{-12}	-4.944_{0}	7.533_{-13}	-4.618_{0}	4.751_{-11}	-4.004_{0}

For the p-version, we expect an exponential convergence for smooth solutions u since the local error estimate

$$\|u - u_{\mathrm{PU},L}\| = O\left(\exp\left(-b\sqrt{\mathsf{dof}_L}\right)\right) \tag{7.4}$$

holds on every cover patch ω_i for smooth solutions u.

Example 1 (unit square). In our first example we consider the Dirichlet problem (7.1) in $\Omega = (0,1)^2$. We choose f and g such that the solution u is given by

$$u(x) = \|x\|_2^5. \tag{7.5}$$

We apply the h-version of our PUM with linear polynomials to approximate (7.1). The covers C_Ω are generated using N points of the Halton(2, 3) sequence with increasing N. They exhibit a somewhat locally varying patch size; see Figure 6.1. Consequently, there will be some fluctuation in the measured convergence rates ρ.

The results for the h-version experiment with linear polynomials are given in Table 7.1. The measured rates ρ show the algebraic convergence (7.3) of our PUM in the L^2- and energy-norm with rates ρ near the optimal values of $\rho_2 = -1$ and $\rho_E = -\frac{1}{2}$ for the h-version. In the L^∞-norm we measure a convergence rate ρ_∞ of -0.6 which is between -1 and $-\frac{1}{2}$ as expected.[11]

Let us now consider the Neumann problem (7.2). Again, the solution is given by (7.5). Here, we apply the p-version of our PUM. Since the solution (7.5) is not analytic in Ω we may not expect an exponential convergence of the p-version. From the numbers given in Table 7.2 and the convergence history displayed in Figure 7.1

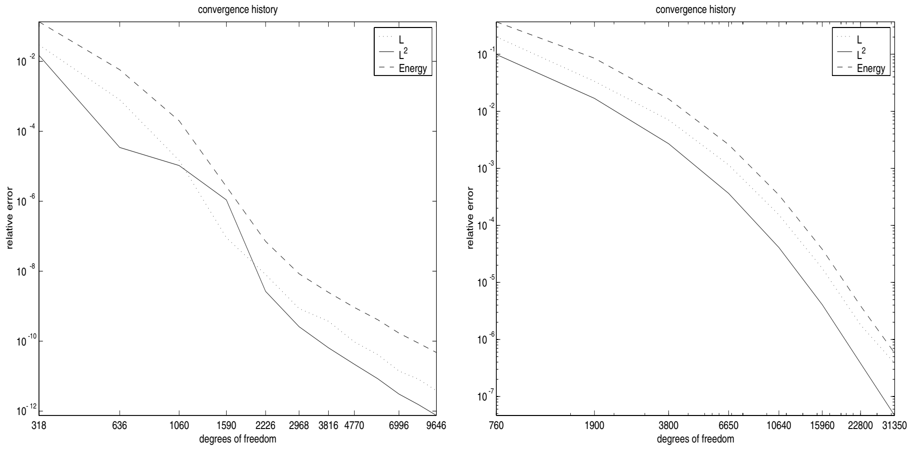


FIG. 7.1. *Convergence history for problem* (7.2) *in two dimensions (left) with solution* (7.5). *Convergence history for problem* (7.2) *in three dimensions (right) with solution* (7.6).

TABLE 7.3

Errors (e) and convergence rates (ρ^A) in different norms for problem (7.1) *in three dimensions with solution* (7.6).

N	card(P)	p	dof	e_∞	ρ^A_∞	e_2	ρ^A_2	e_E	ρ^A_E
16	50	1	200	4.565_{-1}	–	1.182_0	–	1.828_0	–
128	414	1	1656	2.580_{-1}	-2.699_{-1}	1.499_{-1}	-9.769_{-1}	4.273_{-1}	-6.876_{-1}
1024	3543	1	14172	7.914_{-2}	-5.505_{-1}	5.452_{-2}	-4.711_{-1}	2.683_{-1}	-2.168_{-1}
8192	26699	1	106796	3.119_{-2}	-4.610_{-1}	9.091_{-3}	-8.869_{-1}	1.197_{-1}	-3.996_{-1}

(left) we can observe that the convergence starts off at an exponential rate but breaks down to a polynomial rate as anticipated for higher polynomial degrees p.

Example 2 (unit cube). In our second example we consider (7.1) with Dirichlet boundary conditions on the unit cube in three dimensions. Here, we now choose f and g such that the solution u is given by

$$u(x) = \exp\left(4\|x\|_1\right). \tag{7.6}$$

Again, we use the h-version of our PUM with linear polynomials to approximate (7.1). The covers are generated using N points of the Halton$(2, 3, 5)$ sequence. The numerical results are given in Table 7.3. Here, it seems that the convergence in the L^2- and the energy-norm is actually better than the theory (for uniform point sets) suggests. We measure a convergence rate ρ_2 close to -1 but would expect only a convergence rate of about $-\frac{2}{3}$. This behavior, however, is due to the fact that the size of patches ω_i based on a Halton sequence varies much more in three dimensions than in two dimensions (for a small number of samples). Therefore, a larger fluctuation in the measured convergence rates ρ will occur. If we use a different number N of Halton points for the initial cover or refine the covers using only twice as many points instead of eight times as many, this behavior becomes much more obvious. To this end we also give the numerical results of an h-version experiment with linear polynomials applied to the Neumann problem (7.2) with solution (7.6) in Table 7.4. Here, we clearly see the fluctuation in the convergence rates due to the unstructured refinement induced by the Halton sequence. If we use the grid points of a uniform grid to generate the

Table 7.4

Errors (e) and convergence rates (ρ^A) in different norms for problem (7.2) *in three dimensions with solution* (7.6)*. The covers C_Ω are based on a Halton*(2, 3, 5) *point set.*

N	card(P)	p	dof	e_∞	ρ^A_∞	e_2	ρ^A_2	e_E	ρ^A_E
16	50	1	200	7.184_{-1}	–	4.321_{-1}	–	7.741_{-1}	–
32	106	1	424	4.125_{-1}	-7.383_{-1}	1.384_{-1}	-1.515_0	4.846_{-1}	-6.232_{-1}
64	190	1	760	2.020_{-1}	-1.224_0	9.887_{-2}	-5.766_{-1}	3.701_{-1}	-4.618_{-1}
128	414	1	1656	1.966_{-1}	-3.457_{-2}	8.198_{-2}	-2.405_{-1}	3.351_{-1}	-1.277_{-1}
256	673	1	2692	1.838_{-1}	-1.387_{-1}	3.729_{-2}	-1.621_0	2.630_{-1}	-4.985_{-1}
512	1415	1	5660	8.673_{-2}	-1.011_0	3.464_{-2}	-9.924_{-2}	2.364_{-1}	-1.434_{-1}
1024	3543	1	14172	8.565_{-2}	-1.366_{-2}	2.628_{-2}	-3.008_{-1}	1.936_{-1}	-2.180_{-1}
2048	5874	1	23496	6.756_{-2}	-4.692_{-1}	9.380_{-3}	-2.038_0	1.340_{-1}	-7.267_{-1}
4096	10606	1	42424	6.734_{-2}	-5.719_{-3}	9.039_{-3}	-6.263_{-2}	1.296_{-1}	-5.676_{-2}
8192	26699	1	106796	3.934_{-2}	-5.821_{-1}	7.021_{-3}	-2.738_{-1}	1.016_{-1}	-2.643_{-1}
16384	45151	1	180604	2.197_{-2}	-1.109_0	2.334_{-3}	-2.096_0	6.745_{-2}	-7.790_{-1}

Table 7.5

Errors (e) and convergence rates (ρ^A) in different norms for problem (7.2) *in three dimensions with solution* (7.6)*. The covers C_Ω are based on a uniform point set.*

N	card(P)	p	dof	e_∞	ρ^A_∞	e_2	ρ^A_2	e_E	ρ^A_E
64	64	1	256	4.124_{-1}	–	1.382_{-1}	–	4.846_{-1}	–
512	512	1	2048	1.838_{-1}	-3.887_{-1}	3.734_{-2}	-6.295_{-1}	2.634_{-1}	-2.931_{-1}
4096	4096	1	16384	6.751_{-2}	-4.816_{-1}	9.409_{-3}	-6.628_{-1}	1.351_{-1}	-3.211_{-1}
32768	32768	1	131072	2.199_{-2}	-5.395_{-1}	2.332_{-3}	-6.708_{-1}	6.787_{-2}	-3.311_{-1}

covers C_Ω we can observe a very good correspondence of the measured convergence rates ρ with those of the theory (see Table 7.5)), where the corresponding numerical results for the Neumann problem (7.2) with solution (7.6) are given.

We now turn to the p-version of our PUM in three dimensions. The numerical results for problem (7.2) with Neumann boundary conditions are given in Table 7.6. For the smooth solution (7.6) we expect an exponential convergence behavior of the p-version of our PUM. From the measured values of ρ and the convergence history given in Figure 7.1 (right) we clearly see this behavior.

In summary we have that on simple domains in two and three dimensions the PUM works with the anticipated convergence properties. Now we turn to more complicated (yet still academic) computational domains in three dimensions.

Example 3 (lattice material). In our third example we study the problem of heat conduction in a lattice material. To this end, we use our PUM to discretize the PDE

$$-\Delta u + u = f \text{ in } \Omega \subset (0,1)^3$$

on a domain Ω which describes a lattice material with a characteristic cell width of $\frac{4}{14}$ and a cell number of 3. As boundary conditions we use the Neumann conditions

$$\nabla u = g = \begin{cases} -10(\frac{1}{4} - 2x): & \Gamma_I := \{x \in \partial\Omega \,|\, x_0 = 0 \text{ and } \|x\|^2 < \frac{1}{4}\}, \\ 0: & \Gamma_O := \partial\Omega \setminus \Gamma_I \end{cases}$$

to simulate heat-introduction into the material at the contact points Γ_I to a heat source and outflow conditions on the remaining boundary Γ_O.

We use a $\text{Halton}_0^{4095}(2,3,5)$ set distributed in the bounding box $R_\Omega := (0,1)^3$ as the initial point set P for our cover construction. The local approximation spaces $V_i^{p_i}$

TABLE 7.6

Errors (e) and convergence rates (ρ) in different norms for problem (7.2) *in three dimensions with solution* (7.6).

N	card(P)	p	dof	e_∞	ρ_∞	e_2	ρ_2	e_E	ρ_E
64	190	1	760	2.020_{-1}	$-$	9.887_{-2}	$-$	3.701_{-1}	$-$
64	190	2	1900	3.355_{-2}	-1.959_0	1.682_{-2}	-1.933_0	8.527_{-2}	-1.602_0
64	190	3	3800	7.049_{-3}	-2.251_0	2.698_{-3}	-2.640_0	1.644_{-2}	-2.375_0
64	190	4	6650	1.150_{-3}	-3.240_0	3.631_{-4}	-3.584_0	2.606_{-3}	-3.291_0
64	190	5	10640	1.536_{-4}	-4.283_0	4.095_{-5}	-4.643_0	3.408_{-4}	-4.328_0
64	190	6	15960	1.763_{-5}	-5.339_0	4.062_{-6}	-5.699_0	3.810_{-5}	-5.404_0
64	190	7	22800	1.800_{-6}	-6.397_0	3.809_{-7}	-6.636_0	3.882_{-6}	-6.403_0
64	190	8	31350	3.818_{-7}	-4.869_0	4.652_{-8}	-6.603_0	5.759_{-7}	-5.992_0

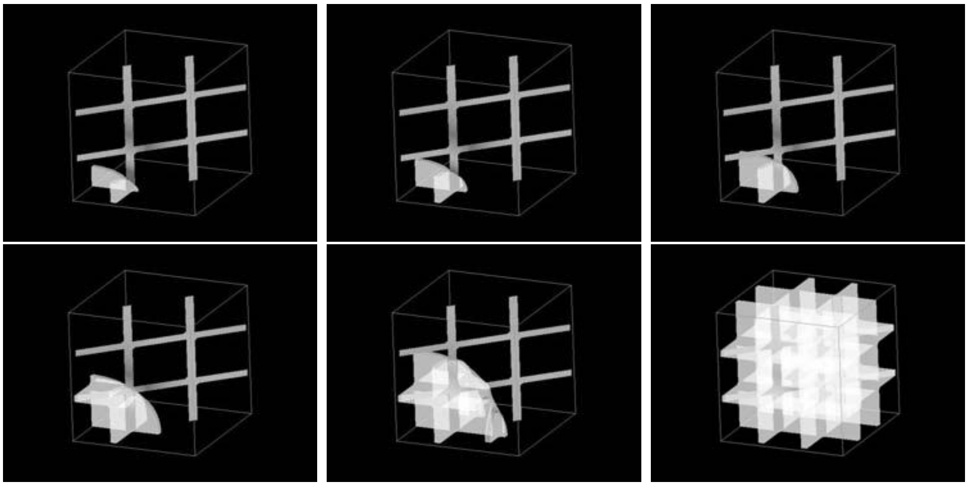


FIG. 7.2. *Isosurfaces of approximate solution and diagonal slice through lattice domain.*

that are assigned to the 5187 patches which overlap the computational domain Ω are quadratic Legendre polynomials. In Figure 7.2 some isosurfaces of the computed solution and a diagonal slice through the material are displayed. From these illustrations we observe the heat-introduction into the material at the contact points Γ_I and the heat-propagation through the material. We clearly see the expected radial shape of the isosurfaces due to the prescribed profile of the Neumann boundary conditions on Γ_I.

8. Concluding remarks. We presented a meshfree Galerkin method for the discretization of a PDE. The method is based on the PU approach and utilizes a novel tree-based cover construction algorithm. The introduction of this algorithm for the cover construction problem and the presented numerical quadrature scheme—both of which are applicable to general domains—improve the computational efficiency during the assembly of the stiffness matrix substantially.

The results of our numerical experiments showed the exponential convergence of the p-version of our PUM for smooth solutions and the anticipated algebraic convergence of the h-version in two and three dimensions. Furthermore, we applied our PUM to the heat conduction problem in lattice materials. Although these examples are still of academic nature, we believe that the substantial improvement of the computation efficiency due to the ideas and algorithms presented in this paper makes the treatment of real world problems with meshfree Galerkin methods seizable in the near future, at least for Neumann boundary conditions. The implementation of Dirichlet boundary conditions in MMs is in general a challenging problem which needs further investigation.

The presented hierarchical cover construction algorithm not only reduces the computational costs significantly but also introduces a hierarchy on the PUM function space. This may be exploited in the development of multilevel solvers [14]. The parallelization of our PUM [15] may be simplified by the tree-based cover construction. Here, we can apply a space filling curves parallelization approach [6] which allows for a cheap dynamic load balancing strategy.

REFERENCES

[1] N. R. Aluru, *A point collocation method based on reproducing kernel approximations*, Internat. J. Numer. Methods Engrg., 47 (2000), pp. 1083–1121.

[2] I. Babuška and J. M. Melenk, *The partition of unity finite element method: Basic theory and applications*, Comput. Methods Appl. Mech. Engrg., 139 (1996), pp. 289–314.

[3] I. Babuška and J. M. Melenk, *The partition of unity method*, Internat. J. Numer. Methods Engrg., 40 (1997), pp. 727–758.

[4] T. Belytschko, Y. Y. Lu, and L. Gu, *Element-free Galerkin methods*, Internat. J. Numer. Methods Engrg., 37 (1994), pp. 229–256.

[5] M. Bern, D. Eppstein, and J. Gilbert, *Provably good mesh generation*, J. Comput. System Sci., 48 (1994), pp. 384–409.

[6] A. Caglar, M. Griebel, M. A. Schweitzer, and G. Zumbusch, *Dynamic load-balancing of hierarchical tree algorithms on a cluster of multiprocessor PCs and on the Cray T3E*, in Proceedings of the 14th Supercomputer Conference, Mannheim, Germany, 1999, H. W. Meuer, ed., Mateo, Mannheim, Germany, 1999.

[7] C. A. M. Duarte, I. Babuška, and J. T. Oden, *Generalized finite element methods for three dimensional structural mechanics problems*, Comput. Structures, 77 (2000), pp. 215–232.

[8] C. A. M. Duarte and J. T. Oden, *hp clouds—A meshless method to solve boundary value problems*, Numer. Methods Partial Differential Equations, 12 (1996), pp. 673–705.

[9] G. Fasshauer, *Solving differential equations with radial basis functions: Multilevel methods and smoothing*, Adv. Comp. Math., 11 (1999), pp. 139–159.

[10] C. Franke and R. Schaback, *Convergence orders of meshless collocation methods using radial basis functions*, Adv. Comput. Math., 8 (1998), pp. 381–399.

[11] C. Franke and R. Schaback, *Solving partial differential equations by collocation using radial basis functions*, Appl. Math. Comput., 93 (1998), pp. 73–82.

[12] T. Gerstner and M. Griebel, *Numerical integration using sparse grids*, Numer. Algorithms, 18 (1998), pp. 209–232.

[13] M. Griebel and M. A. Schweitzer, *A particle-partition of unity method for the solution of elliptic, parabolic and hyperbolic PDEs*, SIAM J. Sci. Comput., 22 (2000), pp. 853–890.

[14] M. Griebel and M. A. Schweitzer, *A Particle-Partition of Unity Method—Part* III*: A Multilevel Solver*, SFB Preprint, Sonderforschungsbereich 256, Institut für Angewandte Mathematik, Universität Bonn, Bonn, Germany, 2001, submitted.

[15] M. Griebel and M. A. Schweitzer, *A particle-partition of unity method—part* IV: *Parallelization*, in Meshfree Methods for Partial Differential Equations, Lect. Notes Comput. Sci. Eng., Springer, 2002, to appear.

[16] F. C. Günther and W. K. Liu, *Implementation of boundary conditions for meshless methods*, Comput. Methods Appl. Mech. Engrg., 163 (1998), pp. 205–230.

[17] X. Han, S. Oliveira, and D. Stewart, *Finding sets covering a point with application to mesh-free Galerkin methods*, SIAM J. Comput., 30 (2000), pp. 1368–1383.

[18] O. Klaas and M. S. Shepard, *Automatic generation of octree-based three-dimensional discretizations for partition of unity methods*, Comput. Mech., 25 (2000), pp. 296–304.

[19] T. J. Liszka, C. A. M. Duarte, and W. W. Tworzydlo, *hp-meshless cloud method*, Comput. Methods Appl. Mech. Engrg., 139 (1996), pp. 263–288.

[20] A.-M. Matache, I. Babuška, and C. Schwab, *Generalized p-FEM in homogenization*, Numer. Math., 86 (2000), pp. 319–375.

[21] J. J. Monaghan, *Why Particle Methods Work*, SIAM J. Sci. Statist. Comput., 3 (1982), pp. 422–433.

[22] J. J. Monaghan, *An introduction to SPH*, Comput. Phys. Comm., 48 (1988), pp. 89–96.

[23] H. Neunzert, A. Klar, and J. Struckmeier, *Particle Methods: Theory and Applications*, Tech. Rep. 95-153, Arbeitsgruppe Technomathematik, Universität Kaiserslautern, Kaiserslautern, Germany, 1995.

[24] E. Novak, K. Ritter, R. Schmitt, and A. Steinbauer, *On a recent interpolatory method for high dimensional integration*, J. Comput. Appl. Math., 15 (1999), pp. 499–522.

[25] R. Rannacher, *Zur L^∞-Konvergenz linearer finiter Elemente beim Dirichletproblem*, Math. Z., 149 (1976), pp. 69–77.

[26] C. Schwab and A.-M. Matache, *High order generalized FEM for lattice materials*, in Proceedings of the 3rd European Conference on Numerical Mathematics and Advanced Applications, Finland, 1999, P. Neittaanmäki, T. Tiihonen, and P. Tarvainen, eds., World S cientific, Singapore, 2000, pp. 58–71.

[27] M. A. Schweitzer, *Ein Partikel–Galerkin–Verfahren mit Ansatzfunktionen der Partition of Unity Method*, Diplomarbeit, Institut für Angewandte Mathematik, Universität Bonn, Bonn, Germany, 1997.

[28] R. Scott, *Optimal L^∞-estimates for the finite element method on irregular meshes*, Math. Comp., 30 (1976), pp. 681–697.

[29] G. Strang and G. J. Fix, *An Analysis of the Finite Element Method*, Prentice–Hall, Englewood Cliffs, NJ, 1973.

[30] T. Strouboulis, I. Babuška, and K. Copps, *The design and analysis of the generalized finite element method*, Comput. Methods Appl. Mech. Engrg., 181 (1998), pp. 43–69.

SIAM J. SCI. COMPUT.
Vol. 23, No. 5, pp. 1683–1706

NAVIER–STOKES EQUATIONS IN ROTATION FORM: A ROBUST MULTIGRID SOLVER FOR THE VELOCITY PROBLEM*

MAXIM A. OLSHANSKII† AND ARNOLD REUSKEN‡

Abstract. The topic of this paper is motivated by the Navier–Stokes equations in *rotation* form. Linearization and application of an implicit time stepping scheme results in a linear stationary problem of Oseen type. In well-known solution techniques for this problem such as the Uzawa (or Schur complement) method, a subproblem consisting of a coupled nonsymmetric system of linear equations of diffusion-reaction type must be solved to update the velocity vector field. In this paper we analyze a standard finite element method for the discretization of this coupled system, and we introduce and analyze a multigrid solver for the discrete problem. Both for the discretization method and the multigrid solver the question of robustness with respect to the amount of diffusion and variation in the convection field is addressed. We prove stability results and discretization error bounds for the Galerkin finite element method. We present a convergence analysis of the multigrid method which shows the robustness of the solver. Results of numerical experiments are presented which illustrate the stability of the discretization method and the robustness of the multigrid solver.

Key words. finite elements, multigrid, convection-diffusion, Navier–Stokes equations, rotation form, vorticity

AMS subject classifications. 65N30, 65N55, 76D17, 35J55

PII. S1064827500374881

1. Introduction.

The incompressible Navier–Stokes problem written in velocity-pressure variables has several equivalent formulations. Very popular is the *convection* form of the problem: find velocity $\mathbf{u}(t,\mathbf{x})$ and kinematic pressure $p(t,\mathbf{x})$ such that

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} - \nu\Delta\mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p = \mathbf{f} \quad &\text{in} \quad \Omega\times(0,T], \\ \operatorname{div}\mathbf{u} = 0 \quad &\text{in} \quad \Omega\times(0,T], \end{aligned} \tag{1.1}$$

with given force field $\mathbf{f}$ and viscosity $\nu > 0$. Suitable boundary and initial conditions have to be added to (1.1). One alternative to (1.1) is the *rotation* form of the Navier–Stokes problem:

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} - \nu\Delta\mathbf{u} + (\operatorname{curl}\mathbf{u})\times\mathbf{u} + \nabla P = \mathbf{f} \quad &\text{in} \quad \Omega\times(0,T], \\ \operatorname{div}\mathbf{u} = 0 \quad &\text{in} \quad \Omega\times(0,T], \end{aligned} \tag{1.2}$$

which results from (1.1) after replacing the kinematic pressure by the Bernoulli (or dynamic, or total; cf., e.g., [18]) pressure $P = p + \frac{1}{2}\mathbf{u}\cdot\mathbf{u}$ and using the identity $(\mathbf{u}\cdot\nabla)\mathbf{u} = (\operatorname{curl}\mathbf{u})\times\mathbf{u} + \frac{1}{2}\nabla(\mathbf{u}\cdot\mathbf{u})$. In the three-dimensional case × stands for the vector product and $\operatorname{curl}\mathbf{u} := \nabla\times\mathbf{u}$. In two dimensions, $\operatorname{curl}\mathbf{u} := -\frac{\partial u_1}{\partial x_2} + \frac{\partial u_2}{\partial x_1}$ and

*Received by the editors July 10, 2000; accepted for publication (in revised form) September 3, 2001; published electronically January 30, 2002.

http://www.siam.org/journals/sisc/23-5/37488.html

†Department of Mechanics and Mathematics, Moscow State University, Moscow 119899, Russia (ay@olshan.msk.ru). The research of this author was partially supported by RFBR grant 99-01-00263. Part of this author's work was done while he was a visiting researcher at RWTH University in Aachen.

‡Institut für Geometrie und Praktische Mathematik, RWTH-Aachen, D-52056 Aachen, Germany (reusken@igpm.rwth-aachen.de).

$a \times \mathbf{u} := (-au_2, au_1)^T$ for a scalar a. Linearization and application of an implicit time stepping scheme to (1.2) results in an Oseen-type problem in which the equations are of the form

$$\begin{aligned} -\nu\Delta\mathbf{u} + \mathbf{w}\times\mathbf{u} + \alpha\mathbf{u} + \nabla P = \mathbf{f} \quad &\text{in} \quad \Omega \\ \operatorname{div}\mathbf{u} = 0 \quad &\text{in} \quad \Omega, \end{aligned} \tag{1.3}$$

with $\alpha \geq 0$ and $\mathbf{w} = \operatorname{curl}\mathbf{a}$, where $\mathbf{a}$ is a known approximation of $\mathbf{u}$. Note that the above linearization of $(\operatorname{curl}\mathbf{u})\times\mathbf{u}$ ensures the ellipticity of (1.3) in a certain sense (cf. section 2). One strategy to solve (1.3) is an Uzawa-type algorithm, in which a Schur complement problem $\mathbf{S}_{\mathrm{rot}}P = \tilde{\mathbf{g}}$ for the pressure has to be solved. The Schur complement operator has the formal representation $\mathbf{S}_{\mathrm{rot}} = -\operatorname{div}(-\nu\Delta + \mathbf{w}\times + \alpha I)^{-1}\nabla$. The operator $(-\nu\Delta + \mathbf{w}\times + \alpha I)^{-1}$ in this Schur complement is the solution operator of the problem

$$\begin{aligned} -\nu\Delta\mathbf{u} + \mathbf{w}\times\mathbf{u} + \alpha\mathbf{u} = \mathbf{f} \quad &\text{in} \quad \Omega, \\ \mathbf{u} = 0 \quad &\text{on} \quad \partial\Omega, \end{aligned} \tag{1.4}$$

where, for simplicity, we used homogeneous Dirichlet boundary conditions. The exact solution of (1.4) can be replaced by a suitable approximation like in the inexact Uzawa method [3] or in block preconditioners for (1.3) (see, e.g., [11], [19]).

Linearization and application of an implicit time stepping scheme to the convection form (1.1) result in equations as in (1.3) with $\mathbf{w}\times\mathbf{u}$ replaced by $(\mathbf{a}\cdot\nabla)\mathbf{u}$. The Uzawa technique applied to this linear stationary problem for $\mathbf{u}$ and p corresponds to a Schur complement problem with operator $\mathbf{S}_{\mathrm{conv}} = -\operatorname{div}(-\nu\Delta + \mathbf{a}\cdot\nabla + \alpha I)^{-1}\nabla$. The operator $(-\nu\Delta + \mathbf{a}\cdot\nabla + \alpha I)^{-1}$ in this Schur complement is the solution operator of decoupled convection-diffusion(-reaction) problems. Hence in this approach an efficient solver for convection-diffusion equations is of major importance. In the setting of this paper we are particularly interested in finite element discretization methods and multigrid solvers for the discrete problem. There is extensive literature on these solution techniques for convection-diffusion problems; see, e.g., [1], [4], [9], [14], [15], [16], [20], [21], [23], and the references therein. Important topics are appropriate stabilization techniques for the finite element discretization and robustness of the multigrid solvers for convection dominated problems.

In this paper we study the problem (1.4), which can be seen as the counterpart, for the Navier–Stokes equations in rotation form, of the convection-diffusion problems that correspond to the Navier–Stokes problem in convection form. Note that, opposite to the convection-diffusion problems, the problem (1.4) is a coupled system. In this paper we restrict ourselves to the two-dimensional case, since for this case we are able to give complete error analyses for a finite element discretization and a multigrid solver. However, the methodology (see [12]) and all multigrid tools can be extended to the three-dimensional case as well. We allow $\alpha = 0$, which corresponds to the linearization of a *stationary* Navier–Stokes problem in rotation form. We will prove that, under certain reasonable assumptions on the rotation function $\mathbf{w}$, the standard Galerkin finite element discretization method, without any stabilization, is a useful method (see Theorem 3.2 and Remark 3.2). The bounds for the discretization error that are shown to hold are similar to finite element error bounds for scalar linear *reaction-diffusion* problems (as, e.g., in [17], [22]). We consider a multigrid solver for the discrete problem that results from the Galerkin discretization of (1.4) with standard conforming finite elements. It is proved that a multigrid W-cycle method with a canonical prolongation and restriction and a block Richardson smoother is a robust solver for this problem, in the sense that its contraction number (in the Euclidean norm) is bounded by a constant smaller than one independent of all relevant parameters. Although to prove a robust convergence of the multigrid method we need more restrictive assumptions on $\mathbf{w}$, numerical experiments demonstrate good performance of the method, even if such assumptions do not hold. Such a theoretical robustness result is not known for multigrid applied to convection-diffusion problems. Moreover, in the multigrid solver we do not need so-called robust smoothers or matrix-dependent prolongations and restrictions, which are believed to be important for robustness of multigrid applied to convection-diffusion problems. We will show results of numerical experiments that illustrate the stability of the discretization method and the robustness of the multigrid solver. Both in the analysis and the numerical experiments it can be observed that the problem (1.4) resembles a scalar reaction-diffusion problem. Note that from the numerical solution point of view reaction-diffusion equations are believed to be simpler than convection-diffusion equations.

Recently, in [12], a new preconditioning technique for a discretization of the Schur complement operator $\mathbf{S}_{\text{rot}}$ has been introduced, which has good robustness properties with respect to variation in ν and in the mesh size parameter. In this paper we consider only the inner solution operator that appears in the Schur complement operator. Of course, a stabilization may be needed in the outer iterations for (1.3). This subject is addressed in [10], where it is shown that a Petrov–Galerkin-type stabilization method for (1.3) yields optimal error bounds. The possible impact to (1.4) of additional terms resulting from stabilized finite element method for (1.3) is not considered in this paper. Generally, such terms enhance ellipticity of (1.4).

The results in [12], [10], and in the present paper show that for the application of coupled (pressure-velocity) solvers and implicit schemes the rotation form of the Navier–Stokes equations has interesting advantages compared to the convection form. Some numerical experiments with a low order finite element method for rotation form of the incompressible Navier–Stokes equations and comparision with the convection form can be found in [13]. However, relatively little is known about the numerical solution of the Navier–Stokes equations in rotation form, and we believe that this topic deserves further research.

The remainder of the paper is organized as follows. In section 2 notation and assumptions are introduced. Furthermore, continuity and regularity results for the continuous problem are proved. In section 3 the finite element method is treated. We prove discretization error bounds in a problem dependent norm and in the L_2-norm. In section 4 a multigrid solver for the discrete problem is introduced. A convergence analysis is presented that is based on smoothing and approximation properties. In section 5 we show results of a few numerical experiments.

2. Preliminaries and a priori estimates.

Let Ω be a convex polygonal domain in $\mathbb{R}^2$. This assumption on Ω will be needed to obtain sufficient regularity, which strongly simplifies the multigrid convergence theory based on the smoothing and approximation property. However, multigrid methods are known to preserve their typical fast convergence, if this assumption is violated.

By $(\cdot,\cdot)$ and $\|\cdot\|$ we denote the scalar product and the corresponding norm in $L_2(\Omega)^n, n = 1, 2$. The standard norm in the Sobolev space $H^k(\Omega)^2$ is denoted by $\|\cdot\|_k$. For $\mathbf{u} = (u_1, u_2)$, $\mathbf{v} = (v_1, v_2) \in L_2(\Omega)^2$ we have $(\mathbf{u}, \mathbf{v}) = (u_1, v_1) + (u_2, v_2)$. The norm on the space $L_\infty(\Omega)$ is denoted by $\|\cdot\|_\infty$.

For a scalar a and vector $\mathbf{v}$ we define the vector product $a \times \mathbf{v} := (-av_2, av_1)^T$.

We consider the variational formulation of (1.4) in the two-dimensional case: for given $\nu > 0,\ \alpha > 0,\ w \in L_\infty(\Omega),\ \mathbf{f} \in L_2(\Omega)^2$, determine $\mathbf{u} \in \mathbf{U} := H_0^1(\Omega)^2$ such that

$$a(\mathbf{u}, \mathbf{v}) = (\mathbf{f}, \mathbf{v}) \quad \text{for all } \mathbf{v} \in \mathbf{U}, \tag{2.1}$$

where

$$a(\mathbf{u}, \mathbf{v}) = \nu(\nabla \mathbf{u}, \nabla \mathbf{v}) + \alpha(\mathbf{u}, \mathbf{v}) + (w \times \mathbf{u}, \mathbf{v}) \quad \text{for } \mathbf{u}, \mathbf{v} \in \mathbf{U}.$$

Here we use the notation $(\nabla \mathbf{u}, \nabla \mathbf{v}) := \sum_{i=1}^2 (\nabla u_i, \nabla v_i) = \sum_{i,j=1}^2 (\frac{\partial u_i}{\partial x_j}, \frac{\partial v_i}{\partial x_j})$.

Throughout the paper we use C to denote some generic strictly positive constant independent of ν, α, and w .

The definition of the vector product implies $(w \times \mathbf{u}, \mathbf{v}) = -(w \times \mathbf{v}, \mathbf{u})$ for all $\mathbf{u}, \mathbf{v} \in L_2(\Omega)^2$, and thus the bilinear form $a(\cdot, \cdot)$ is elliptic:

$$C\,\nu \|\mathbf{u}\|_1^2 \le a(\mathbf{u}, \mathbf{u}) \quad \text{for all } \mathbf{u} \in \mathbf{U}\ .$$

Using $\|w \times \mathbf{u}\| \le \|w\|_\infty \|\mathbf{u}\|$ we obtain the continuity of the bilinear form:

$$a(\mathbf{u}, \mathbf{v}) \le (\nu + \alpha + \|w\|_\infty)\|\mathbf{u}\|_1 \|\mathbf{v}\|_1 \quad \text{for all } \mathbf{u}, \mathbf{v} \in \mathbf{U}. \tag{2.2}$$

From the Lax–Milgram lemma it follows that the variational problem (2.1) has a unique solution.

For the analysis below we introduce a parameter dependent norm on $\mathbf{U}$:

$$|||\mathbf{u}|||_\tau = \left(\nu\|\nabla \mathbf{u}\|^2 + \alpha\|\mathbf{u}\|^2 + \frac{\tau}{\|w\|_\infty}\|w \times \mathbf{u}\|^2\right)^{\frac{1}{2}}, \quad \tau \ge 0.$$

If $w = 0$, then the third term on the right-hand side is dropped. The constant appearing in the Friedrichs inequality is denoted by C_F:

$$\|\varphi\| \le C_F \|\nabla \varphi\| \quad \text{for all } \varphi \in H_0^1(\Omega).$$

The domain Ω is such that for any $g \in L_2(\Omega)$ the solution of the variational problem

$$\text{find } \varphi \in H_0^1(\Omega) \text{ such that } (\nabla \varphi, \nabla v) = (g, v) \ \text{ for all } \ v \in H_0^1(\Omega) \tag{2.3}$$

is an element of $H^2(\Omega)$ and satisfies the regularity estimate $\|\varphi\|_2 \le C_P \|g\|$.

For the analysis in the remainder of this paper the following three conditions are formulated. We denote $c_w := \text{ess inf}_\Omega |w|$.

(A1) Condition (A1) is satisfied if $\alpha + c_w > 0$ and

$$\eta := \frac{\|w\|_\infty}{\alpha + c_w} \le C.$$

(A2) Condition (A2) is satisfied if

$$w(\mathbf{x}) \ge 0 \text{ a.e. in } \Omega \ \text{ or } w(\mathbf{x}) \le 0 \text{ a.e. in } \Omega.$$

(A3) Condition (A3) is fulfilled if $\nabla w \in L_q(\Omega)^2$ for some $q > 2$ and

$$\|\nabla w\|_{L_q} \le C\, \|w\|_\infty.$$

If w is a finite element function, then C is assumed to be independent of h.

In the analysis below it will be explicitly stated which of these conditions are assumed.

Remark 2.1. (A2) holds, for example, if w stems from the effect of Coriolis forces (cf., e.g., [6]); (A1) holds if w is continuous and does not have any zeros in Ω or if in a time stepping scheme we have lower bound for α: $0 < \alpha_{\min} \leq \alpha$. □

Note that $(|w|\mathbf{u}, \mathbf{u}) = (|w| \times \mathbf{u}, 1 \times \mathbf{u}) \geq 0$, and thus we have for $\mathbf{u} \in L_2(\Omega)^2$

$$c_w \|\mathbf{u}\|^2 \leq (|w| \times \mathbf{u}, 1 \times \mathbf{u}). \tag{2.4}$$

Using $(|w| \times \mathbf{u}, 1 \times \mathbf{u}) \leq \||w| \times \mathbf{u}\|\|1 \times \mathbf{u}\| = \|w \times \mathbf{u}\|\|\mathbf{u}\|$ we get

$$(\alpha + c_w)\|\mathbf{u}\| \leq \|w \times \mathbf{u}\| + \alpha\|\mathbf{u}\|. \tag{2.5}$$

The inequalities (2.4) and (2.5) are used in the analysis below.

2.1. Analysis of the continuous problem. In this section we will derive a regularity result (Theorem 2.1) and a continuity result (Lemma 2.2). In the latter, opposite to the result in (2.2), the problem dependent norm $|||\cdot|||_\tau$ is used. The continuity result is used in the derivation of the discretization error bounds in section 3.

THEOREM 2.1. *For* $\mathbf{f} \in L_2(\Omega)^2$ *let* $\mathbf{u} \in \mathbf{U}$ *be the solution of problem* (2.1). *Then* $\mathbf{u}$ *is an element of* $H^2(\Omega)^2$ *and the estimates*

$$\nu\|\nabla\mathbf{u}\|^2 + \alpha\|\mathbf{u}\|^2 \leq c(\nu, \alpha)\|\mathbf{f}\|^2 \ , \tag{2.6}$$

$$\nu^2\|\mathbf{u}\|_2^2 + C_P^2\|w \times \mathbf{u}\|^2 \leq 2C_P^2\big(4 + 2c(\nu, \alpha)^2\|w\|_\infty^2\big)\|\mathbf{f}\|^2 \tag{2.7}$$

hold, with $c(\nu, \alpha) = \frac{C_F^2}{\nu + C_F^2\alpha}$. *If conditions* (A1) *and* (A3) *are satisfied, then*

$$\nu^2\|\mathbf{u}\|_2^2 + \nu(\|w\|_\infty + \alpha)\|\nabla\mathbf{u}\|^2 + \alpha^2\|\mathbf{u}\|^2 + \|w \times \mathbf{u}\|^2 \leq C\|\mathbf{f}\|^2 \tag{2.8}$$

with a constant C *independent of* $\mathbf{f}$, ν, α, *and* w.

Proof. Define $\tilde{\mathbf{f}} = \mathbf{f} - w \times \mathbf{u} - \alpha\mathbf{u}$. Note that $\tilde{\mathbf{f}} \in L_2(\Omega)^2$ and $(\nabla\mathbf{u}, \nabla\mathbf{v}) = -\frac{1}{\nu}(\tilde{\mathbf{f}}, \mathbf{v})$ for all $\mathbf{v} \in \mathbf{U}$. Hence, due to the regularity result for the Poisson equation (2.3), we have $\mathbf{u} \in H^2(\Omega)^2$ and

$$\|\mathbf{u}\|_2 \leq \frac{C_P}{\nu}\|\tilde{\mathbf{f}}\| \leq \frac{C_P}{\nu}(\|\mathbf{f}\| + \|w \times \mathbf{u}\| + \alpha\|\mathbf{u}\|). \tag{2.9}$$

Note that $\|\mathbf{u}\|^2 = c(\nu, \alpha)(\nu C_F^{-2} + \alpha)\|\mathbf{u}\|^2 \leq c(\nu, \alpha)(\nu\|\nabla\mathbf{u}\|^2 + \alpha\|\mathbf{u}\|^2)$. Using this and taking $\mathbf{v} = \mathbf{u}$ in (2.1) we get

$$\nu\|\nabla\mathbf{u}\|^2 + \alpha\|\mathbf{u}\|^2 \leq \|\mathbf{f}\|\|\mathbf{u}\| \leq \|\mathbf{f}\|c(\nu, \alpha)^{\frac{1}{2}}(\nu\|\nabla\mathbf{u}\|^2 + \alpha\|\mathbf{u}\|^2)^{\frac{1}{2}}, \tag{2.10}$$

and thus the result in (2.6) holds. We also have, using (2.6),

$$\|w \times \mathbf{u}\|^2 \leq \|w\|_\infty^2\|\mathbf{u}\|^2 \leq c(\nu, \alpha)\|w\|_\infty^2(\nu\|\nabla\mathbf{u}\|^2 + \alpha\|\mathbf{u}\|^2) \leq c(\nu, \alpha)^2\|w\|_\infty^2\|\mathbf{f}\|^2. \tag{2.11}$$

Combining this estimate with (2.9), and noting that $\alpha\|\mathbf{u}\| \leq \|\mathbf{f}\|$, yields

$$\begin{aligned}\nu^2\|\mathbf{u}\|_2^2 + C_P^2\|w \times \mathbf{u}\|^2 &\leq C_P^2(\|\mathbf{f}\| + c(\nu, \alpha)\|w\|_\infty\|\mathbf{f}\| + \|\mathbf{f}\|)^2 + C_P^2 c(\nu, \alpha)^2\|w\|_\infty^2\|\mathbf{f}\|^2 \\ &= C_P^2((2 + c(\nu, \alpha)\|w\|_\infty)^2 + c(\nu, \alpha)^2\|w\|_\infty^2)\|\mathbf{f}\|^2 \\ &\leq 2C_P^2(3 + 2c(\nu, \alpha)^2\|w\|_\infty^2)\|\mathbf{f}\|^2,\end{aligned}$$

and thus the estimate (2.7) is proved.

Now assume the conditions (A1) and (A3) to be valid. Since $\mathbf{f} \in L_2(\Omega)^2$ and $\mathbf{u} \in H^2(\Omega)^2$, (1.4) is satisfied in a strong sense, and thus $\|-\nu\Delta\mathbf{u}+\alpha\mathbf{u}+w\times\mathbf{u}\| = \|\mathbf{f}\|$ holds. Taking the square of this identity and noting that $(\mathbf{u}, w \times \mathbf{u}) = 0$ results in

$$\nu^2\|\Delta\mathbf{u}\|^2 + 2\nu\alpha\|\nabla\mathbf{u}\|^2 + \alpha^2\|\mathbf{u}\|^2 + 2\nu(\nabla\mathbf{u}, \nabla(w\times\mathbf{u})) + \|w\times\mathbf{u}\|^2 = \|\mathbf{f}\|^2. \tag{2.12}$$

A simple computation yields $(\nabla\mathbf{u}, \nabla(w \times \mathbf{u})) = -(\nabla u_1, u_2\nabla w) + (\nabla u_2, u_1\nabla w)$ and

$$|(\nabla\mathbf{u}, \nabla(w\times\mathbf{u}))| \le \|\nabla\mathbf{u}\|(\|u_1\nabla w\|^2 + \|u_2\nabla w\|^2)^{\frac{1}{2}}. \tag{2.13}$$

Take q as in (A3) and define $\tilde{q} = \frac{1}{2}q$. The Hölder inequality with $\frac{1}{p} + \frac{1}{\tilde{q}} = 1$ and the injection $H_1(\Omega) \hookrightarrow L_{2p}(\Omega)$ yields, for $i = 1, 2$,

$$\begin{aligned} \|u_i\nabla w\| = (u_i^2, \nabla w\cdot\nabla w)^{\frac{1}{2}} &\le \|u_i\|_{L_{2p}}\|\nabla w\cdot\nabla w\|_{L_{\tilde{q}}}^{\frac{1}{2}} \\ &\le C\|\nabla u_i\|\|\nabla w\|_{L_q} \le C\|\nabla u_i\|\|w\|_\infty. \end{aligned} \tag{2.14}$$

In the last inequality in (2.14) we used (A3). The combination of (2.13) and (2.14) yields

$$2\nu|(\nabla\mathbf{u}, \nabla(w\times\mathbf{u}))| \le \bar{c}\,\nu\|w\|_\infty\,\|\nabla\mathbf{u}\|^2.$$

From this result and (2.12) we obtain

$$\nu^2\|\Delta\mathbf{u}\|^2 + 2\nu\alpha\|\nabla\mathbf{u}\|^2 + \alpha^2\|\mathbf{u}\|^2 + \|w\times\mathbf{u}\|^2 \le \|\mathbf{f}\|^2 + \bar{c}\,\nu\|w\|_\infty\,\|\nabla\mathbf{u}\|^2. \tag{2.15}$$

From (2.1) and (2.5) it follows that, for $\delta > 0$,

$$\begin{aligned} \nu\|\nabla\mathbf{u}\|^2 \le \|\mathbf{f}\|\,\|\mathbf{u}\| &= \frac{1}{\sqrt{\delta}(\alpha+c_w)}\|\mathbf{f}\|\sqrt{\delta}(\alpha+c_w)\|\mathbf{u}\| \\ &\le \frac{\|\mathbf{f}\|^2}{2\delta(\alpha+c_w)^2} + \delta(\alpha^2\|\mathbf{u}\|^2 + \|w\times\mathbf{u}\|^2). \end{aligned} \tag{2.16}$$

If we set $\delta = (4\,\bar{c}\,\|w\|_\infty)^{-1}$ and multiply (2.16) with $\frac{1}{2\delta}$ we obtain

$$2\bar{c}\nu\|w\|_\infty\|\nabla\mathbf{u}\|^2 \le \bar{c}^2\frac{\|w\|_\infty^2}{(\alpha+c_w)^2}\|\mathbf{f}\|^2 + \frac{1}{2}\alpha^2\|\mathbf{u}\|^2 + \frac{1}{2}\|w\times\mathbf{u}\|^2\ .$$

Adding this to (2.15) yields

$$\begin{aligned} &\nu^2\|\Delta\mathbf{u}\|^2 + \nu(\bar{c}\|w\|_\infty + 2\alpha)\|\nabla\mathbf{u}\|^2 + \alpha^2\|\mathbf{u}\|^2 + \|w\times\mathbf{u}\|^2 \\ &\quad\le \Big(1 + \bar{c}^2\frac{\|w\|_\infty^2}{(\alpha+c_w)^2}\Big)\|\mathbf{f}\|^2 + \frac{1}{2}\alpha^2\|\mathbf{u}\|^2 + \frac{1}{2}\|w\times\mathbf{u}\|^2. \end{aligned}$$

Using assumption (A1), i.e., $\frac{\|w\|_\infty^2}{(\alpha+c_w)^2} = \eta^2 \le C$ and $\|\mathbf{u}\|_2 \le C_P\|\Delta\mathbf{u}\|$, the result in (2.8) follows. □

Note that in (2.6) and (2.7) with $\alpha = 0$ we have regularity estimates of the form $\|\mathbf{u}\|_1 = O(\nu^{-1})$ and $\|\mathbf{u}\|_2 = O(\nu^{-2})$, which show a similar behavior as regularity results for convection-diffusion problems of the form $-\nu\Delta u + \mathbf{a}\cdot\nabla u = f$ (cf. [16]). The result in (2.8), which holds if conditions (A1) and (A3) are satisfied, yields regularity estimates of the form $\|\mathbf{u}\|_1 = O(\nu^{-1/2})$ and $\|\mathbf{u}\|_2 = O(\nu^{-1})$. These bounds show a behavior that is typical for the solution of reaction-diffusion problems of the form $-\nu\Delta u + bu = f$ if $b > 0$ (cf. [17]). In section 4.2 the regularity result (2.8) will be used in the convergence analysis of the multigrid method.

Lemma 2.2. *Take $\tau > 0$. The following holds:*

$$a(\mathbf{v},\mathbf{u}) \le C_\tau\, |||\mathbf{v}|||_\tau \Big(\nu\|\nabla\mathbf{u}\|^2 + (\alpha + \|w\|_\infty)\|\mathbf{u}\|^2\Big)^{\frac{1}{2}} \quad \textit{for all } \mathbf{v},\mathbf{u} \in \mathbf{U}. \tag{2.17}$$

If condition (A1) *is satisfied, then*

$$a(\mathbf{v},\mathbf{u}) \le C_\tau\, |||\mathbf{v}|||_\tau\, |||\mathbf{u}|||_\tau \quad \textit{for all } \mathbf{v},\mathbf{u} \in \mathbf{U}. \tag{2.18}$$

The constants C_τ may depend on τ.

Proof. For $\mathbf{v},\mathbf{u} \in \mathbf{U}$ we have

$$\begin{aligned} a(\mathbf{v},\mathbf{u}) &= \nu(\nabla\mathbf{v},\nabla\mathbf{u}) + \alpha(\mathbf{v},\mathbf{u}) + (w \times \mathbf{v},\mathbf{u}) \\ &\le \nu\|\nabla\mathbf{v}\|\|\nabla\mathbf{u}\| + \alpha\|\mathbf{v}\|\|\mathbf{u}\| + \|w \times \mathbf{v}\|\|\mathbf{u}\|. \end{aligned} \tag{2.19}$$

We define $\kappa := \tau\|w\|_\infty^{-1}$. If we use $\|w \times \mathbf{v}\|\|\mathbf{u}\| = (\kappa^{\frac{1}{2}}\|w \times \mathbf{v}\|)(\kappa^{-\frac{1}{2}}\|\mathbf{u}\|)$ and apply the Cauchy–Schwarz inequality in (2.19) we obtain

$$\begin{aligned} a(\mathbf{v},\mathbf{u}) &\le \Big(\nu\|\nabla\mathbf{v}\|^2 + \alpha\|\mathbf{v}\|^2 + \kappa\|w \times \mathbf{v}\|^2\Big)^{\frac{1}{2}} \Big(\nu\|\nabla\mathbf{u}\|^2 + \alpha\|\mathbf{u}\|^2 + \kappa^{-1}\|\mathbf{u}\|^2\Big)^{\frac{1}{2}} \\ &\le C_\tau|||\mathbf{v}|||_\tau\Big(\nu\|\nabla\mathbf{u}\|^2 + (\alpha + \|w\|_\infty)\|\mathbf{u}\|^2\Big)^{\frac{1}{2}}, \end{aligned} \tag{2.20}$$

and thus the result in (2.17) holds. If condition (A1) is satisfied we get, using (2.5),

$$\begin{aligned} \|w \times \mathbf{v}\|\,\|\mathbf{u}\| &\le \|w \times \mathbf{v}\|\frac{1}{\alpha + c_w}(\alpha\|\mathbf{u}\| + \|w \times \mathbf{u}\|) \\ &\le \kappa^{\frac{1}{2}}\|w \times \mathbf{v}\|\frac{\kappa^{-\frac{1}{2}}(\alpha^{\frac{1}{2}} + \kappa^{-\frac{1}{2}})}{\alpha + c_w}(\alpha^{\frac{1}{2}}\|\mathbf{u}\| + \kappa^{\frac{1}{2}}\|w \times \mathbf{u}\|) \\ &\le C_\tau(\kappa^{\frac{1}{2}}\|w \times \mathbf{v}\|)(\alpha\|\mathbf{u}\|^2 + \kappa\|w \times \mathbf{u}\|^2)^{\frac{1}{2}}. \end{aligned} \tag{2.21}$$

In the last inequality in (2.21) we used condition (A1):

$$\frac{\kappa^{-\frac{1}{2}}(\alpha^{\frac{1}{2}} + \kappa^{-\frac{1}{2}})}{\alpha + c_w} \le \frac{\frac{3}{2}\kappa^{-1} + \frac{1}{2}\alpha}{\alpha + c_w} \le \frac{3}{2\tau}\eta + \frac{\alpha}{2(\alpha + c_w)} \le C_\tau.$$

From the results in (2.19), (2.21), and the Cauchy–Schwarz inequality, we obtain (2.18). □

3. Finite element method.

In this section we apply a standard finite element method to the problem (2.1) and derive bounds for the discretization error.

Let $(\mathcal{T}_h)$ be a quasi-uniform family of triangulations of Ω, with mesh size parameter h, and $\mathbf{U}_h \subset \mathbf{U}$ be a finite element subspace of $\mathbf{U}$, consisting of piecewise polynomials of degree $k \in \mathbb{N}$. The finite element Galerkin discretization of the problem (2.1) is as follows: Find $\mathbf{u}_h \in \mathbf{U}_h$ such that

$$a(\mathbf{u}_h,\mathbf{v}_h) = (\mathbf{f},\mathbf{v}_h) \quad \text{for all } \mathbf{v}_h \in \mathbf{U}_h. \tag{3.1}$$

To measure the effect of different terms in (1.4) we introduce mesh numbers:[1]

$$\mathrm{Ek}_h = \frac{\nu}{\|w\|_\infty h^2}, \qquad \mathrm{D}_h = \frac{\alpha h^2}{\nu}.$$

First we prove the stability of $a(\mathbf{u},\mathbf{v})$ on $\mathbf{U}_h$. Below we use the inverse inequality

$$\|\nabla \mathbf{v}_h\| < \mu_u h^{-1}\|\mathbf{v}_h\| \quad \text{for all } \ \mathbf{v}_h \in \mathbf{U}_h.$$

The L_2-orthogonal projection $\mathrm{P}_h : L_2(\Omega)^2 \to \mathbf{U}_h$ is defined by

$$(\mathrm{P}_h\mathbf{u},\mathbf{v}_h) = (\mathbf{u},\mathbf{v}_h) \quad \text{for all } \mathbf{v}_h \in \mathbf{U}_h. \tag{3.2}$$

We will assume the following approximation property of the spaces $\mathbf{U}_h$ (cf., e.g., [5]): their exists interpolation operator $I_h : \mathbf{U} \to \mathbf{U}_h$ such that

$$\|\mathbf{u} - I_h\mathbf{u}\| \le Ch^m\|\mathbf{u}\|_m, \quad m = 0,1,2 \quad \text{for } \mathbf{u} \in \mathbf{U}\cap H^m(\Omega)^2, \tag{3.3}$$

$$\|\mathbf{u} - I_h\mathbf{u}\|_1 \le Ch^{m-1}\|\mathbf{u}\|_m, \quad m = 1,2 \quad \text{for } \mathbf{u} \in \mathbf{U}\cap H^m(\Omega)^2. \tag{3.4}$$

In (3.3) we use the notation $H^0(\Omega)^2 := L_2(\Omega)^2$ and $\|\cdot\|_0 := \|\cdot\|$.

Lemma 3.1. *Assume that conditions* (A1) *and* (A2) *are fulfilled. If* $\mathrm{Ek}_h > 1$ *and* $\mathrm{D}_h < 1$, *condition* (A3) *is also assumed. Then there exists some* $\tau \in (0,1]$ *such that*

$$\inf_{\mathbf{u}_h\in\mathbf{U}_h}\ \sup_{\mathbf{v}_h\in\mathbf{U}_h} \frac{a(\mathbf{u}_h,\mathbf{v}_h)}{|||\mathbf{u}_h|||_\tau\, |||\mathbf{v}_h|||_\tau} \ge C > 0. \tag{3.5}$$

Proof. Take a fixed $\mathbf{u}_h \in \mathbf{U}_h$. Note that

$$\begin{aligned}(w\times\mathbf{u}_h, \mathrm{P}_h(w\times\mathbf{u}_h)) &= (\mathrm{P}_h(w\times\mathbf{u}_h), \mathrm{P}_h(w\times\mathbf{u}_h)),\\ (\mathbf{u}_h, \mathrm{P}_h(w\times\mathbf{u}_h)) &= 0.\end{aligned}$$

Using (2.4) and condition (A2) it follows that

$$\begin{aligned}c_w\|\mathbf{u}_h\|^2 &\le (|w|\times\mathbf{u}_h, 1\times\mathbf{u}_h) = (\mathrm{P}_h(|w|\times\mathbf{u}_h), 1\times\mathbf{u}_h)\\ &= (|\mathrm{P}_h(w\times\mathbf{u}_h)|, 1\times\mathbf{u}_h) \le \|\mathrm{P}_h(w\times\mathbf{u}_h)\|\|\mathbf{u}_h\|,\end{aligned}$$

and thus

$$(\alpha + c_w)\|\mathbf{u}_h\| \le \alpha\|\mathbf{u}_h\| + \|\mathrm{P}_h(w\times\mathbf{u}_h)\|. \tag{3.6}$$

We take

$$\tau = \min\{1, \mu_u^{-2}, \tilde{c}^{-1}\}, \tag{3.7}$$

where $\tilde{c}$ is a constant (independent of all parameters) that will occur in the proof. Let $\kappa := \tau\|w\|_\infty^{-1}$. Using (3.6) we obtain

$$\begin{aligned}\alpha\|\mathbf{u}_h\|^2 + \kappa\|w\times\mathbf{u}_h\|^2 &\le (\alpha + \kappa\|w\|_\infty^2)\|\mathbf{u}_h\|^2\\ &\le \frac{2(\alpha+\kappa\|w\|_\infty^2)}{(\alpha+c_w)^2}(\alpha^2\|\mathbf{u}_h\|^2 + \|\mathrm{P}_h(w\times\mathbf{u}_h)\|^2)\\ &\le \frac{2(\alpha+\kappa\|w\|_\infty^2)(\alpha+\kappa^{-1})}{(\alpha+c_w)^2}(\alpha\|\mathbf{u}_h\|^2 + \kappa\|\mathrm{P}_h(w\times\mathbf{u}_h)\|^2).\end{aligned}$$

[1] The abbreviation and definition of Ek is chosen to be consistent with the definition of the Ekman number in the theory of rotating flows. However, the latter is only a particular case (w = const).

Note that $\tau^{-1}+\tau \le \max\{1,\mu_u^2,\tilde{c}\}+1 \le C$ and thus, using condition (A1),

$$\frac{(\alpha+\kappa\|w\|_\infty^2)(\alpha+\kappa^{-1})}{(\alpha+c_w)^2} = \frac{\alpha^2+(\tau^{-1}+\tau)\alpha\|w\|_\infty+\|w\|_\infty^2}{(\alpha+c_w)^2} \le C\frac{\alpha^2+\|w\|_\infty^2}{(\alpha+c_w)^2} \le C(1+\eta^2) \le C.$$

Hence,

$$\alpha\|\mathbf{u}_h\|^2+\kappa\|w\times\mathbf{u}_h\|^2 \le C(\alpha\|\mathbf{u}_h\|^2+\kappa\|\mathrm{P}_h(w\times\mathbf{u}_h)\|^2). \tag{3.8}$$

To prove (3.5) we choose $\mathbf{v}_h=\mathbf{u}_h+\kappa\mathrm{P}_h(w\times\mathbf{u}_h)$. Then

$$\begin{aligned} a(\mathbf{u}_h,\mathbf{v}_h) &= \nu\|\nabla\mathbf{u}_h\|^2+\alpha\|\mathbf{u}_h\|^2+\nu\kappa(\nabla\mathbf{u}_h,\nabla\mathrm{P}_h(w\times\mathbf{u}_h))+\kappa\|\mathrm{P}_h(w\times\mathbf{u}_h)\|^2 \\ &\ge \nu\|\nabla\mathbf{u}_h\|^2+\alpha\|\mathbf{u}_h\|^2-\nu\kappa\|\nabla\mathbf{u}_h\|\,\|\nabla\mathrm{P}_h(w\times\mathbf{u}_h)\|+\kappa\|\mathrm{P}_h(w\times\mathbf{u}_h)\|^2. \end{aligned} \tag{3.9}$$

For the estimation of the term $\|\nabla\mathrm{P}_h(w\times\mathbf{u}_h)\|$ we distinguish three cases: $\mathrm{Ek}_h\le 1$ (case 1), $\mathrm{D}_h\ge 1$ (case 2), and $\mathrm{Ek}_h>1$ and $\mathrm{D}_h<1$ (case 3).
In case 1 we have

$$\begin{aligned} (\nu\kappa)^{\frac12}\|\nabla\mathrm{P}_h(w\times\mathbf{u}_h)\| &\le \left(\frac{\nu\tau\mu_u^2}{\|w\|_\infty h^2}\right)^{\frac12}\|\mathrm{P}_h(w\times\mathbf{u}_h)\| \\ &= (\mathrm{Ek}_h\tau\mu_u^2)^{\frac12}\|\mathrm{P}_h(w\times\mathbf{u}_h)\| \le \|\mathrm{P}_h(w\times\mathbf{u}_h)\|. \end{aligned} \tag{3.10}$$

Using this in (3.9) and applying the Cauchy–Schwarz inequality, we get

$$a(\mathbf{u}_h,\mathbf{v}_h) \ge \frac12\nu\|\nabla\mathbf{u}_h\|^2+\alpha\|\mathbf{u}_h\|^2+\frac12\kappa\|\mathrm{P}_h(w\times\mathbf{u}_h)\|^2. \tag{3.11}$$

In case 2 we have

$$\begin{aligned} \nu^{\frac12}\kappa\|\nabla\mathrm{P}_h(w\times\mathbf{u}_h)\| &\le \nu^{\frac12}\kappa\mu_u h^{-1}\|w\|_\infty\|\mathbf{u}\| = \tau\mu_u\mathrm{D}_h^{-\frac12}\alpha^{\frac12}\|\mathbf{u}\| \\ &\le \tau^{\frac12}\mu_u\mathrm{D}_h^{-\frac12}\alpha^{\frac12}\|\mathbf{u}\| \le \alpha^{\frac12}\|\mathbf{u}\|. \end{aligned} \tag{3.12}$$

Using this in (3.9) and applying the Cauchy–Schwarz inequality, we get

$$a(\mathbf{u}_h,\mathbf{v}_h) \ge \frac12\nu\|\nabla\mathbf{u}_h\|^2+\frac12\alpha\|\mathbf{u}_h\|^2+\kappa\|\mathrm{P}_h(w\times\mathbf{u}_h)\|^2. \tag{3.13}$$

For case 3 first note that, using condition (A3) and the result in (2.14) it follows that

$$\begin{aligned} \|\nabla(w\times\mathbf{u}_h)\|^2 &= \sum_{i=1}^2 \|(u_h)_i\nabla w\|^2+\|w\nabla(u_h)_i\|^2+2((u_h)_i\nabla w, w\nabla(u_h)_i) \\ &\le 2\sum_{i=1}^2 \|(u_h)_i\nabla w\|^2+\|w\nabla(u_h)_i\|^2 \le c_1\|w\|_\infty^2\|\nabla\mathbf{u}_h\|^2. \end{aligned}$$

We use that the L_2-orthogonal projection is bounded in the H^1-norm (cf. [2]):

$$\|\mathrm{P}_h\mathbf{u}\|_1 \le c_2\|\mathbf{u}\|_1 \qquad \text{for } \mathbf{u}\in\mathbf{U}.$$

For the constant $\tilde{c}$ in (3.7) we take $\tilde{c} = 2c_2\sqrt{c_1}$ and then obtain

$$\kappa\|\nabla \mathrm{P}_h(w \times \mathbf{u}_h)\| \le c_2\kappa\|\nabla(w \times \mathbf{u}_h)\| \le c_2\sqrt{c_1}\kappa\|w\|_\infty\|\nabla\mathbf{u}_h\| \le \frac{1}{2}\|\nabla\mathbf{u}_h\|. \tag{3.14}$$

Using this in (3.9) results in

$$a(\mathbf{u}_h, \mathbf{v}_h) \ge \frac{1}{2}\nu\|\nabla\mathbf{u}_h\|^2 + \alpha\|\mathbf{u}_h\|^2 + \kappa\|\mathrm{P}_h(w \times \mathbf{u}_h)\|^2. \tag{3.15}$$

The combination of (3.11), (3.13), (3.15) with (3.8) proves that

$$a(\mathbf{u}_h, \mathbf{v}_h) \ge C|||\mathbf{u}_h|||_\tau^2 \tag{3.16}$$

holds. The results in (3.10), (3.12), and (3.14) imply

$$\nu\kappa^2\|\nabla\mathrm{P}_h(w \times \mathbf{u}_h)\|^2 \le |||\mathbf{u}_h|||_\tau^2 \ .$$

Using this it follows that

$$\begin{aligned}
|||\mathbf{v}_h|||_\tau^2 &= \nu\|\nabla(\mathbf{u}_h + \kappa\mathrm{P}_h(w \times \mathbf{u}_h))\|^2 + \alpha\|\mathbf{u}_h + \kappa\mathrm{P}_h(w \times \mathbf{u}_h)\|^2 \\
&\quad + \kappa\|\mathrm{P}_h(w \times \mathbf{u}_h + \kappa w \times \mathrm{P}_h(w \times \mathbf{u}_h))\|^2 \\
&\le 2(\nu\|\nabla\mathbf{u}_h\|^2 + \nu\kappa^2\|\nabla\mathrm{P}_h(w \times \mathbf{u}_h)\|^2) + \alpha\|\mathbf{u}_h\|^2 + \kappa^2\alpha\|\mathrm{P}_h(w \times \mathbf{u}_h)\|^2 \\
&\quad + 2\kappa(\|\mathrm{P}_h(w \times \mathbf{u}_h)\|^2 + \kappa^2\|\mathrm{P}_h(w \times \mathrm{P}_h(w \times \mathbf{u}_h))\|^2) \\
&\le 2\nu\|\nabla\mathbf{u}_h\|^2 + 2|||\mathbf{u}_h|||_\tau^2 + \alpha(1+\tau^2)\|\mathbf{u}_h\|^2 + 2\kappa(1+\tau^2)\|\mathrm{P}_h(w \times \mathbf{u}_h)\|^2 \\
&\le 2\nu\|\nabla\mathbf{u}_h\|^2 + 2\alpha\|\mathbf{u}_h\|^2 + 4\kappa\|\mathrm{P}_h(w \times \mathbf{u}_h)\|^2 + 2|||\mathbf{u}_h|||_\tau^2 \\
&\le 6|||\mathbf{u}_h|||_\tau^2.
\end{aligned}$$

The combination of the latter estimate and (3.16) completes the proof. □

Remark 3.1. Note that τ in Lemma 3.1 does not depend on ν, α, or w.

Remark 3.2. Using the mesh-dependent norm

$$|||\mathbf{u}|||_{\tau,h} = \left(\nu\|\nabla\mathbf{u}\|^2 + \alpha\|\mathbf{u}\|^2 + \frac{\tau}{\|w\|_\infty}\|\mathrm{P}_h(w \times \mathbf{u})\|^2\right)^{\frac{1}{2}} \tag{3.17}$$

the stability of $a(\cdot,\cdot)$ on $\mathbf{U}_h$ can be proved without assumption (A1) and (A2) on w, since estimate (3.8) is not needed. Moreover, continuity of $a(\cdot,\cdot)$ on $\mathbf{U}_h \times \mathbf{U}$ in the mesh-dependent norm (3.17) can be proved without the assumptions (A1), (A2). This then results in satisfactory discretization error bounds in the norm $|||\cdot|||_{\tau,h}$. (See the treatment of the Oseen problem in [10].) However, for a certain duality argument in the proof of the approximation property in the multigrid convergence analysis (see Theorem 3.3 and section 4) we need the continuity of $a(\cdot,\cdot)$ on $\mathbf{U} \times \mathbf{U}$, and then the mesh-dependent norm becomes inconvenient.

We now derive discretization error bounds for the finite element method using standard arguments based on Galerkin orthogonality, stability, continuity, and approximation properties of the finite element spaces.

THEOREM 3.2. *Let* $\mathbf{u}$ *and* $\mathbf{u}_h$ *be the solution of* (2.1) *and* (3.1), *respectively. Let the assumptions of Lemma* 3.1 *be fulfilled and take* $\tau \in (0,1]$ *as in Lemma* 3.1. *Then the following inequalities hold:*

$$|||\mathbf{u} - \mathbf{u}_h|||_\tau \le C_\tau\, h^j(\nu^{\frac{1}{2}}\|\mathbf{u}\|_{j+1} + (\alpha^{\frac{1}{2}} + \|w\|_\infty^{\frac{1}{2}})\|\mathbf{u}\|_j), \quad j = 0, 1, \tag{3.18}$$

$$|||\mathbf{u} - \mathbf{u}_h|||_\tau \le C_\tau\, h(\nu^{\frac{1}{2}} + (\alpha^{\frac{1}{2}} + \|w\|_\infty^{\frac{1}{2}})h)\|\mathbf{u}\|_2. \tag{3.19}$$

The constants C_τ *are independent of* ν, α, w, $\mathbf{u}$, *and* h *but may depend on* τ.

Proof. Let $\hat{\mathbf{u}}_h$ be an arbitrary function in $\mathbf{U}_h$. Take τ as in Lemma 3.1. Then there exists $\mathbf{v}_h \in \mathbf{U}_h$ such that

$$C|||\mathbf{u}_h - \hat{\mathbf{u}}_h|||_\tau \, |||\mathbf{v}_h|||_\tau \le a(\mathbf{u}_h - \hat{\mathbf{u}}_h, \mathbf{v}_h).$$

Using Galerkin orthogonality and the continuity result in (2.18) we obtain

$$a(\mathbf{u}_h - \hat{\mathbf{u}}_h, \mathbf{v}_h) = a(\mathbf{u} - \hat{\mathbf{u}}_h, \mathbf{v}_h) \le C_\tau |||\mathbf{u} - \hat{\mathbf{u}}_h|||_\tau |||\mathbf{v}_h|||_\tau.$$

Hence,

$$|||\mathbf{u}_h - \hat{\mathbf{u}}_h|||_\tau \le C_\tau |||\mathbf{u} - \hat{\mathbf{u}}_h|||_\tau \tag{3.20}$$

holds. From the triangle inequality and (3.20) it follows that

(3.21)
$$\begin{aligned} |||\mathbf{u} - \mathbf{u}_h|||_\tau^2 &\le C_\tau |||\mathbf{u} - \hat{\mathbf{u}}_h|||_\tau^2 \\ &\le C_\tau \left(\nu \|\nabla(\mathbf{u} - \hat{\mathbf{u}}_h)\|^2 + \alpha\|\mathbf{u} - \hat{\mathbf{u}}_h\|^2 + \frac{\tau}{\|w\|_\infty}\|w \times (\mathbf{u} - \hat{\mathbf{u}}_h)\|^2 \right) \\ &\le C_\tau \left(\nu\|\mathbf{u} - \hat{\mathbf{u}}_h\|_1^2 + (\alpha + \tau\|w\|_\infty)\|\mathbf{u} - \hat{\mathbf{u}}_h\|^2 \right). \end{aligned}$$

According to (3.3) and (3.4) $\hat{\mathbf{u}}_h = I_h\mathbf{u}$ can be taken such that

$$\|\mathbf{u} - \hat{\mathbf{u}}_h\|_1^2 \le Ch^{2j}\|\mathbf{u}\|_{j+1}^2, \quad \|\mathbf{u} - \hat{\mathbf{u}}_h\|^2 \le Ch^{2j}\|\mathbf{u}\|_j^2, \quad j = 0, 1.$$

Using this in (3.21) proves the result in (3.18). If we use the inequalities

$$\|\mathbf{u} - \hat{\mathbf{u}}_h\|_1^2 \le Ch^2\|\mathbf{u}\|_2^2, \quad \|\mathbf{u} - \hat{\mathbf{u}}_h\|^2 \le Ch^4\|\mathbf{u}\|_2^2,$$

in (3.21) we get the result in (3.19). □

Note that $\|w\|_\infty$ occurs in the estimates (3.18)–(3.19) in a similar way as α, which measures the *reaction.*

We now prove a discretization error bound in the L_2-norm. This result will play an important role in the convergence analysis of the multigrid method.

THEOREM 3.3. *Assume that the conditions* (A1), (A2)*, and* (A3) *are fulfilled. For* $\mathbf{f} \in L_2(\Omega)^2$ *let* $\mathbf{u}$ *and* $\mathbf{u}_h$ *be the solutions of* (2.1) *and* (3.1)*, respectively. Then*

$$\|\mathbf{u} - \mathbf{u}_h\| \le C \min\left\{ \frac{h^2}{\nu}, \frac{1}{\alpha + \|w\|_\infty} \right\} \|\mathbf{f}\| \tag{3.22}$$

holds with a constant C independent of $\nu, \alpha, w, h,$ *and* $\mathbf{f}$.

Proof. Take $\mathbf{f} \in L_2(\Omega)^2$ and let $\mathbf{u}$, $\mathbf{u}_h$ be the solutions of (2.1) and (3.1), respectively. From (3.18) and the regularity estimate (2.8) it follows that

$$\begin{aligned} |||\mathbf{u} - \mathbf{u}_h|||_\tau &\le C_\tau h \left(\nu^{\frac{1}{2}}\|\mathbf{u}\|_2 + (\alpha^{\frac{1}{2}} + \|w\|_\infty^{\frac{1}{2}})\|\mathbf{u}\|_1 \right) \\ &\le C_\tau \frac{h}{\sqrt{\nu}} \left(\nu^2\|\mathbf{u}\|_2^2 + \nu(\alpha + \|w\|_\infty)\|\nabla\mathbf{u}\|^2 \right)^{\frac{1}{2}} \le C_\tau \frac{h}{\sqrt{\nu}}\|\mathbf{f}\|. \end{aligned} \tag{3.23}$$

We now apply a duality argument. For this we introduce the adjoint bilinear form

$$a^*(\mathbf{u}, \mathbf{v}) = \nu(\nabla\mathbf{u}, \nabla\mathbf{v}) + \alpha(\mathbf{u}, \mathbf{v}) - (w \times \mathbf{u}, \mathbf{v}) \quad \text{for } \mathbf{u}, \mathbf{v} \in \mathbf{U},$$

and the adjoint problem

$$\text{find} \quad \tilde{\mathbf{u}} \in \mathbf{U} \;\; \text{such that} \;\; a^*(\tilde{\mathbf{u}}, \mathbf{v}) = (\tilde{\mathbf{f}}, \mathbf{v}) \qquad \text{for all } \mathbf{v} \in \mathbf{U},$$

with $\tilde{\mathbf{f}} := \mathbf{u} - \mathbf{u}_h \in \mathbf{U} \subset L_2(\Omega)^2$. Let $\tilde{\mathbf{u}}_h \in \mathbf{U}_h$ be the discrete solution of the adjoint problem, i.e., $a^*(\tilde{\mathbf{u}}_h, \mathbf{v}_h) = (\tilde{\mathbf{f}}, \mathbf{v}_h)$ for all $\mathbf{v}_h \in \mathbf{U}_h$. Note that $a^*(\cdot,\cdot)$ equals $a(\cdot,\cdot)$ if, in $a(\cdot,\cdot)$, we replace w by $-w$. The results in Lemma 3.1 and Theorem 3.2 do not depend on $\operatorname{sign}(w)$ and thus hold for the adjoint problem, too. Moreover, since the choice of τ in Lemma 3.1 does not depend on w (cf. Remark 3.1), the estimate (3.23) holds for the original and the adjoint problem with the same τ value. Using this discretization error bound for the original and adjoint problem and the continuity result of Lemma 2.2 we obtain

$$\begin{aligned}\|\mathbf{u} - \mathbf{u}_h\|^2 &= (\tilde{\mathbf{f}}, \tilde{\mathbf{f}}) = a^*(\tilde{\mathbf{u}}, \tilde{\mathbf{f}}) = a(\tilde{\mathbf{f}}, \tilde{\mathbf{u}}) = a(\mathbf{u} - \mathbf{u}_h, \tilde{\mathbf{u}}) = a(\mathbf{u} - \mathbf{u}_h, \tilde{\mathbf{u}} - \tilde{\mathbf{u}}_h) \\ &\le C_\tau |||\mathbf{u} - \mathbf{u}_h|||_\tau |||\tilde{\mathbf{u}} - \tilde{\mathbf{u}}_h|||_\tau \le C_\tau \frac{h^2}{\nu} \|\mathbf{f}\| \, \|\tilde{\mathbf{f}}\| = C_\tau \frac{h^2}{\nu} \|\mathbf{f}\| \, \|\mathbf{u} - \mathbf{u}_h\|.\end{aligned}$$

Hence, $\|\mathbf{u} - \mathbf{u}_h\| \le C_\tau \frac{h^2}{\nu} \|\mathbf{f}\|$ holds, which proves the first bound in (3.22). For the second bound we note that from (2.5) and (A1) it follows that

(3.24)
$$\begin{aligned}\|\mathbf{u} - \mathbf{u}_h\| &\le \frac{1}{\alpha + c_w} (\alpha \|\mathbf{u} - \mathbf{u}_h\| + \|w \times (\mathbf{u} - \mathbf{u}_h)\|) \\ &\le \frac{1}{\alpha + \|w\|_\infty} \frac{\alpha + \|w\|_\infty}{\alpha + c_w} \left(\alpha^{\frac12} + \frac{\|w\|_\infty^{\frac12}}{\tau^{\frac12}} \right) \left(\alpha^{\frac12} \|\mathbf{u} - \mathbf{u}_h\| + \frac{\tau^{\frac12}}{\|w\|_\infty^{\frac12}} \|w \times (\mathbf{u} - \mathbf{u}_h)\| \right) \\ &\le \frac{2}{\alpha + \|w\|_\infty} (1 + \eta) \tau^{-\frac12} (\alpha^{\frac12} \tau^{\frac12} + \|w\|_\infty^{\frac12}) |||\mathbf{u} - \mathbf{u}_h|||_\tau \\ &\le C_\tau \frac{1}{\alpha + \|w\|_\infty} (\alpha^{\frac12} + \|w\|_\infty^{\frac12}) |||\mathbf{u} - \mathbf{u}_h|||_\tau.\end{aligned}$$

Finally, note that due to (3.18) with $j = 0$ and the results in (2.5), (2.8) we get

$$\begin{aligned}(\alpha^{\frac12} + \|w\|_\infty^{\frac12}) |||\mathbf{u} - \mathbf{u}_h|||_\tau &\le (\alpha^{\frac12} + \|w\|_\infty^{\frac12}) (\nu^{\frac12} \|\mathbf{u}\|_1 + (\alpha^{\frac12} + \|w\|_\infty^{\frac12}) \|\mathbf{u}\|) \\ &\le \nu^{\frac12} (\alpha^{\frac12} + \|w\|_\infty^{\frac12}) \|\mathbf{u}\|_1 + 2(\alpha + \|w\|_\infty) \|\mathbf{u}\| \\ &\le \nu^{\frac12} (\alpha^{\frac12} + \|w\|_\infty^{\frac12}) \|\mathbf{u}\|_1 + 2(1 + \eta)(\|w \times \mathbf{u}\| + \alpha \|\mathbf{u}\|) \\ &\le C \big(\nu (\alpha + \|w\|_\infty) \|\nabla \mathbf{u}\|^2 + \alpha^2 \|\mathbf{u}\|^2 + \|w \times \mathbf{u}\|^2 \big)^{\frac12} \\ &\le C \|\mathbf{f}\|.\end{aligned}$$

This in combination with (3.24) yields the second bound in (3.22). □

4. A solver for the discrete problem. For the approximate solution of the discrete problem we apply a multigrid method. The method and its convergence analysis will be presented in a matrix-vector form as in Hackbusch [8].

4.1. Multigrid components. For the application of the multigrid solver we assume that the quasi-uniform family of triangulations of Ω results from a *global regular refinement* technique. This yields a hierarchy of nested finite element spaces

$$\mathbf{U}_0 \subset \mathbf{U}_1 \subset \cdots \subset \mathbf{U}_k \subset \cdots \subset \mathbf{U}.$$

The corresponding mesh size parameter is denoted by h_k and satisfies

$$c_0 2^{-k} \leq h_k/h_0 \leq c_1 2^{-k}$$

with positive constants c_0 and c_1 independent of k. Note that $\mathbf{U}_k = U_k \times U_k$, where U_k is a standard conforming finite element space consisting of scalar functions. For the matrix-vector formulation of the discrete problem we use the standard nodal basis in U_k, denoted by $\{\phi_i\}_{1\leq i\leq n_k}$, and the isomorphism

$$P_k : \mathbb{R}^{n_k} \to U_k, \qquad P_k x = \sum_{i=1}^{n_k} x_i \phi_i.$$

For the product space $\mathbf{U}_k = U_k \times U_k$ we use the isomorphism

$$\mathbf{P}_k : X_k := \mathbb{R}^{2n_k} \to \mathbf{U}_k, \quad \mathbf{P}_k \mathbf{x} = \mathbf{P}_k \begin{pmatrix} x^1 \\ x^2 \end{pmatrix} = P_k x^1 \times P_k x^2, \quad x^i \in \mathbb{R}^{n_k}, \quad i = 1, 2.$$

On $\mathbb{R}^{n_k}$ and X_k we use scaled Euclidean scalar products: $\langle x, y\rangle_k = h_k^2 \sum_{i=1}^{n_k} x_i y_i$ for $x, y \in \mathbb{R}^{n_k}$ and $\langle \mathbf{x}, \mathbf{y}\rangle_k = \langle x^1, y^1\rangle_k + \langle x^2, y^2\rangle_k$ for $\mathbf{x}$, $\mathbf{y} \in X_k$. The corresponding norms are denoted by $\|\cdot\|$. The adjoint $\mathbf{P}_k^* : \mathbf{U}_k \to X_k$ satisfies $(\mathbf{P}_k\mathbf{x}, \mathbf{v}) = \langle \mathbf{x}, \mathbf{P}_k^*\mathbf{v}\rangle_k$ for all $\mathbf{x} \in X_k$, $\mathbf{v} \in \mathbf{U}_k$. Note that the following norm equivalence holds:

$$C^{-1}\|\mathbf{x}\| \leq \|\mathbf{P}_k\mathbf{x}\| \leq C\|\mathbf{x}\| \quad \text{for all } \mathbf{x} \in X_k, \tag{4.1}$$

with a constant C independent of k. The stiffness matrix $L_k : \mathbb{R}^{2n_k} \to \mathbb{R}^{2n_k}$ on level k is defined by

$$\langle L_k\mathbf{x}, \mathbf{y}\rangle_k = a(\mathbf{P}_k\mathbf{x}, \mathbf{P}_k\mathbf{y}) \quad \text{for all } \mathbf{x}, \mathbf{y} \in X_k. \tag{4.2}$$

This matrix has the block structure

$$L_k = \begin{pmatrix} \nu A + \alpha M & -M_w \\ M_w & \nu A + \alpha M \end{pmatrix},$$

with

$$\begin{aligned} \langle Ax, y\rangle_k &= (\nabla P_k x, \nabla P_k y), \quad \langle Mx, y\rangle_k = (P_k x, P_k y), \\ \langle M_w x, y\rangle_k &= (w P_k x, P_k y) \end{aligned} \tag{4.3}$$

for all $x, y \in \mathbb{R}^{n_k}$. Note that A is a stiffness matrix for a single (velocity) component, M is a mass matrix, and M_w is of mass matrix type corresponding to the bilinear form $[x, y] \to (wx, y)$. The latter is not necessarily a scalar product. The matrices A, M, M_w are symmetric and A and M are positive definite.

For the prolongation and restriction in the multigrid algorithm we use the canonical choice:

$$\begin{aligned} p_k &: X_{k-1} \to X_k, \qquad p_k = \mathbf{P}_k^{-1}\mathbf{P}_{k-1}, \\ r_k &: X_k \to X_{k-1}, \qquad r_k = \mathbf{P}_{k-1}^*(\mathbf{P}_k^*)^{-1} = \left(\tfrac{h_k}{h_{k-1}}\right)^2 p_k^T. \end{aligned} \tag{4.4}$$

Consider a smoother of the form

$$\mathbf{x}^{\text{new}} = \mathbf{x}^{\text{old}} - W_k^{-1}(L_k\mathbf{x}^{\text{old}} - \mathbf{b}) \quad \text{for } \mathbf{x}^{\text{old}}, \mathbf{b} \in X_k$$

with the corresponding iteration matrix denoted by $S_k = I - W_k^{-1}L_k$.

The damped block Jacobi method corresponds to

$$W_k = \omega^{-1} \begin{pmatrix} \mathrm{diag}(\nu A + \alpha M) & -\mathrm{diag}(M_w) \\ \mathrm{diag}(M_w) & \mathrm{diag}(\nu A + \alpha M) \end{pmatrix}, \tag{4.5}$$

with a damping parameter $\omega \in (0, 1]$. This type of smoother will be used in our numerical experiments in section 5. In the convergence analysis of the multigrid method we consider a smoother of block Richardson type:

$$W_k = \begin{pmatrix} \beta_1 I & -\beta_2 I \\ \beta_2 I & \beta_1 I \end{pmatrix}, \tag{4.6}$$

where I is the identity matrix and β_1, β_2 are suitable scaling factors. With the components defined above, a standard multigrid algorithm with μ_1 pre- and μ_2 postsmoothing iterations can be formulated (cf. [8]) with an iteration matrix M_k on level k that satisfies the recursion

$$\begin{aligned} &M_0(\mu_1, \mu_2) = 0, \\ &M_k(\mu_1, \mu_2) = S_k^{\mu_2} \left(I - p_k (I - M_{k-1}^{\gamma}) L_{k-1}^{-1} r_k L_k \right) S_k^{\mu_1}, \quad k = 1, 2, \dots . \end{aligned}$$

The choices $\gamma = 1$ and $\gamma = 2$ correspond to the V- and W-cycle, respectively. For analysis of this multigrid method we use the framework of [7], [8] based on the approximation and smoothing property. In sections 4.2 and 4.3 we will prove the following approximation and smoothing properties:

$$\|L_k^{-1} - p_k\, L_{k-1}^{-1} r_k\| \le C \left(\frac{\nu}{h^2} + \alpha + \|w\|_\infty \right)^{-1}, \tag{4.7}$$

$$\|L_k S_k^{\mu_1}\| \le \frac{C}{\sqrt{\mu_1}} \left(\frac{\nu}{h^2} + \alpha + \|w\|_\infty \right). \tag{4.8}$$

As a direct consequence of (4.7) and (4.8) one obtains a bound for the contraction number of the two-grid method:

$$\|(I - p_k L_{k-1}^{-1} r_k L_k) S_k^{\mu_1}\| \le \frac{C}{\sqrt{\mu_1}} \, . \tag{4.9}$$

Using the analysis in [8, Theorem 10.6.25] the convergence of the multigrid W-cycle can be obtained as a consequence of the approximation and smoothing property. In section 4.3 we will prove $\|S_k\| \le 1$. Using this and (4.7), (4.8), Theorem 10.6.25 from [8] yields the following result.

THEOREM 4.1. *Assume* (A1)–(A3) *hold; then for any* $\psi \in (0, 1)$ *there exists* $\bar{\mu}_0 > 0$ *independent of the problem parameters* ν, α *and the level number* k *such that for the contraction number of the multigrid W-cycle with smoothing* (4.6) *we have*

$$\|M_k(\mu, 0)\| \le \psi \quad \textit{for all } \mu \ge \bar{\mu}_0. \qquad \square$$

This proves the robustness of the multigrid W-cycle with respect to variation in the problem parameters ν *and* α *and the mesh size* h_k.

This robustness is confirmed by the numerical experiments in section 5.

4.2. Approximation property. The analysis of the approximation property is as in [7], [8]. The key ingredient is the finite element error bound in Theorem 3.3.

THEOREM 4.2. *Let the assumptions* (A1)–(A3) *be valid; then*

$$\|L_k^{-1} - p_k\, L_{k-1}^{-1} r_k\| \le C \left(\frac{\nu}{h_k^2} + \alpha + \|w\|_\infty \right)^{-1} \le C \|L_k\|^{-1}. \tag{4.10}$$

Proof. Take $\mathbf{y}_k \in X_k$. The constants C that appear in the proof do not depend on $\nu, \alpha, \mathbf{y}_k$, or k. Let $\mathbf{s}^* \in \mathbf{U}$, $\mathbf{s}_k \in \mathbf{U}_k$, and $\mathbf{s}_{k-1} \in \mathbf{U}_{k-1}$ be such that

$$\begin{aligned} a(\mathbf{s}^*, \mathbf{v}) &= ((\mathbf{P}_k^*)^{-1}\mathbf{y}_k, \mathbf{v}) \quad \text{for all } \mathbf{v} \in \mathbf{U}, \\ a(\mathbf{s}_k, \mathbf{v}) &= ((\mathbf{P}_k^*)^{-1}\mathbf{y}_k, \mathbf{v}) \quad \text{for all } \mathbf{v} \in \mathbf{U}_k, \\ a(\mathbf{s}_{k-1}, \mathbf{v}) &= ((\mathbf{P}_k^*)^{-1}\mathbf{y}_k, \mathbf{v}) \quad \text{for all } \mathbf{v} \in \mathbf{U}_{k-1}. \end{aligned}$$

Putting $\mathbf{f} = (\mathbf{P}_k^*)^{-1}\mathbf{y}_k \in L_2(\Omega)^2$ in Theorem 3.3, we obtain

$$\|\mathbf{s}^* - \mathbf{s}_l\| \le C \min\left\{ \frac{h_l^2}{\nu}, \frac{1}{\alpha + \|w\|_\infty} \right\} \|(\mathbf{P}_k^*)^{-1}\mathbf{y}_k\| \quad \text{for } l \in \{k-1, k\}.$$

Due to $h_{k-1} \le c h_k$ this yields

$$\|\mathbf{s}_k - \mathbf{s}_{k-1}\| \le C \min\left\{ \frac{h_k^2}{\nu}, \frac{1}{\alpha + \|w\|_\infty} \right\} \|(\mathbf{P}_k^*)^{-1}\mathbf{y}_k\|.$$

From (4.2) and (4.4) it follows that $\mathbf{s}_k = \mathbf{P}_k L_k^{-1}\mathbf{y}_k$ and $\mathbf{s}_{k-1} = \mathbf{P}_{k-1} L_{k-1}^{-1} r_k \mathbf{y}_k$. Thus, using (4.1), we get

$$\begin{aligned} \|(L_k^{-1} - p_k L_{k-1}^{-1} r_k)\mathbf{y}_k\| &\le C \|\mathbf{P}_k L_k^{-1}\mathbf{y}_k - \mathbf{P}_{k-1} L_{k-1}^{-1} r_k \mathbf{y}_k\| = C\|\mathbf{s}_k - \mathbf{s}_{k-1}\| \\ &\le C \, \min\left\{ \frac{h_k^2}{\nu}, \frac{1}{\alpha + \|w\|_\infty} \right\} \|(\mathbf{P}_k^*)^{-1}\mathbf{y}_k\| \\ &\le C \, \min\left\{ \frac{h_k^2}{\nu}, \frac{1}{\alpha + \|w\|_\infty} \right\} \|\mathbf{y}_k\|. \end{aligned}$$

Note that $\min\{\frac{1}{p}, \frac{1}{q}\} \le \frac{2}{p+q}$ for all $p, q > 0$. Hence the first inequality in (4.10) is proved. For the second inequality in (4.10) we note that

$$\begin{aligned} \|L_k\| &= \left\| \begin{pmatrix} \nu A + \alpha M & \emptyset \\ \emptyset & \nu A + \alpha M \end{pmatrix} + \begin{pmatrix} \emptyset & -M_w \\ M_w & \emptyset \end{pmatrix} \right\| \\ &\le \|\nu A + \alpha M\| + \|M_w\| \le \nu\|A\| + (\alpha + \|w\|_\infty)\|M\|. \end{aligned}$$

Using $\|A\| \le C h_k^{-2}$ and $\|M\| \le C$ we obtain $\|L_k\| \le C(\nu h_k^{-2} + \alpha + \|w\|_\infty)$. □

4.3. Smoothing property. Let a_1, m_1 be positive constants independent of ν, α, and k such that for spectral radius of the matrices in (4.3) we have

$$\rho(A) \le \frac{a_1}{h_k^2}, \qquad \rho(M) \le m_1.$$

Furthermore, let $w_{\min} = \text{ess inf}_\Omega\, w$ and $w_{\max} = \text{ess sup}_\Omega\, w$ and define

$$C_w = \begin{cases} w_{\max} & \text{if } w_{\max} \ge -w_{\min}, \\ w_{\min} & \text{if } w_{\max} < -w_{\min}. \end{cases}$$

Note that $|C_w| = \|w\|_\infty$. In the analysis below we use the following elementary result.

Lemma 4.3. *Assume that for $B \in \mathbb{R}^{n\times n}$ and $\Lambda \in (0,\infty)$ we have $B^TB \le \Lambda(B+B^T)$. Then $\|I-\omega B\| \le 1$ holds for any $\omega \in [0, \frac{1}{\Lambda}]$.*

This result follows from

$$\begin{aligned} 0 \le (I-\omega B)^T(I-\omega B) &= I - \omega(B+B^T) + \omega^2 B^TB \\ &\le I - \omega(1-\omega\Lambda)(B+B^T) \le I. \qquad \Box \end{aligned}$$

Using this lemma we prove that the contraction number of the block Richardson method is bounded by 1.

Lemma 4.4. *Assume that* (A1) *and* (A2) *are satisfied. Consider the block Richardson method with W_k as in* (4.6) *and*

$$\beta_1 = \frac{\nu a_1}{h_k^2} + \alpha\kappa_1 m_1, \quad \beta_2 = \kappa_2 C_w, \quad \textit{with constants}$$

$$\kappa_1 \ge 2(1+\eta^2), \quad \kappa_2 \ge 4m_1\eta. \tag{4.11}$$

Then the following inequality holds:

$$\|I - W_k^{-1}L_k\| \le 1.$$

Proof. A straightforward computation yields

$$W_k^{-1}L_k = R_1 + R_2, \qquad \text{with} \tag{4.12}$$

$$R_1 = \frac{\nu}{\beta_1^2+\beta_2^2}\begin{pmatrix} \beta_1 A & \beta_2 A \\ -\beta_2 A & \beta_1 A \end{pmatrix},$$

$$R_2 = \frac{1}{\beta_1^2+\beta_2^2}\begin{pmatrix} \beta_1\alpha M + \beta_2 M_w & \beta_2\alpha M - \beta_1 M_w \\ -\beta_2\alpha M + \beta_1 M_w & \beta_1\alpha M + \beta_2 M_w \end{pmatrix}.$$

From

$$\frac{1}{2}(R_1^T + R_1) = \frac{\nu\beta_1}{\beta_1^2+\beta_2^2}\begin{pmatrix} A & 0 \\ 0 & A \end{pmatrix}, \qquad R_1^TR_1 = \frac{\nu^2}{\beta_1^2+\beta_2^2}\begin{pmatrix} A^2 & 0 \\ 0 & A^2 \end{pmatrix}$$

it follows that

$$R_1^TR_1 \le \frac{1}{2}(R_1^T+R_1) \;\Leftrightarrow\; \nu A \le \beta_1 I \;\Leftrightarrow\; \nu A \le \left(\frac{\nu a_1}{h_k^2} + \alpha\kappa_1 m_1\right) I.$$

The last inequality holds, due to $\rho(A) \le \frac{a_1}{h_k^2}$ and $\alpha\kappa_1 m_1 \ge 0$. Application of Lemma 4.3 yields

$$\|I - 2R_1\| \le 1\ . \tag{4.13}$$

For the matrix R_2 we obtain

$$\frac{1}{2}(R_2^T+R_2) = \frac{1}{\beta_1^2+\beta_2^2}\begin{pmatrix} \beta_1\alpha M + \beta_2 M_w & \emptyset \\ \emptyset & \beta_1\alpha M + \beta_2 M_w \end{pmatrix},$$

$$R_2^TR_2 = \frac{1}{\beta_1^2+\beta_2^2}\begin{pmatrix} \alpha^2M^2 + M_w^2 & \alpha(M_wM - MM_w) \\ -\alpha(M_wM - MM_w) & \alpha^2M^2 + M_w^2 \end{pmatrix}.$$

We use the notation $\hat{M} = \beta_1\alpha M + \beta_2 M_w$. Note that $R_2^TR_2 \le \frac{1}{2}(R_2^T + R_2)$ holds if the following two conditions are satisfied:

$$\alpha^2M^2 + M_w^2 \le \frac{1}{2}\hat{M}, \tag{4.14}$$

$$\alpha|\langle(M_wM - MM_w)x, y\rangle_k| \le \frac{1}{4}\Big(\langle\hat{M}x, x\rangle_k + \langle\hat{M}y, y\rangle_k\Big), \tag{4.15}$$

for all $x, y \in \mathbb{R}^{n_k}$. We first consider (4.14). We have $M_w^2 \leq \|w\|_\infty^2 M^2 \leq m_1 \|w\|_\infty^2 M$. Due to (A2) the matrix M_w is definite and $C_w M_w$ is positive definite; moreover, $C_w M_w \geq |C_w| c_w M = \|w\|_\infty c_w M$. Using this we obtain

$$\alpha^2 M^2 + M_w^2 \leq (m_1 \alpha^2 + m_1 \|w\|_\infty^2) M,$$
$$\frac{1}{2}\hat{M} \geq \frac{1}{2}(\kappa_1 m_1 \alpha^2 M + \kappa_2 C_w M_w) \geq \frac{1}{2}(\kappa_1 m_1 \alpha^2 + \kappa_2 \|w\|_\infty c_w) M.$$

Hence, (4.14) is fulfilled if the inequality

$$m_1 \alpha^2 + m_1 \|w\|_\infty^2 \leq \frac{1}{2}(\kappa_1 m_1 \alpha^2 + \kappa_2 \|w\|_\infty c_w)$$

holds. Substitution of $\|w\|_\infty = \eta(\alpha + c_w)$ and rearranging terms results in the equivalent inequality

$$\alpha^2 m_1 \left(\frac{1}{2}\kappa_1 - (1 + \eta^2)\right) + \alpha c_w \eta \left(\frac{1}{2}\kappa_2 - 2 m_1 \eta\right) + \eta c_w^2 \left(\frac{1}{2}\kappa_2 - m_1 \eta\right) \geq 0.$$

This inequality holds for κ_1, κ_2 as in (4.11). Hence, with κ_1, κ_2 as in (4.11) the condition (4.14) is fulfilled. To prove (4.15) we note that

$$\begin{aligned} \alpha|\langle (M_w M - M M_w) x, y\rangle_k| &\leq \alpha(\langle|M_w M x, y\rangle_k| + \alpha|\langle M M_w x, y\rangle_k|, \\ \alpha|\langle M_w M x, y\rangle_k| = \alpha|\langle M x, M_w y\rangle_k| &\leq \tfrac{1}{2}\big(\alpha^2 \langle M^2 x, x\rangle_k + \langle M_w^2 y, y\rangle_k\big), \\ \alpha|\langle M M_w x, y\rangle_k| = \alpha|\langle M_w x, M y\rangle_k| &\leq \tfrac{1}{2}\big(\langle M_w^2 x, x\rangle_k + \alpha^2 \langle M^2 y, y\rangle_k\big). \end{aligned}$$

Thus (4.15) follows from (4.14). We conclude that (4.15) and (4.14) are satisfied for κ_1, κ_2 as in (4.11). Hence, $R_2^T R_2 \leq \frac{1}{2}(R_2^T + R_2)$ holds. And due to Lemma 4.3

$$\|I - 2R_2\| \leq 1. \tag{4.16}$$

Finally, (4.12), (4.13), and (4.16) yield

$$\|I - W_k^{-1} L_k\| = \|I - (R_1 + R_2)\| \leq \frac{1}{2}\|I - 2R_1\| + \frac{1}{2}\|I - 2R_2\| \leq 1. \qquad \square$$

Theorem 4.5. *Assume that* (A1) *and* (A2) *are satisfied. Consider the block Richardson method with* W_k *as in* (4.6) *and*

$$\beta_1 = 2\left(\frac{\nu a_1}{h_k^2} + \alpha \kappa_1 m_1\right), \qquad \beta_2 = 2\kappa_2 C_w,$$

with constants κ_1, κ_2 *from* (4.11). *Then the following estimate holds:*

$$\|L_k S_k^{\mu_1}\| \leq \frac{C}{\sqrt{\mu_1}}\left(\frac{\nu}{h^2} + \alpha + \|w\|_\infty\right), \quad \mu_1 = 1, 2, \ldots\ . \tag{4.17}$$

Proof. From Lemma 4.4 we obtain

$$\|I - 2W_k^{-1} L_k\| \leq 1. \tag{4.18}$$

Furthermore,

$$\begin{aligned} \|W_k\| &= \rho\left(\begin{pmatrix} \beta_1 I & -\beta_2 I \\ \beta_2 I & \beta_1 I \end{pmatrix}\begin{pmatrix} \beta_1 I & \beta_2 I \\ -\beta_2 I & \beta_1 I \end{pmatrix}\right)^{\frac{1}{2}} \\ &= (\beta_1^2 + \beta_2^2)^{\frac{1}{2}} \leq \beta_1 + \beta_2 \leq C\left(\frac{\nu}{h^2} + \alpha + \|w\|_\infty\right). \end{aligned} \tag{4.19}$$

From (4.18) and (4.19) and Theorem 10.6.8 in [8] the result in (4.17) follows. $\square$

5. Numerical results. In this section results of a few numerical experiments related to the accuracy of the discretization method and the convergence behavior of the multigrid solver are presented. For the discretization we use linear conforming finite elements on a uniform triangulation of the unit square. The mesh size parameter is $h = h_k = 2^{-k},\ k = 4, 5, \dots, 9$.

In our experiments we consider problems with an a priori known continuous solution $\mathbf{u} \in H^2(\Omega)^2 \cap \mathbf{U}$ to the problem (2.1). Discretization errors are measured as follows. Let $\hat{\mathbf{u}}_h \in \mathbf{U}_h$ be the nodal interpolant of the continuous solution $\mathbf{u}$ and $\mathbf{u}_h \in \mathbf{U}_h$ be the solution of the discrete problem. As a measure for the discretization error we take

$$err(\mathbf{u}, h, \nu) = \frac{\|\hat{\mathbf{u}}_h - \mathbf{u}_h\|}{\|\mathbf{f}\|}. \tag{5.1}$$

For the iterative solution of the discrete problem a multigrid V-cycle is applied. The prolongations and restrictions in this multigrid method are the canonical ones, as in (4.4). For the smoother a damped block Jacobi method as in (4.5) is used. Thus for each pair of nodal values of $\{u_1, u_2\}$ a 2×2 linear system is solved. The damping parameter ω in each smoothing step is determined in a dynamic way based on a residual minimization criterion: We set $\omega = (\mathbf{q}, \mathbf{q})/(\mathbf{q}, \mathbf{r})$, where for grid level k

$$\mathbf{r} = \bar{W}_k^{-1}(L_k \mathbf{x}^{old} - b), \qquad \mathbf{q} = \bar{W}_k^{-1} L_k \mathbf{r},$$

and $\bar{W}_k$ equals W_k from (4.5) for $\omega = 1$.

We always use two pre- and two postsmoothing iterations. For the starting vector in the iterative solver we take $\mathbf{u}^0 = 0$. The iterations are stopped as soon as the residual, in the Euclidean norm, is at least a factor 10^9 smaller than the starting residual.

We consider test problems with different choices for w. Note that in the setting of a (linearized) Navier–Stokes problem $w = \operatorname{curl} \mathbf{v} = -\frac{\partial v_2}{\partial x} + \frac{\partial v_1}{\partial y}$, where $\mathbf{v} = (v_1(x, y), v_2(x, y))$ is an approximation of the flow field. In Experiment I we consider a problem which corresponds to a flow with rotating vortices. In Experiment II we take a flow field $\mathbf{v}$ with a parabolic boundary layer behavior. Both in Experiment I and Experiment II the right-hand side is taken such that the continuous solution $\mathbf{u}$ equals the flow field $\mathbf{v}$. This seems a reasonable choice if the problem (2.1) results from a linearized Navier–Stokes problem. Finally, in Experiment III a flow $\mathbf{v}$ which exhibits an internal layer behavior is considered.

In all the experiments we present results for the case $\alpha = 0$. For $\alpha > 0$ in our numerical experiments we always observed better results than for $\alpha = 0$, both with respect to the discretization error and with respect to the multigrid convergence.

Experiment Ia. We take $\mathbf{v}_r = (v_1, v_2)$, with

$$\begin{aligned} v_1(x, y) &= 4(2y - 1)x(1 - x), \\ v_2(x, y) &= -4(2x - 1)y(1 - y), \end{aligned} \tag{5.2}$$

and $w = \operatorname{curl} \mathbf{v}_r$. This type of convection $\mathbf{v}_r$ simulates a rotating vortex. For this w the conditions (A2) and (A3) are fulfilled. Related to (A1) we note that $\|w\|_\infty = \mathcal{O}(1)$ and $c_w = 0$. However, based on the fact that w equals zero only at the corner points of the domain, one could say that (A1) is "almost" fulfilled. For several values of h and ν the quantity $err(\mathbf{u}, h, \nu)$ is given in Table 5.1.

In Figure 5.1 the differences $(u_1 - (u_h)_1)(0.5, y)$ and $(\frac{\partial u_1}{\partial y} - \frac{\partial (u_h)_1}{\partial y})(0.5, y)$ between (the derivatives of) the first components of the continuous and finite element solution

TABLE 5.1
$err(\mathbf{u}, h, \nu)$ for Experiment Ia.

	h					
ν	1/16	1/32	1/64	1/128	1/256	1/512
1	4.5e-4	1.1e-4	2.8e-5	7.2e-6	1.8e-6	4.5e-7
1e-2	8.6e-3	2.1e-3	5.2e-4	1.3e-4	3.3e-5	8.2e-6
1e-4	1.0e-2	2.7e-3	7.0e-4	1.7e-4	4.4e-5	1.1e-5
1e-6	1.0e-2	2.7e-3	7.7e-4	2.1e-4	5.4e-5	1.3e-5
1e-8	1.0e-2	2.7e-3	7.7e-4	2.1e-4	5.9e-5	1.6e-5

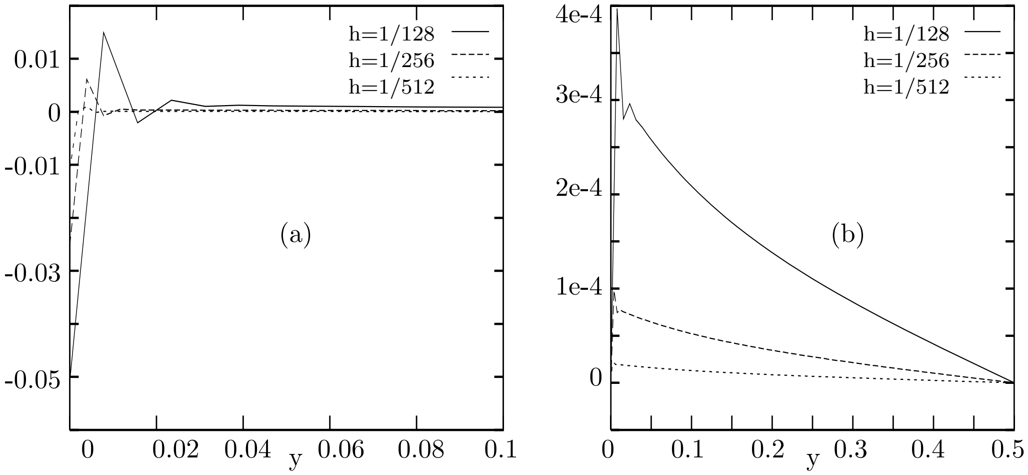


FIG. 5.1. *Discretization error in Experiment* Ia*;* $\nu = 10^{-6}$, $x = 0.5$ (a) *in y-derivative,* (b) *in solution.*

TABLE 5.2
V-cycle convergence for Experiment Ia.

	h				
ν	1/32	1/64	1/128	1/256	1/512
1	**11**(0.15)	**11**(0.15)	**11**(0.15)	**11**(0.15)	**11**(0.15)
1e-2	**11**(0.14)	**11**(0.14)	**11**(0.14)	**11**(0.15)	**11**(0.15)
1e-4	**6**(0.03)	**7**(0.05)	**9**(0.10)	**11**(0.14)	**11**(0.15)
1e-6	**5**(0.01)	**5**(0.01)	**5**(0.01)	**7**(0.04)	**7**(0.05)
1e-8	**5**(0.01)	**5**(0.01)	**5**(0.01)	**5**(0.01)	**5**(0.01)

Number of iterations and average reduction factor

are plotted for the case $\nu = 10^{-6}$. Because of the symmetry the error in the solution is shown only on half of the interval (Figure 5.1b) and the error in the solution derivative only on the interval $[0, 0.1]$ near the boundary (Figure 5.1a). The numerical boundary layer, typical for reaction-diffusion problems with dominating reaction terms, is clearly seen. Results for the convergence behavior of the multigrid method are shown in Table 5.2.

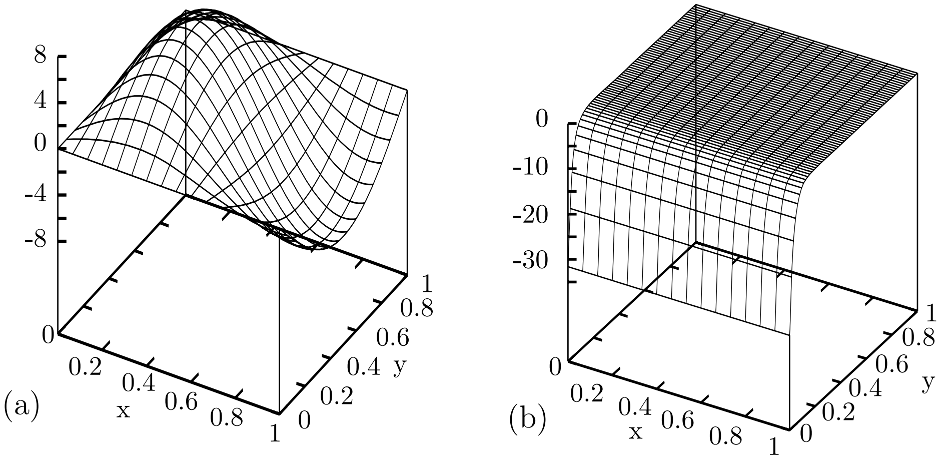


Fig. 5.2. (a) *Function* w *in Experiment* Ib*;* (b) *function* w *in Experiment* II*,* $\nu = 10^{-3}$.

Table 5.3
$err(\mathbf{u}, h, \nu)$ *for Experiment* Ib.

	h					
ν	1/16	1/32	1/64	1/128	1/256	1/512
1	1.9e-3	4.9e-4	1.2e-4	3.0e-5	7.5e-6	1.9e-6
1e-2	1.5e-2	3.6e-3	9.0e-4	2.3e-4	5.7e-5	1.4e-5
1e-4	4.8e-2	7.1e-3	1.8e-3	4.5e-4	1.1e-4	2.9e-5
1e-6	1.4e-1	7.8e-2	1.0e-2	9.5e-4	2.3e-4	5.7e-5
1e-8	1.4e-1	9.7e-2	6.7e-2	2.9e-2	2.0e-3	1.4e-4

Experiment Ib. We take $\mathbf{v}_R = (v_1, v_2)$, with

$$v_1(x,y) = \frac{1}{\psi}\sin(\psi\pi x)\cos(\pi y), \qquad v_2(x,y) = -\cos(\psi\pi x)\sin(\pi y), \tag{5.3}$$

and $w = \operatorname{curl}\mathbf{v}_R$. This models a flow with two vortices rotating in opposite directions. Note that the conditions (A1) and (A2) are not fulfilled. For the parameter ψ we choose $\psi = 1.6$. One vortex lies entirely in the computational domain, the second one only partially. The (vorticity) function w for this problem is plotted in Figure 5.2(a). Note the change of sign for w at $x = 0.625$. The error in the discrete solution shown in Table 5.3 is larger compared to example Ia (which might correspond to the strong violation of the conditions (A1) and (A2)). In Figure 5.3 the difference $(u_1 - (u_h)_1)(0.5, y)$ is plotted for $\nu = 10^{-6}$. Note that some local oscillations in the error are observed in the neighborhood of $x = 0.625$, i.e., where condition (A1) is *locally* violated. The results for the convergence behavior of the multigrid method are very similar to those in Table 5.2 for Experiment Ia.

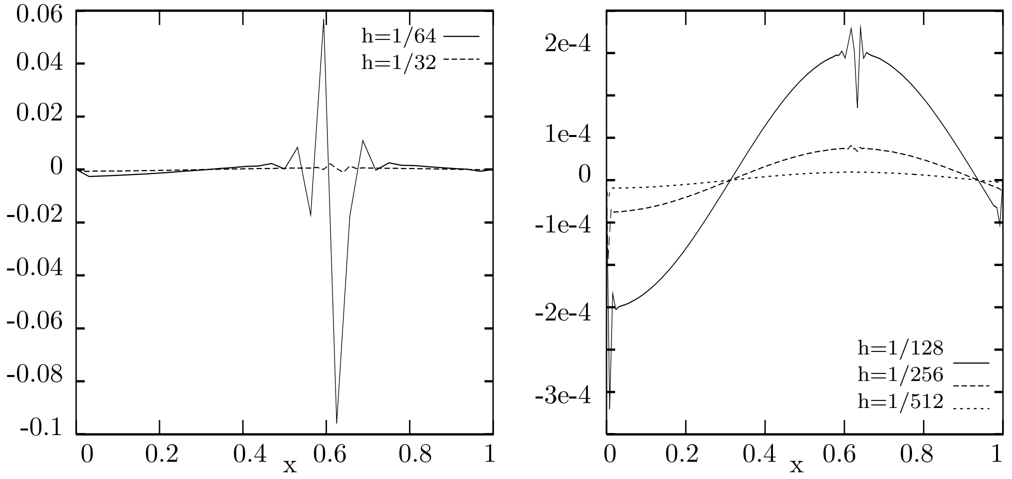


FIG. 5.3. *Error in finite element solutions in Experiment* Ib; $\nu = 10^{-6}$, $y = 0.5$.

TABLE 5.4
$err(\mathbf{u}, h, \nu)$ *for Experiment* II.

	h					
ν	1/16	1/32	1/64	1/128	1/256	1/512
1	7.4e-6	1.8e-6	4.5e-7	1.1e-7	2.8e-8	7.0e-9
1e-2	3.7e-3	8.6e-3	2.1e-4	5.3e-5	1.3e-5	2.2e-6
1e-4	4.2e-2	2.4e-2	3.1e-3	6.8e-4	1.6e-4	4.1e-5
1e-6	1.2e-2	1.2e-2	1.2e-2	1.2e-2	1.0e-2	8.0e-4
1e-8	3.9e-3	3.7e-3	3.7e-3	3.7e-3	3.6e-3	3.6e-3

Experiment II. We take $\mathbf{v}_l = (v_1, v_2)$, with

$$
\begin{aligned}
v_1(x,y) &= 1 - \exp(-y/\sqrt{\nu}),\\
v_2(x,y) &= 0,
\end{aligned}
\tag{5.4}
$$

and $w = \operatorname{curl} \mathbf{v}_l$. This models a parabolic boundary layer behavior in the velocity field. The width of the layer is proportional to $\sqrt{\nu}$. Note that $\|w\|_\infty = O(\nu^{-1/2})$. The vorticity is of $\nu^{-\frac{1}{2}}$ magnitude near the boundary and decays exponentially outside the layer (see Figure 5.2(b)). As before, we take $\mathbf{f}$ such that the continuous solution equals the flow field: $\mathbf{u} = \mathbf{v}_l$. Results for the discretization error are given in Table 5.4. The L_2 norm of $\mathbf{f}$ is $O(\nu^{-\frac{1}{4}})$ for $\nu \to 0$; therefore one has to use a proper scaling of the values from Table 5.4 (e.g., multiplying by 10 for $\nu = 10^{-4}$) to obtain the absolute value of the error $\|\hat{\mathbf{u}}_h - \mathbf{u}_h\|$ (cf. (5.1)).

In Figure 5.4 we plot $u_1(0.5, y)$ and $(u_h)_1(0.5, y)$ for the cases $\nu = 10^{-3}$ and $\nu = 10^{-4}$ and for several h values. The finite element solution is a poor approximation to the continuous one if the boundary layer is not resolved: $h > \nu^{\frac{1}{2}}$. However, for $h \sim \nu^{\frac{1}{2}}$ the results are quite good, although both the *mesh* Reynolds numbers and Ek_h^{-1} are very large (e.g., $\approx 10^2$ for $\nu = 10^{-4}$). Moreover, no global oscillations are observed even for very coarse meshes. We expect that a significant improvement can be obtained if this simple full Galerkin discretization is combined with local grid

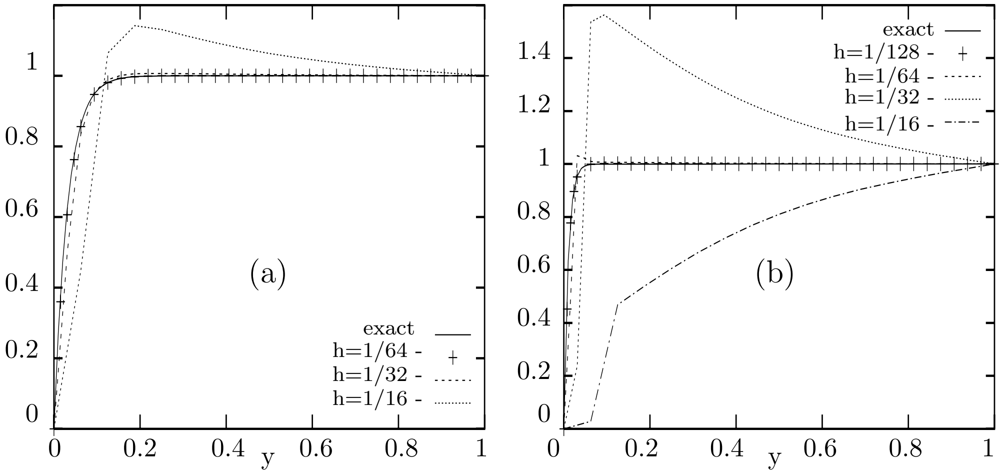


FIG. 5.4. *Exact and discrete solutions in Experiment* II*;* $x = 0.5$*:* (a) $\nu = 10^{-3}$*;* (b) $\nu = 10^{-4}$.

TABLE 5.5
V-cycle convergence for Experiment II.

	h				
ν	1/32	1/64	1/128	1/256	1/512
1	**11**(0.15)	**11**(0.15)	**11**(0.15)	**11**(0.15)	**11**(0.15)
1e-2	**12**(0.16)	**11**(0.15)	**11**(0.15)	**11**(0.15)	**11**(0.15)
1e-4	**18**(0.30)	**17**(0.29)	**16**(0.26)	**14**(0.22)	**13**(0.19)
1e-6	**23**(0.40)	**29**(0.48)	**29**(0.49)	**28**(0.41)	**29**(0.48)
1e-8	**15**(0.24)	**19**(0.33)	**23**(0.40)	**28**(0.47)	**25**(0.43)

Number of iterations and average reduction factor

refinement in the boundary layer. In Table 5.5 numerical results for the multigrid method are presented. Note that assumptions (A1) and (A2) were also violated in this experiment. Hence our convergence analysis of the multigrid method does not apply here. One reason for the deterioration of multigrid convergence compared to the case Ib could be weaker regularity of the function w.

Experiment III. In this experiment we try to model the presence of an internal layer. To this end, for the convection field we take the model of the Euler flow (extreme case if $\nu \to 0$), where the tangential velocity component is discontinuous on some line in the interior of the domain. Hence the flow, potential a.e., has a vorticity concentrated on this line (so-called vortex sheet). We take $w = \operatorname{curl} \mathbf{v}_d$, with $\mathbf{v}_d = (v_1, v_2)$, and, for a given constant ψ,

$$\left\{ \begin{array}{l} v_1(x,y) = \cos\psi \\ v_2(x,y) = \sin\psi \end{array} \right. \quad \text{if } \cos\psi > (x - 0.25)\sin\psi,$$

$$\left\{ \begin{array}{l} v_1(x,y) = 0 \\ v_2(x,y) = 0 \end{array} \right. \quad \text{if } \cos\psi \le (x - 0.25)\sin\psi.$$

Using the parameter ψ one can vary the angle under which the layer enters the domain. We set $\psi = \pi/3$ so the grid is not aligned to the layer. For the discrete velocity $\mathbf{v}_h^d \in$

Table 5.6
V-cycle convergence for Experiment III.

	h				
ν	1/32	1/64	1/128	1/256	1/512
1	**11**(0.15)	**11**(0.15)	**11**(0.15)	**11**(0.15)	**11**(0.15)
1e-2	**13**(0.20)	**13**(0.19)	**14**(0.22)	**14**(0.21)	**13**(0.19)
1e-4	**19**(0.33)	**19**(0.34)	**20**(0.35)	**21**(0.36)	**22**(0.38)
1e-6	**17**(0.29)	**20**(0.36)	**24**(0.42)	**28**(0.47)	**30**(0.50)
1e-8	**17**(0.29)	**20**(0.35)	**24**(0.42)	**28**(0.48)	**32**(0.53)

Number of iterations and average reduction factor

$\mathbf{U}_h$ we take the nodal interpolant of $\mathbf{v}_d$, and set $w = \operatorname{curl} \mathbf{v}_h^d$, obtaining a piecewise constant function w, which is essentially mesh-dependent due to the discontinuity of $\mathbf{v}_d$ ($\|w\|_\infty = O(h^{-1})$). Results for the convergence behavior of the multigrid method are given in Table 5.6.

Since discontinuous solutions are generally not allowed for viscous motions and our given data are mesh-dependent, we do not consider discretization errors in this example.

5.1. Discussion of numerical results. Recall that the analysis in the previous sections yields, for the case $\alpha = 0$,

$$err(\mathbf{u}, h, \nu) \leq c \min\{\nu^{-1}h^2, \|w\|_\infty^{-1}\} \tag{5.5}$$

under certain assumptions on w. These assumptions are "almost valid" for the problem Ia and do not hold for the problems Ib and II.

The results of the numerical experiments indeed show the $O(h^2)$ behavior of $err(\mathbf{u}, h, \nu)$ unless ν is very small. In the latter case the second, ν- and h-independent, upper bound for $err(\mathbf{u}, h, \nu)$ in (5.5) is observed and $O(h^2)$ convergence is recovered for smaller h. For fixed h and $\nu \to 0$ a growth of the error is observed (up to some limit). In the experiments Ia,b this growth appears to be less than $O(\nu^{-1})$, indicating that the ν-dependence in (5.5) might be somewhat pessimistic for these cases.

Although in the last two examples the multigrid convergence for a small values of ν is somewhat worse, the multigrid V-cycle with block Jacobi smoothing appears to be a very robust solver. The convergence rates for realistic values of viscosity (in laminar flows $1 - 10^{-4}$) are excellent.

REFERENCES

[1] J. Bey and G. Wittum, *Downwind numbering: Robust multigrid for convection-diffusion problems*, Appl. Numer. Math., 23 (1997), pp. 177–192.

[2] J. Bramble and J. Xu, *Some estimates for a weighted L^2 Projection*, Math. Comp., 56 (1991), pp. 463–476.

[3] J. H. Bramble, J. E. Pasciak, and A. T. Vassilev, *Uzawa type algorithms for nonsymmetric saddle point problems*, Math. Comp., 69 (2000), pp. 667–689.

[4] A. Brandt and I. Yavneh, *Accelerated multigrid convergence and high-Reynolds recirculating flows*, SIAM J. Sci. Comput., 14 (1993), pp. 607–626.

[5] P. G. Ciarlet, *Basic error estimates for elliptic problems*, in Handbook of Numerical Analysis, Vol. 2, P.G. Ciarlet and J. L. Lions, eds., North-Holland, Amsterdam, 1991, pp. 17–351.

[6] R. Codina, *Finite element solution of the Stokes problem with dominating Coriolis force*, Comput. Meth. Appl. Mech. Engrg., 142 (1997), pp. 215–234.

[7] W. Hackbusch, *Multi-grid Methods and Applications*, Springer, Berlin, Heidelberg, 1985.

[8] W. Hackbusch, *Iterative Solution of Large Sparse Systems of Equations*, Springer, Berlin, Heidelberg, 1994.

[9] W. Hackbusch and T. Probst, *Downwind Gauss-Seidel smoothing for convection dominated problems*, Numer. Linear Algebra Appl., 4 (1997), pp. 85–102.

[10] G. Lube and M. A. Olshanskii, *Stable Finite Element Calculation of Incompressible Flow Using the Rotational Form of Convection*, Preprint 189, Institut für Geometrie und Praktische Mathematik of RWTH-Aachen, Aachen, Germany, 2000, IMA J. Numer. Anal., to appear.

[11] M. F. Murphy, G. H. Golub, and A. J. Wathen, *A note on preconditioning for indefinite linear systems*, SIAM J. Sci. Comput., 21 (2000), pp. 1969–1972.

[12] M. A. Olshanskii, *An iterative solver for the Oseen problem and the numerical solution of incompressible Navier–Stokes equations*, Numer. Linear Algebra Appl., 6 (1999), pp. 353–378.

[13] M. A. Olshanskii, *A low order Galerkin finite element method for the Navier–Stokes equations of incompressible flow: A stabilization issue and iterative methods*, Preprint 202, Institut für Geometrie und Praktische Mathematik of RWTH-Aachen, Aachen, Germany, 2001; also available online from http://www.igpm.rwth-aachen.de./www/rep_2001.html.

[14] A. Ramage, *A multigrid preconditioner for stabilized discretisations of advection-diffusion problems*, J. Comput. Appl. Math., 110 (1999), pp. 187–203.

[15] A. Reusken, *Fourier analysis of a robust multigrid method for convection-diffusion equations*, Numer. Math., 71 (1995), pp. 365–397.

[16] H.-G. Roos, M. Stynes, and L. Tobiska, *Numerical methods for singularly perturbed differential equations: Convection diffusion and flow problems*, Springer Ser. Comput. Math. 24, Springer, Berlin, Heidelberg, 1996.

[17] A. H. Schatz and L. B. Wahlbin, *On the finite element method for singularly perturbed reaction-diffusion problems in two and one dimentions*, Math. Comput., 40 (1983), pp. 47–89.

[18] L. I. Sedov, *Mechanics of the Continuum Media*, Nauka, Moscow, 1970.

[19] D. J. Silvester, H. C. Elman, D. Kay, and A. J. Wathen, *Efficient preconditioning of the linearized Navier–Stokes equations for incompressible flow*, J. Comput. Appl. Math., 128 (2001) pp. 261–279.

[20] S. Turek, *Efficient Solvers for Incompressible Flow Problems: An Algorithmic Approach in View of Computational Aspects*, Lecture Notes Comput. Sci. Eng. 6, Springer, Berlin, Heidelberg, 1999.

[21] C. B. Vreugdehil and B. Koren, eds., *Numerical Methods for Advection-Diffusion Problem*, Notes Numer. Fluid Mech., 45, Vieweg, Braunschweig, Weisbaden, 1993.

[22] L. B. Wahlbin, *Local behavior in finite element methods*, in Handbook of Numerical Analysis, Vol. 2, P. G. Ciarlet and J. L. Lions, eds., North-Holland, Amsterdam, 1991, pp. 353–522.

[23] I. Yavneh, C. H. Venner, and A. Brandt, *Fast multigrid solution of the advection problem with closed characteristics*, SIAM J. Sci. Comput., 19 (1998), pp. 111–125.

SIAM J. Sci. Comput.
Vol. 23, No. 5, pp. 1707–1730

OPTIMAL TWO-DIMENSIONAL FINITE DIFFERENCE GRIDS PROVIDING SUPERCONVERGENCE*

NAIL K. YAMALEEV†

Abstract. A novel multidimensional grid adaptation method based on minimization of the leading truncation error term of arbitrary second- and higher-order finite difference approximations is proposed. The method does not explicitly require the truncation error estimate, but increases the design order of approximation globally by one, so that the same finite difference operator is superconvergent on the optimal grid. If the differential operator and the metric coefficients are evaluated identically by some hybrid approximation, the single optimal grid generator can be used in the entire computational domain and does not depend on points where the hybrid discretization switches from one approximation to another. If one family of the coordinate lines is given a priori, then the analytical optimal mapping is constructed for any consistent second-order finite difference approximation of ∇f in two dimensions. The present approach can be extended directly to nonlinear partial differential equations and three dimensions. Numerical calculations show that the truncation error obtained on the two-dimensional optimal grid is both superconvergent and reduced by several orders of magnitude in comparison with the uniform grid results for all the test examples considered.

Key words. truncation error, optimal grid, finite difference approximation

AMS subject classifications. 65M50, 65L50

PII. S1064827500378994

1. Introduction.

Grid adaptation has become one of the main subjects of interest and research because grid point distribution has a strong effect on the numerical solution accuracy and therefore directly determines the computational cost. One of the most important problems associated with grid adaptation is the fact that the concentration of grid points in regions that most influence the accuracy of the numerical solution may at the same time introduce additional error due to grid nonuniformity [1], [2], [3], [4].

Most, if not all, grid adaptation criteria are based on the equidistribution principle. Many commonly used equidistribution methods of generating adaptive grids are modifications of a technique originally proposed by Babuška and Rheinboldt [5]. As shown in [5] for one-dimensional (1D) finite element discretizations, the grid point distribution is asymptotically optimal if some error measure is equally distributed over the field. This idea has been well defined for 1D finite element methods, but a proper concept has not been developed either for other discretization techniques or problems involving multiple dimensions. One of the widely used approaches in multiple dimensions is to redistribute grid points in accordance with the arc length and the local curvature of the numerical solution [6], [7], [8], [9]. Dwyer [6] and Catherall [7] use the equidistribution principle in a 1D fashion so that grid points are redistributed along one family of fixed mesh lines. In [8], adaptivity is incorporated by using a variational approach, which can be treated as a multidimensional extension of the 1D equidistribution principle. Further development of this idea can be found in [9].

*Received by the editors October 3, 2000; accepted for publication (in revised form) September 10, 2001; published electronically January 30, 2002. This work was supported by the National Research Council and was performed while the author held a National Research Council Research Associateship Award at NASA Langley Research Center.

http://www.siam.org/journals/sisc/23-5/37899.html

†National Research Council, NASA Langley Research Center, Mail Stop 128, Hampton, VA 23681-2199 (nail@tabdemo.larc.nasa.gov).

An alternative technique is to equidistribute the local truncation error or its estimate. In [10], the optimal coordinate transformation is constructed as the solution of a constrained parameter optimization problem that minimizes a measure of the truncation error. The error measure used is a finite difference evaluation of the third derivative of the numerical solution calculated in the computational space. In [11] a cell vertex scheme is reformulated as a finite element method. This method allows construction of an a posteriori error estimate based on calculation of the finite element residual of the approximate solution, which is similar to the finite difference truncation error.

Another class of methods is based on error estimates that can be derived by evaluating the solution interpolation error [12], [13], [14], [15]. For second-order spatial finite element discretizations, this approach is reduced to estimation of the local curvature of the numerical solution. Ait-Ali-Yahia et al. [12] defined the cell edge length squared times the second derivative of the numerical solution as an a posteriori error estimate. Peraire et al. [14] determined the local principal directions of the symmetric Hessian matrix containing information about the local curvature of the solution. Then the equidistribution technique was applied separately in each principal direction. D'Azevedo [15] derived an anisotropic coordinate transformation by interpreting the Hessian matrix of the data function as a metric tensor that measures the local interpolation error. Asymptotically optimal element shapes and sizes that minimize the error/area ratio for linear triangles are generated by the transformation as an image of a regular mesh.

All the equidistribution methods mentioned above redistribute grid points in accordance with one or another error estimate obtained on a nonadaptive grid. On one hand, constructing a sufficiently accurate and reliable error estimate is very difficult, and, consequently, the grid optimality may be destroyed by poor accuracy of the error estimate. On the other hand, the error estimate norm is directly dependent on the metric coefficients. As a result, the grid adaptation itself changes the error distribution. To account for this change in the error distribution, the grid adaptation procedure based on the error equidistribution strategy should be iterated until the error estimate norm is equally distributed over the field. Note that, for moving meshes dynamically adapted to the solution, the iterative procedure should be done at each time step to attain the optimal mesh characterized by having the error equidistributed throughout the domain. One of the main disadvantages of multidimensional grid adaptation techniques is a lack of rigorously proved results showing that the grid adaptation criterion used provides global error reduction in the numerical solution.

The main idea of the present paper is to construct a two-dimensional (2D) optimal coordinate transformation based on the grid adaptation criteria proposed in [4], which minimizes the global asymptotic truncation error of an arbitrary pth-order finite difference approximation of both f_x and f_y. In contrast to most multidimensional grid adaptation criteria, the present method does not explicitly require an a posteriori error estimate, and, at the same time, the design order of approximation is increased globally by one. As a result, the same finite difference operator is superconvergent on the optimal grid. Furthermore, for second-order approximations, the proposed grid adaptation equations can be integrated analytically under the assumption that grid points move along a given family of fixed curvilinear coordinate lines. This analytical approach is extended to generate the optimal grid for conservation laws. It is shown that if grid points are redistributed in accordance with the new grid adaptation criteria, the global order of approximation of the conservation law equation is increased from 2 to 3 on the optimal grid. Another very attractive feature of the present approach is its applicability to hybrid approximations that depend on some basic properties of the solution, such as flow direction, sonic line, and others. If the metric coefficients are evaluated by the same hybrid discretization used for the differential operator, the new grid adaptation criterion remains valid throughout the computational domain regardless of points where the hybrid scheme switches from one approximation to another. Generalization of this approach to three and higher dimensions is straightforward [4]. The numerical examples considered illustrate the ability of the method and corroborate the theoretical analysis.

2. Grid adaptation criteria in two dimensions. A simply connected domain D in 2D space with Cartesian coordinates (x, y) is considered. Without loss of generality the physical domain D is assumed bounded by four boundaries, B_1, B_2, B_3, B_4. A computational domain Q is defined as a unit square in a 2D space with Cartesian coordinates (ξ, η). A differentiable one-to-one coordinate transformation between the physical and computational domains is given by

$$\begin{aligned} x &= x(\xi, \eta), \\ y &= y(\xi, \eta). \end{aligned} \tag{2.1}$$

This mapping is assumed not singular, and the Jacobian of the transformation is a strictly positive function:

$$J = x_\xi y_\eta - x_\eta y_\xi > 0. \tag{2.2}$$

The transformation (2.1) maps the boundary of Q one-to-one on the boundary of D in such a way that

$$\begin{aligned} &\xi = 0 \quad \text{at boundary } B_1, \qquad &\eta = 0 \quad \text{at boundary } B_2, \\ &\xi = 1 \quad \text{at boundary } B_3, \qquad &\eta = 1 \quad \text{at boundary } B_4. \end{aligned} \tag{2.3}$$

Nodes and cells of a grid in the physical domain D are generated by the transformation (2.1) as images of nodes and cells of a uniform rectangular grid constructed in the computational domain Q.

We consider the truncation error of the first derivative f_x of a sufficiently smooth function $f(x, y)$. The first derivative is approximated on a 2D curvilinear grid generated by the mapping (2.1). The 2D transformation of f_x is given by

$$f_x = \frac{y_\eta f_\xi - y_\xi f_\eta}{J}. \tag{2.4}$$

Approximating the ξ and η derivatives in (2.4) by some pth- and qth-order finite difference formulas, respectively, yields

$$\begin{aligned} L_h^p(v_\xi) &= \sum_{l=i-l_1}^{i+l_2} a_l v_{lk}, \\ L_h^q(v_\eta) &= \sum_{m=k-m_1}^{k+m_2} b_m v_{im}, \end{aligned} \tag{2.5}$$

where v denotes f, x, or y, $v_{lm} = v(l\Delta\xi, m\Delta\eta)$, and L_h^p and L_h^q are pth- and qth-order finite difference operators, accordingly. The indices l_1, l_2 and m_1, m_2, as well as the coefficients a_l and b_m, depend on particular approximations used for evaluating the ξ

and η derivatives. Equation (2.5) takes into account that the metric coefficients x_ξ, y_ξ and x_η, y_η are evaluated by the same finite difference operators used for calculating f_ξ and f_η, respectively.

Henceforth, it is assumed that the functions $f(\xi,\eta)$, $x(\xi,\eta)$, and $y(\xi,\eta)$ are $(p+1)$-times continuously differentiable with respect to ξ and $(q+1)$-times continuously differentiable with respect to η on Q, i.e., $f, x, y \in C^{p+1,q+1}(Q)$. Note that the present analysis remains valid if f, x, and y are piecewise $C^{p+1,q+1}(Q)$ functions. Actually, if any of the f, x, or y functions is not smooth enough (i.e., it is not $C^{p+1,q+1}(Q)$), then the physical domain D should be divided into a number of subdomains so that in each subdomain all the functions $f(\xi,\eta)$, $x(\xi,\eta)$, and $y(\xi,\eta)$ are $C^{p+1,q+1}(Q)$. Under this assumption we can expand (2.5) in a Taylor series with respect to ξ_i and η_k, respectively. Omitting the indices i and k, one can write

$$\begin{aligned}
&\sum_{l=i-l_1}^{i+l_2} a_l v_{lk} = v_\xi + C_p v_\xi^{(p+1)} \Delta\xi^p + O(\Delta\xi^{p+1}),\\
&\sum_{m=k-m_1}^{k+m_2} b_m v_{im} = v_\eta + C_q v_\eta^{(q+1)} \Delta\eta^q + O(\Delta\eta^{q+1}),
\end{aligned} \tag{2.6}$$

where

$$v_\xi^{(p+1)} = \frac{\partial^{p+1} v}{\partial \xi^{p+1}}, \quad v_\eta^{(q+1)} = \frac{\partial^{q+1} v}{\partial \eta^{q+1}}, \quad \Delta\xi = \frac{1}{I}, \quad \Delta\eta = \frac{1}{K},$$

and where C_p and C_q are constants dependent on a_l and b_m, respectively. Substituting (2.6) into (2.5) yields

$$L_h^{p,q}(f_x) = \frac{\delta_\eta y \delta_\xi f - \delta_\xi y \delta_\eta f}{\delta_\xi x \delta_\eta y - \delta_\eta x \delta_\xi y} + O(\Delta\xi^{p+1}, \Delta\eta^{q+1}), \tag{2.7}$$

where the differential operators δ_ξ and δ_η are defined by

$$\begin{aligned}
\delta_\xi &= \tfrac{\partial}{\partial\xi} + C_p \Delta\xi^p \tfrac{\partial^{p+1}}{\partial\xi^{p+1}},\\
\delta_\eta &= \tfrac{\partial}{\partial\eta} + C_q \Delta\eta^q \tfrac{\partial^{q+1}}{\partial\eta^{q+1}}.
\end{aligned} \tag{2.8}$$

As the mapping used is nonsingular $(0 < J < +\infty\ \forall \xi, \eta \in Q)$, the denominator of (2.7) can be linearized to give

$$\begin{aligned}
L_h(f_x) = \tfrac{1}{J} \Big[& y_\eta f_\xi - y_\xi f_\eta + C_p \Delta\xi^p \left(y_\eta f_\xi^{(p+1)} - y_\xi^{(p+1)} f_\eta \right)\\
& + C_q \Delta\eta^q (f_\xi y_\eta^{(q+1)} - y_\xi f_\eta^{(q+1)}) \Big]\\
& \times \left[1 - \tfrac{C_p \Delta\xi^p}{J} \left(y_\eta x_\xi^{(p+1)} - y_\xi^{(p+1)} x_\eta \right) - \tfrac{C_q \Delta\eta^q}{J} \left(x_\xi y_\eta^{(q+1)} - y_\xi x_\eta^{(q+1)} \right) \right]\\
& + O(\Delta\xi^{p+1}, \Delta\eta^{q+1}).
\end{aligned} \tag{2.9}$$

It should be emphasized that the error introduced by the linearization is of the order of $O(\Delta\xi^{2p}, \Delta\eta^{2q}, \Delta\xi^p\Delta\eta^q)$. The linearization has been performed under the following assumptions:

$$\begin{aligned}
\Delta\xi^p \left| y_\eta x_\xi^{(p+1)} - x_\eta y_\xi^{(p+1)} \right| &\ll J,\\
\Delta\eta^q \left| x_\xi y_\eta^{(q+1)} - y_\xi x_\eta^{(q+1)} \right| &\ll J,
\end{aligned}$$

which can be treated as conditions for the minimum number of grid points needed for the approximation. By multiplying out the terms in the square brackets and neglecting higher-order terms, we can write the leading truncation error term $T_{p,q}$ as follows:

$$\begin{aligned} T_{p,q}(\xi,\eta) = \tfrac{1}{J}\Big\{ & C_p \Delta\xi^p \Big[y_\eta f_\xi^{(p+1)} - y_\xi^{(p+1)} f_\eta - f_x \Big(y_\eta x_\xi^{(p+1)} - x_\eta y_\xi^{(p+1)} \Big) \Big] \\ & + C_q \Delta\eta^q \Big[f_\xi y_\eta^{(q+1)} - y_\xi f_\eta^{(q+1)} - f_x \Big(x_\xi y_\eta^{(q+1)} - y_\xi x_\eta^{(q+1)} \Big) \Big] \Big\}. \end{aligned} \tag{2.10}$$

The truncation error $T_{p,q}$ consists of two parts: one part arises from the approximation of f_ξ and f_η, and the other occurs due to the evaluation of the metric coefficients x_ξ, y_ξ, x_η, and y_η. It should be emphasized that any grid adaptation based on minimization or equidistribution of the first part of the truncation error alone is not sufficient because the second part of the truncation error may drastically increase in regions where the metric coefficients change rapidly. In other words, any inconsistent grid adaptation transfers the error from the first part of the truncation error to that of the second, and vice versa. To minimize both parts of the truncation error simultaneously we impose the following restriction on the coordinate mapping. From (2.10) it follows that if absolute value of the first expression in the square brackets is less than $O(\Delta\xi)$, and absolute value of the second expression is less than $O(\Delta\eta)$, then the global truncation error is $O(\Delta\xi^{p+1}, \Delta\eta^{q+1})$ rather than $O(\Delta\xi^p, \Delta\eta^q)$. Thus, to increase the order of the finite difference approximation (2.7) globally by one, grid points should be redistributed so that the following equations hold:

$$\begin{aligned} J\Big(y_\eta f_\xi^{(p+1)} - y_\xi^{(p+1)} f_\eta \Big) &= (y_\eta f_\xi - y_\xi f_\eta)\Big(y_\eta x_\xi^{(p+1)} - x_\eta y_\xi^{(p+1)} \Big) + O(\Delta\xi) J^2 \\ J\Big(f_\xi y_\eta^{(q+1)} - y_\xi f_\eta^{(q+1)} \Big) &= (y_\eta f_\xi - y_\xi f_\eta)\Big(x_\xi y_\eta^{(q+1)} - y_\xi x_\eta^{(q+1)} \Big) + O(\Delta\eta) J^2. \end{aligned} \tag{2.11}$$

By removing the parentheses and rearranging corresponding terms, we reduce (2.11) to

$$\begin{aligned} y_\eta \Big[f_\xi^{(p+1)} - f_y y_\xi^{(p+1)} - f_x x_\xi^{(p+1)} \Big] &= O(\Delta\xi) J, \\ -y_\xi \Big[f_\eta^{(q+1)} - f_y y_\eta^{(q+1)} - f_x x_\eta^{(q+1)} \Big] &= O(\Delta\eta) J. \end{aligned} \tag{2.12}$$

The equations in (2.12) can be considered optimal grid generation equations in the sense of minimizing the leading truncation error term of the first derivative f_x. The optimal grid adaptation criteria for f_y, which is

$$f_y = \frac{-x_\eta f_\xi + x_\xi f_\eta}{J}, \tag{2.13}$$

can be derived in the same fashion as has been done for f_x. Actually, using the same approximations (2.5) for all the ξ and η derivatives in (2.13), and linearizing the corresponding leading truncation error term, the grid adaptation criteria for f_y can be obtained in the following form:

$$\begin{aligned} -x_\eta \Big[f_\xi^{(p+1)} - f_y y_\xi^{(p+1)} - f_x x_\xi^{(p+1)} \Big] &= O(\Delta\xi) J, \\ x_\xi \Big[f_\eta^{(q+1)} - f_y y_\eta^{(q+1)} - f_x x_\eta^{(q+1)} \Big] &= O(\Delta\eta) J. \end{aligned} \tag{2.14}$$

The presence of the $O(\Delta\xi)$ and $O(\Delta\eta)$ terms on the right-hand side of (2.12) and (2.14) indicates that the grid adaptation criteria are rather stable under perturbations of the optimal grid.

At the same time, a reduction of the $O(\Delta\xi)$ and $O(\Delta\eta)$ terms in (2.12) and (2.14) decreases the corresponding truncation errors on the optimal grid. Therefore, we can neglect these terms so that (2.12) and (2.14) become identical and can be written as

$$\begin{aligned} f_\xi^{(p+1)} - f_y y_\xi^{(p+1)} - f_x x_\xi^{(p+1)} &= 0, \\ f_\eta^{(q+1)} - f_y y_\eta^{(q+1)} - f_x x_\eta^{(q+1)} &= 0. \end{aligned} \tag{2.15}$$

From the above consideration it follows that if we can construct the mapping which satisfies (2.15), then the leading truncation error term of the gradient of $f(x, y)$ vanishes on the same optimal grid. Because (2.12) and (2.14) have similar properties, hereafter only (2.12) is considered.

Note that if $y_\xi = 0$ in the entire computational domain, (2.12) reduces to

$$x_\xi f_\xi^{(p+1)} - f_\xi x_\xi^{(p+1)} = O(\Delta\xi) x_\xi^2, \tag{2.16}$$

which can be treated as a 1D analogue of (2.12) in the ξ coordinate. The main properties of (2.16) are analyzed in [4]. Another very useful characteristic of the optimal mapping is that the equations in (2.12) are invariant with respect to both translation and stretching of the x, y and ξ, η coordinates.

Remark 2.1. It should be noted that the equations in (2.12) are not a system of equations and can be considered separately. If improving the accuracy with respect to the ξ coordinate alone is necessary, a grid must be generated such that only the first equation in (2.12) holds. However, if increasing the order of approximation of f_x by one in both the ξ and η coordinates simultaneously is desirable, then the grid must obey the system of equations (2.12).

Remark 2.2. A characteristic feature of the equations in (2.12) is that they do not depend on the coefficients C_p and C_q. The reason is that the metric coefficients and the first derivatives f_ξ and f_η are approximated by using the same finite difference formulas (2.5). Consequently, if in each spatial direction the metric coefficients and the first derivatives of $f(\xi, \eta)$ are evaluated consistently by some hybrid finite difference operators, then the grid adaptation criteria (2.12) can be applied in the whole computational domain regardless of points where the hybrid scheme switches from one approximation to another. If the identical numerical approximation is the case, the optimal grid point distribution is generic because it depends only on the order of approximation and is completely independent of the particular finite difference formula used.

Remark 2.3. As follows from the analysis presented in [4], the grid adaptation equation does not guarantee that the coordinate mapping obtained as the solution of (2.12) is not singular, i.e., $0 < J < +\infty$. Because (2.12) converts to (2.16), if the dimension of the space is decreased by one, the same singularity may occur in two dimensions as well. In other words, the equations in (2.12) do not ensure existence and uniqueness of the one-to-one coordinate transformation. A way to overcome this problem in the case of second-order approximations will be discussed in the next section.

The equations in (2.12) must be closed by corresponding boundary conditions. These equations can be shown to be pth-order partial differential equations (PDEs), and, therefore, p boundary conditions should be imposed at each pair of the opposite boundaries (i.e., $\xi = 0$ and $\xi = 1$, $\eta = 0$ and $\eta = 1$) to find the unique solution.

However, at each individual boundary, only one boundary condition is available. For example, the boundary conditions in the ξ coordinate are

$$\xi(x,y)=0, \quad \xi(x,y)=1. \tag{2.17}$$

In other words, the equations in (2.12) are not closed. The situation becomes even more uncertain when only one of the grid adaptation criteria (2.12) is used. However, this uncertainty yields additional degrees of freedom, and at the same time it is conceivable that more than one optimal grid exists to satisfy the criteria (2.12). From this standpoint, the equations in (2.12) should be treated as the grid adaptation criteria rather than as the optimal grid generation equations [4].

3. Optimal grids for second-order approximations. If both f_ξ, f_η and the corresponding metric coefficients x_ξ, y_ξ and x_η, y_η are approximated by some second-order formulas, the grid adaptation criteria (2.15) written for $p=q=2$ become

$$\begin{aligned} f_{\xi\xi\xi} &= f_x x_{\xi\xi\xi} + f_y y_{\xi\xi\xi},\\ f_{\eta\eta\eta} &= f_x x_{\eta\eta\eta} + f_y y_{\eta\eta\eta}. \end{aligned} \tag{3.1}$$

With

$$\begin{aligned} \frac{\partial}{\partial\xi} &= x_\xi \frac{\partial}{\partial x} + y_\xi \frac{\partial}{\partial y},\\ \frac{\partial}{\partial\eta} &= x_\eta \frac{\partial}{\partial x} + y_\eta \frac{\partial}{\partial y}, \end{aligned} \tag{3.2}$$

$f_{\xi\xi\xi}$ can be written in the physical space as follows:

$$\begin{aligned} f_{\xi\xi\xi} = &\left[x_\xi \frac{\partial}{\partial x} + y_\xi \frac{\partial}{\partial y}\right]^3 f = f_{xxx}x_\xi^3 + 3f_{xxy}x_\xi^2 y_\xi + 3f_{xyy}x_\xi y_\xi^2 + f_{yyy}y_\xi^3\\ &+ 3f_{xx}x_\xi x_{\xi\xi} + 3f_{xy}(x_{\xi\xi}y_\xi + y_{\xi\xi}x_\xi) + 3f_{yy}y_\xi y_{\xi\xi} + f_x x_{\xi\xi\xi} + f_y y_{\xi\xi\xi}. \end{aligned}$$

Calculating $f_{\eta\eta\eta}$ in a similar fashion and substituting these expressions into (3.1) yields

$$\begin{aligned} &3(f_{xx}x_\xi + f_{xy}y_\xi)x_{\xi\xi} + 3(f_{xy}x_\xi + f_{yy}y_\xi)y_{\xi\xi} + f_{xxx}x_\xi^3 + 3f_{xxy}x_\xi^2 y_\xi\\ &\quad +3f_{xyy}y_\xi^2 x_\xi + f_{yyy}y_\xi^3 = 0,\\ &3(f_{xx}x_\eta + f_{xy}y_\eta)x_{\eta\eta} + 3(f_{xy}x_\eta + f_{yy}y_\eta)y_{\eta\eta} + f_{xxx}x_\eta^3 + 3f_{xxy}x_\eta^2 y_\eta\\ &\quad + 3f_{xyy}y_\eta^2 x_\eta + f_{yyy}y_\eta^3 = 0. \end{aligned} \tag{3.3}$$

With (3.2), the equations in (3.3) are simplified to

$$\begin{aligned} &3f_{x\xi}x_{\xi\xi} + 3f_{y\xi}y_{\xi\xi} + f_{xx\xi}x_\xi^2 + 2f_{xy\xi}x_\xi y_\xi + f_{yy\xi}y_\xi^2 = 0\\ &3f_{x\eta}x_{\eta\eta} + 3f_{y\eta}y_{\eta\eta} + f_{xx\eta}x_\eta^2 + 2f_{xy\eta}x_\eta y_\eta + f_{yy\eta}y_\eta^2 = 0. \end{aligned} \tag{3.4}$$

Taking into account that the function $f(x,y)$ is given, the first equation in (3.4) contains only the derivatives with respect to ξ, while the second equation contains only the derivatives with respect to η. This property leads to the following integration procedure for (3.4). To integrate the first equation in (3.4) with respect to ξ, we assume that a family of coordinate lines $\eta = \text{const}$ is known. Therefore, at each point of the physical space, we can define a function $\alpha(\xi,\eta)$ as follows:

$$\alpha(\xi,\eta) = \tan\phi = \frac{y_\xi}{x_\xi}, \tag{3.5}$$

where ϕ is a known angle between a mesh line $\eta = \text{const}$ and the x-axis. By using the above equation, we can write the $y_{\xi\xi}$ derivative as

$$y_{\xi\xi} = (\alpha x_\xi)_\xi = \alpha_\xi x_\xi + \alpha x_{\xi\xi}. \tag{3.6}$$

Substituting (3.5) and (3.6) into (3.4) yields

$$3(f_{x\xi} + \alpha f_{y\xi})x_{\xi\xi} + 3\alpha_\xi f_{y\xi} x_\xi + x_\xi^2 \left(f_{xx\xi} + 2f_{xy\xi}\alpha + \alpha^2 f_{yy\xi}\right) = 0. \tag{3.7}$$

Since

$$\begin{aligned} f_{x\xi} &= f_{xx}x_\xi + f_{xy}y_\xi = (f_{xx} + \alpha f_{xy})x_\xi, \\ f_{y\xi} &= f_{xy}x_\xi + f_{yy}y_\xi = (f_{xy} + \alpha f_{yy})x_\xi, \end{aligned}$$

(3.7) can be reduced to

$$\begin{aligned} &x_\xi \left\{3\left(f_{xx} + 2\alpha f_{xy} + f_{yy}\alpha^2\right) x_{\xi\xi} \right. \\ &\left. + \, x_\xi \left[(f_{xx} + 2f_{xy}\alpha + f_{yy}\alpha^2)_\xi + \alpha_\xi(f_{xy} + \alpha f_{yy})\right]\right\} = 0. \end{aligned} \tag{3.8}$$

Without loss of generality, we assume that $x_\xi \neq 0$. Actually, if $x_\xi = 0$, then $y_\xi \neq 0$; otherwise the Jacobian of the transformation would be degenerated. In this case, the function α given by (3.5) can be redefined as x_ξ/y_ξ, and (3.8) can be rewritten in terms of y_ξ and $y_{\xi\xi}$ as

$$\begin{aligned} &y_\xi \left\{3\left(f_{xx}\alpha^2 + 2\alpha f_{xy} + f_{yy}\right) y_{\xi\xi} \right. \\ &\left. + \, y_\xi \left[(f_{xx}\alpha^2 + 2f_{xy}\alpha + f_{yy})_\xi + \alpha_\xi(f_{xx}\alpha + f_{xy})\right]\right\} = 0. \end{aligned} \tag{3.9}$$

Because the main properties of (3.8) and (3.9) are identical, hereafter only equation (3.8) is considered. Equation (3.8) can readily be integrated to give

$$x_\xi = C_1 \left(f_{xx} + 2f_{xy}\alpha + f_{yy}\alpha^2\right)^{-1/3} \exp\left[-\frac{1}{3}\int_0^\xi \frac{\alpha_t(f_{xy} + \alpha f_{yy})}{f_{xx} + 2f_{xy}\alpha + f_{yy}\alpha^2} dt\right], \tag{3.10}$$

where C_1 is a constant of the integration. Equation (3.10) can be treated as an analytical optimal grid for an arbitrary second-order approximation of ∇f in the sense of minimization of the leading truncation error term with respect to the ξ coordinate. If a grid is generated in accordance with the optimal mapping (3.10), the leading truncation error term in ξ is equal to zero in the entire physical domain D, and the global order of accuracy with respect to ξ is increased from 2 to 3.

A similar approach can be applied to construct the optimal grid in the η coordinate. Actually, assuming that a family of grid lines $\xi = \text{const}$ is given, we introduce a function $\beta(\xi, \eta)$ and derive the optimal grid point distribution in η in a similar manner as has just been done for ξ. Thus,

$$y_\eta = C_2 \left(f_{xx}\beta^2 + 2f_{xy}\beta + f_{yy}\right)^{-1/3} \exp\left[-\frac{1}{3}\int_0^\eta \frac{\beta_t(f_{xx}\beta + f_{xy})}{f_{xx}\beta^2 + 2f_{xy}\beta + f_{yy}} dt\right], \tag{3.11}$$

where

$$\beta(\xi, \eta) = \tan\psi = \frac{x_\eta}{y_\eta}.$$

The mapping given by (3.11) provides that the leading truncation error term with respect to η vanishes on the optimal grid. Note that in the 1D case, (3.10) and (3.11) are simplified as follows:

$$\begin{aligned} x_\xi &= C_1(f_{xx})^{-1/3}, \\ y_\eta &= C_2(f_{yy})^{-1/3}. \end{aligned} \tag{3.12}$$

This result is identical to that obtained earlier in one dimension [4].

As mentioned in Remark 2.3, (3.10) does not provide that the Jacobian of the transformation J is positive in the entire physical domain. Therefore, the optimal mapping (3.10) is modified as follows. Because the optimal grid (3.10) has been obtained under the assumption that mesh lines $\eta =$ const do not intersect each other, to provide that $J > 0$ it is necessary that an arc length increment Δs measured along a mesh line $\eta =$ const must be strictly positive. The arc length increment in ξ can easily be calculated to yield

$$\Delta s = \sqrt{x_\xi^2 + y_\xi^2}\Delta\xi = |x_\xi|\sqrt{1+\alpha^2}\Delta\xi. \tag{3.13}$$

From (3.13) it follows that Δs is strictly positive in D if the following constraint is imposed on α and, consequently, on the metric coefficients:

$$F(\xi,\eta) = f_{xx} + 2f_{xy}\alpha + f_{yy}\alpha^2 \neq 0. \tag{3.14}$$

The above constraint can be incorporated into numerical calculations by modifying the function F as

$$\tilde{F}(\xi,\eta) = \begin{cases} |F(\xi,\eta)|\,, & |F(\xi,\eta)| \geq \epsilon, \\ \frac{F(\xi,\eta)^2+\epsilon^2}{2\epsilon}\,, & |F(\xi,\eta)| < \epsilon. \end{cases} \tag{3.15}$$

Equation (3.15) is a necessary condition for positiveness and nonsingularity of the Jacobian of the transformation and, consequently, for existence of the one-to-one optimal mapping. Furthermore, (3.15) provides smoothness of the metric coefficients x_ξ and y_ξ. Note that (3.15) is not a sufficient condition. The optimal grid point distribution written in terms of the normalized arc length s is

$$s(\xi,\eta) = \frac{\int_0^\xi \tilde{F}^{-1/3}\sqrt{1+\alpha^2}\exp\left[-\frac{1}{3}\int_0^v \frac{\alpha_t(f_{xy}+f_{yy}\alpha)}{\tilde{F}}dt\right]dv}{\int_0^1 \tilde{F}^{-1/3}\sqrt{1+\alpha^2}\exp\left[-\frac{1}{3}\int_0^v \frac{\alpha_t(f_{xy}+f_{yy}\alpha)}{\tilde{F}}dt\right]dv}. \tag{3.16}$$

In numerical calculations, the second derivatives f_{xx}, f_{yy}, and f_{xy} are approximated numerically and, therefore, depend on the location of grid points in the physical domain. To find the optimal grid point distribution given by (3.16), the following iteration procedure is applied. At each grid point, the approximation of the second derivatives can be updated when the new grid point distribution is found. In its turn, the updated second derivatives generate a new optimal grid (3.16).

Let us show that this iteration technique is equivalent to the Picard iteration method (see, e.g., [16]). Actually, (3.16) can be interpreted as the following integral equation for the optimal mapping $s(\xi,\eta)$:

$$s(\xi,\eta) = \int_0^\xi G(t,s)dt, \tag{3.17}$$

where $\eta = \text{const}$. Taking into account the fact that $\tilde{F} \in C^1(Q)$, it follows that the function G is continuously differentiable $\forall s \in [0,1]$ and, consequently, satisfies the Lipschitz condition. Hence, the integral operator (3.17) is contractive on $[0,1]$ and maps $[0,1]$ into itself. Therefore, the iteration procedure based on (3.17) converges uniformly to the optimal mapping $s(\xi,\eta)$. The x and y coordinates of the optimal grid in ξ are obtained by 1D interpolation of s performed along each line $\eta = \text{const}$. The optimal grid in η can be constructed in a similar manner.

Despite the optimal grid point distribution (3.16) being derived under the assumption that a family of mesh lines $\eta = \text{const}$ is prescribed, this relation can be used directly along the boundaries $\eta = 0$ and $\eta = 1$ which are given a priori. Actually, the unknown function $\alpha(\xi,\eta)$ in (3.16) can always be determined on the boundaries $\eta = 0$ and $\eta = 1$ because at each boundary point we can calculate the boundary mesh line slope ϕ (3.5). In other words, if the function $f(x,y)$ and the physical domain boundary ∂D are given, the optimal grid point distribution along the boundaries can be found by using (3.10) and (3.11). In these equations, the unknown functions α and β are tangents of angles between the boundary mesh lines $\xi(x,y) = 0$, $\xi(x,y) = 1$ and $x = 0$; and $\eta(x,y) = 0$, $\eta(x,y) = 1$ and $y = 0$, respectively. The functions $\alpha(\xi,\eta)$ and $\beta(\xi,\eta)$ should satisfy the following equations:

$$\begin{aligned} x_{\xi\eta} &= x_{\eta\xi}, \\ y_{\xi\eta} &= y_{\eta\xi}, \end{aligned} \tag{3.18}$$

where the metric coefficients x_ξ, y_ξ and x_η, y_η are given by (3.10) and (3.11), respectively. Substituting (3.10) and (3.11) into (3.18) results in a very complicated system of differential equations for the functions α and β; these equations cannot be integrated analytically.

The above consideration gives rise to the following iterative grid generation procedure. First, the optimal grid point distribution on the boundaries is calculated by using (3.10) and (3.11). Then any grid generation technique (e.g., a transfinite interpolation) and the optimal boundary point distribution are used to generate an initial grid. This initial grid, together with an interpolation procedure, serves to define mesh lines along which the points will move during adaptation. After that, the $\eta = \text{const}$ grid lines are fixed, and grid points are redistributed along these lines in an optimal manner by using (3.10). In turn, this redistribution generates new coordinate lines $\xi = \text{const}$ so that we can redistribute grid points along this family of lines in accordance with (3.11). This sweep alters the grid point distribution along the $\xi = \text{const}$ lines and generates new mesh lines $\eta = \text{const}$. The grid generation procedure should be iterated until convergence. We should stress, however, that there is no proof that this iteration method converges. Moreover, as will be shown later, the solution of the system of equations (2.12) is not unique. This nonuniqueness gives an indication that for some functions $f(x,y)$, the system of equations (2.12) with the boundary conditions (2.3) is not well posed.

One interesting property of (3.1) is that the first equation in the system is invariant with respect to the following coordinate transformation:

$$\begin{aligned} \tilde{\xi} &= \xi, \\ \tilde{\eta} &= v_1(\eta), \end{aligned}$$

while the second equation is invariant with respect to a similar transformation:

$$\begin{aligned} \tilde{\xi} &= v_2(\xi), \\ \tilde{\eta} &= \eta, \end{aligned}$$

where v_1 and v_2 are smooth functions such that

$$v_i(0) = 0, \quad v_i(1) = 1, \quad i = \overline{1,2}, \quad \frac{dv_1}{d\eta} > 0, \quad \frac{dv_2}{d\xi} > 0.$$

This invariance means that if the optimal grid is constructed so that only the first equation in (3.1) holds, a family of the optimal grids (3.2) exists on which the leading truncation error term with respect to ξ vanishes. In other words, if only one of the equations in (3.1) is used for grid generation, the optimal grid is not unique. At the same time, if the physical domain D is bounded by lines $y = a_0$ and $y = a_1$ corresponding to the boundaries $\eta = 0$ and $\eta = 1$, then a family of the optimal grids satisfying the system of equations (2.12) is

$$\begin{aligned} x &= x(\xi, \eta), \\ y &= v(\eta), \end{aligned} \tag{3.19}$$

where $x(\xi, \eta)$ is defined by (3.16) and the function v meets the following conditions:

$$v(0) = a_0, \quad v(1) = a_1, \quad v'_\eta > 0 \ \forall \eta \in [0, 1]. \tag{3.20}$$

In contrast to the one-parameter family of the solutions $\tilde{\xi} = \xi$ and $\tilde{\eta} = v_1(\eta)$, which satisfies just the first equation in (3.1), (3.19) is the solution of the system (2.12).

Remark 3.1. Although the present analysis has been performed for simply connected domains, this approach can be extended to multiply connected domains. It can be done by dividing the physical domain into a number of simply connected subdomains with smooth boundaries so that $f(\xi, \eta)$, $x(\xi, \eta)$, and $y(\xi, \eta)$ are $C^{p+1,q+1}$ functions in each subdomain. After that, the present method can be used separately in each subdomain.

Remark 3.2. It should be emphasized that the above approach can be extended directly to three dimensions. 3D grid adaptation criteria analogous to (2.15) have been derived in [4]. These grid generation equations, minimizing the leading truncation error term of ∇f in three dimensions, are

$$\begin{aligned} f_\xi^{(p+1)} - f_x x_\xi^{(p+1)} - f_y y_\xi^{(p+1)} - f_z z_\xi^{(p+1)} &= 0, \\ f_\eta^{(q+1)} - f_x x_\eta^{(q+1)} - f_y y_\eta^{(q+1)} - f_z z_\eta^{(q+1)} &= 0, \\ f_\zeta^{(r+1)} - f_x x_\zeta^{(r+1)} - f_y y_\zeta^{(r+1)} - f_z z_\zeta^{(r+1)} &= 0. \end{aligned} \tag{3.21}$$

Each of the three 3D grid generation equations in (3.21) contains derivatives with respect to just one of the ξ, η, or ζ coordinates, respectively. Therefore, assuming that $x_\xi \neq 0$, and introducing $\alpha_1 = y_\xi / x_\xi$ and $\alpha_2 = z_\xi / x_\xi$, we can integrate the 3D optimal grid generation equation corresponding to the coordinate ξ in the same fashion as in the 2D case.

4. Optimal grids for nonlinear PDEs. The optimal grid generation procedure developed in the foregoing section can be extended directly to a nonlinear PDE. Let us consider a nonlinear scalar equation written in the following form:

$$L(f_x, g_y, f, g, u, x, y) = 0, \tag{4.1}$$

where L is a smooth nonlinear function of its arguments and f and g are nonlinear functions of x, y, and u. Note that (4.1) is solved with respect to u; i.e., this equation is strongly nonlinear.

To construct a pth-order in ξ and qth-order in η approximation of (4.1), we transfer the x and y derivatives into the computational space (ξ, η) and use some pth- and qth-order accurate schemes to evaluate the ξ and η derivatives, accordingly. The other quantities in (4.1) are evaluated exactly. The truncation error of such an approximation is

$$T_{p,q} = L_h(f_x^{\text{ex}}, g_y^{\text{ex}}, f^{\text{ex}}, g^{\text{ex}}, u^{\text{ex}}, x, y) - L(f_x^{\text{ex}}, g_y^{\text{ex}}, f^{\text{ex}}, g^{\text{ex}}, u^{\text{ex}}, x, y),$$

where L_h is a finite difference operator approximating (4.1) and the superscript "ex" denotes the exact solution of (4.1). Expanding the right-hand side of the above equation in a Taylor series with respect to the exact solution and omitting the superscript "ex" yields

$$T_{p,q} = \frac{\partial L}{\partial f_x}\left[(f_x)_h - f_x\right] + \frac{\partial L}{\partial g_y}\left[(g_y)_h - g_y\right] = \frac{\partial L}{\partial f_x} T^f_{p,q} + \frac{\partial L}{\partial g_y} T^g_{p,q}, \tag{4.2}$$

where $T^f_{p,q}$ and $T^g_{p,q}$ are truncation errors of the first derivatives f_x and g_y, respectively. With (2.10) and a similar expression for $T^g_{p,q}$, the leading truncation error term of (4.1) becomes

$$\begin{aligned} T_{p,q}(\xi,\eta) = {} & \tfrac{C_p \Delta\xi^p}{J^2} \left[\tfrac{\partial L}{\partial f_x} \left(y_\eta f_\xi^{(p+1)} - x_\xi^{(p+1)} f_x y_\eta - y_\xi^{(p+1)} f_\eta y_\eta \right) \right. \\ & \left. + \tfrac{\partial L}{\partial g_y} \left(-x_\eta g_\xi^{(p+1)} + x_\xi^{(p+1)} x_\eta g_x + y_\xi^{(p+1)} x_\eta g_y \right) \right] \\ & + \tfrac{C_q \Delta\eta^q}{J^2} \left[\tfrac{\partial L}{\partial f_x} \left(-y_\xi f_\eta^{(q+1)} + x_\eta^{(q+1)} f_x y_\xi + y_\eta^{(q+1)} f_y y_\xi \right) \right. \\ & \left. + \tfrac{\partial L}{\partial g_y} \left(x_\xi g_\eta^{(q+1)} - x_\eta^{(q+1)} x_\xi g_x - y_\eta^{(q+1)} x_\xi g_y \right) \right]. \end{aligned} \tag{4.3}$$

The above equation has been derived by assuming that $\partial L/\partial f_x$ and $\partial L/\partial g_y$ are functions of f, g, u, x, and y only, and do not depend on f_x and g_y. It is also assumed that the metric coefficients are evaluated by the same finite difference operator used for approximating the corresponding first derivatives f_ξ, g_ξ and f_η, g_η. Equation (4.3) indicates that if a grid is generated so that the first term in the square brackets is of the order of $O(\Delta\xi)$, while the second term is of the order of $O(\Delta\eta)$, then the global orders of approximations with respect to ξ and η are increased from p and q to $p+1$ and $q+1$, respectively. Thus, in the sense of minimizing the asymptotic truncation error of (4.1), the optimal grid should satisfy the following criteria:

$$\begin{aligned} \tfrac{\partial L}{\partial f_x} y_\eta f_\xi^{(p+1)} - \tfrac{\partial L}{\partial g_y} x_\eta g_\xi^{(p+1)} &= x_\xi^{(p+1)} \left(\tfrac{\partial L}{\partial f_x} f_x y_\eta - \tfrac{\partial L}{\partial g_y} g_x x_\eta \right) \\ & \quad + y_\xi^{(p+1)} \left(\tfrac{\partial L}{\partial f_x} f_y y_\eta - \tfrac{\partial L}{\partial g_y} g_y x_\eta \right) + O(\Delta\xi) J^2 \\ \tfrac{\partial L}{\partial g_y} x_\xi g_\eta^{(q+1)} - \tfrac{\partial L}{\partial f_x} y_\xi f_\eta^{(q+1)} &= x_\eta^{(q+1)} \left(\tfrac{\partial L}{\partial g_y} g_x x_\xi - \tfrac{\partial L}{\partial f_x} f_x y_\xi \right) \\ & \quad + y_\eta^{(q+1)} \left(\tfrac{\partial L}{\partial g_y} g_y x_\xi - \tfrac{\partial L}{\partial f_x} f_y y_\xi \right) + O(\Delta\eta) J^2. \end{aligned} \tag{4.4}$$

Because reduction of the $O(\Delta\xi)$ and $O(\Delta\eta)$ terms increases the accuracy of the approximation on the optimal grid, these terms are neglected.

In the case where the governing equation (4.1) can be written in conservation law form,

$$\frac{\partial f(u)}{\partial x} + \frac{\partial g(u)}{\partial y} = 0, \tag{4.5}$$

and some second-order schemes are used to evaluate both the ξ and η derivatives, (4.4) is reduced to

$$\begin{aligned} y_\eta f_{\xi\xi\xi} - x_\eta g_{\xi\xi\xi} = &\quad x_{\xi\xi\xi}(f_x y_\eta - x_\eta g_x) + y_{\xi\xi\xi}(f_y y_\eta - x_\eta g_y), \\ -y_\xi f_{\eta\eta\eta} + x_\xi g_{\eta\eta\eta} = &\quad x_{\eta\eta\eta}(x_\xi g_x - f_x y_\xi) + y_{\eta\eta\eta}(x_\xi g_y - f_y y_\xi). \end{aligned} \tag{4.6}$$

It should be emphasized that the conservation law (4.5) is a nonlinear PDE with respect to u because we assume the functions f and g are nonlinear functions of u. The equations in (4.6) take into account that for the conservation law (4.5), the derivatives $\partial L/\partial f_x$ and $\partial L/\partial g_y$ in (4.6) are equal to unity identically. Note that the second equation in (4.6) converts to the first equation by exchanging ξ and η for each other. Therefore, we shall consider just the first equation, and the same results will remain valid for the second equation by similarly exchanging ξ and η.

The equations in (4.6) can be integrated in the same fashion as we have integrated (3.1). Assuming that one family of mesh lines (e.g., $\eta =$ const) is given, we redistribute grid points along these lines in such a way that the first equation in (4.6) is satisfied in the whole computational domain. With (3.2), the first equation in (4.6) is transformed to

$$\begin{aligned} 3(y_\eta f_{x\xi} - x_\eta g_{x\xi})x_{\xi\xi} + 3(f_{y\xi} y_\eta - g_{y\xi} x_\eta)y_{\xi\xi} + (y_\eta f_{xx\xi} - x_\eta g_{xx\xi})x_\xi^2 \\ + 2(f_{xy\xi} y_\eta - x_\eta g_{xy\xi})x_\xi y_\xi + (f_{yy\xi} y_\eta - x_\eta g_{yy\xi})y_\xi^2 = 0. \end{aligned} \tag{4.7}$$

Introducing the function α given by (3.5) and using the relations (3.6) yields

$$3(F y_\eta - G x_\eta) + x_\xi \left\{F_\xi y_\eta - G_\xi x_\eta + \alpha_\xi \left[y_\eta \left(f_{xy} + \alpha f_{yy}\right) - x_\eta \left(g_{xy} + \alpha g_{yy}\right)\right]\right\} = 0, \tag{4.8}$$

where F is given by (3.14) and $G = g_{xx} + 2g_{xy}\alpha + g_{yy}\alpha^2$. Equation (4.8) derived under the assumption that $x_\xi \neq 0$ can be further simplified if we use the function β defined by (3.11). Thus,

$$y_\eta \left\{3(F - G\beta)x_{\xi\xi} + x_\xi \left[F_\xi - G_\xi \beta + \alpha_\xi \left(f_{xy} + \alpha f_{yy} - \beta \left[g_{xy} + \alpha g_{yy}\right]\right)\right]\right\} = 0. \tag{4.9}$$

With $y_\eta \neq 0$, (4.9) can be integrated to give

$$x_\xi = C(F - \beta G)^{-1/3} \exp\left[-\frac{1}{3}\int_0^\xi \frac{\beta_t G + \alpha_t \left(f_{xy} + \alpha f_{yy} - \beta[g_{xy} + \alpha g_{yy}]\right)}{F - \beta G} dt\right]. \tag{4.10}$$

Even though (4.10) depends on both α and β, basic properties of this optimal grid point distribution are similar to those obtained for the optimal mapping (3.10). By analogy with (3.10) and (3.13)–(3.15), the necessary condition that the Jacobian of the transformation is positive in the entire computational domain is satisfied if function $E = F - \beta G$ is modified as follows:

$$\tilde{E} = \begin{cases} |F - \beta G|\ , & |F - \beta G| \geq \epsilon, \\ \frac{(F-\beta G)^2 + \epsilon^2}{2\epsilon}\ , & |F - \beta G| < \epsilon, \end{cases} \tag{4.11}$$

where ϵ is a small positive parameter. As follows from (4.10), if a grid is constructed such that (4.10) holds, then for the conservation law equation (4.5) the global order of approximation with respect to ξ is increased by one.

In contrast to the grid generation procedure developed for ∇f, (4.10) cannot be integrated separately for each mesh line $\eta = \text{const}$ because the function $\beta(\xi, \eta)$ is involved in the formula. The problem can be overcome by taking the unknown function β from the previous iteration so that grid points are redistributed at each mesh line $\eta = \text{const}$ independently. After that, values of β are updated, and the iteration procedure should be repeated until convergence.

Despite the fact that the optimal grid (4.10) has been derived under the condition that one family of coordinate lines is given a priori, this analytical formula can be directly used for calculating boundary layer problems. Assuming that the solution of (4.5) is sufficiently smooth and has a boundary layer in ξ, the unidirectional grid adaptation procedure based on (4.10) can be applied with grid points constrained to move along the $\eta = \text{const}$ family of fixed curvilinear coordinate lines. This family of coordinate lines can be constructed, e.g., by using the conformal mapping that ensures existence and uniqueness of one-to-one coordinate transformations. The coordinate lines $\eta = \text{const}$ obtained this way are used to generate the optimal grid in ξ by using the formula (4.10). This grid adaptation procedure provides that the global order of approximation with respect to ξ is increased from 2 to 3, while the order of approximation in η remains the same. Because of the boundary layer, the asymptotic truncation error in the ξ coordinate is much greater than that in η. Therefore, this unidirectional grid adaptation procedure minimizes the most troublesome part of the global truncation error caused by the boundary layer.

5. Results and discussion. Several 2D test examples are investigated to validate the applicability and efficiency of the new method. For each test example, four series of calculations have been executed on different grids with the same number of grid points. The first calculation is done on a uniform grid generated by a transfinite interpolation of the boundary nodes uniformly redistributed along the boundaries. The second is performed on the optimal grid obtained as the analytical solution of (2.12). The third uses the optimal grid (3.16) generated numerically by approximating the second derivatives f_{xx}, f_{xy}, and f_{yy} with second-order central differences in the computational domain. The integrals in (3.16) are computed with trapezoidal rule integration. The fourth calculation is also executed on the uniform grid; however, instead of a second-order approximation, a third-order accurate discretization is applied to calculate both the ξ and η derivatives in (2.4). At each boundary, one-sided second-order differences are used. In order to estimate the method's accuracy, the L_2 and L_∞ norms of the truncation error of different second-order finite difference approximations of f_x and f_y are calculated on successively refined grids. Note that evaluation of the L_2 integral norm in the computational space differs from that calculated in the physical space by the Jacobian factor. Unlike the L_2 norm, the L_∞ norm is generic in this sense because it is the same in both the physical and transformed spaces.

The first test example is a second-order approximation of the first derivative f_x of a function $f(x, y) = (x - y)^m$ on a unit square $[0, 1] \times [0, 1]$. The parameter m has been set at 20, which results in the function having a pronounced boundary layer of width $O(1/m)$ near points $(0, 1)$ and $(1, 0)$. Isolines of this function are shown in Figure 5.1. A backward second-order approximation,

$$\left(\frac{\partial f}{\partial \xi}\right)_i = \frac{-3f_i + 4f_{i+1} - f_{i+2}}{2\Delta\xi}, \tag{5.1}$$

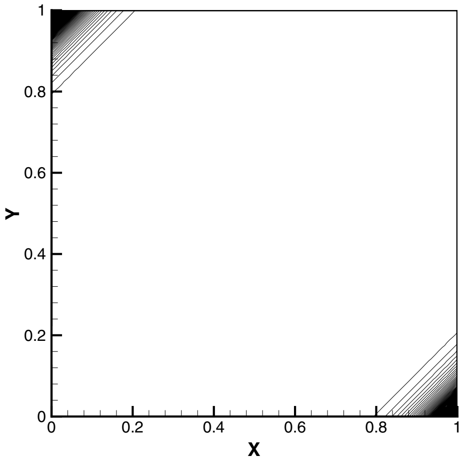


FIG. 5.1. *Isolines of the function* $f(x, y) = (x - y)^m$.

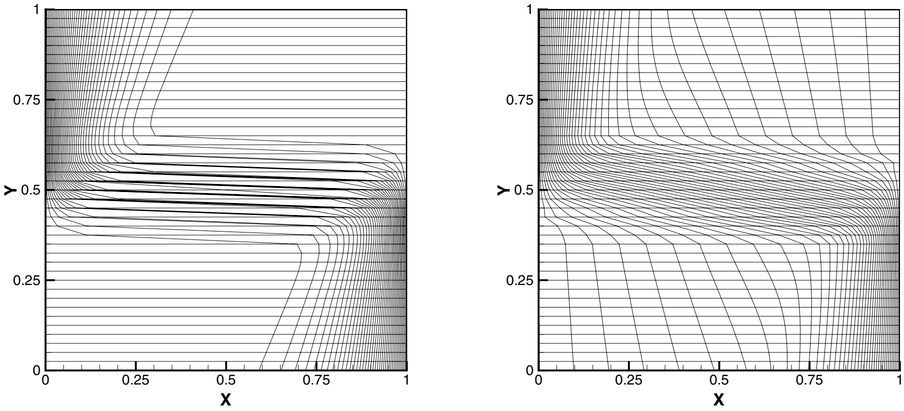


FIG. 5.2. *Analytical (left) and numerical (right) optimal* 40×40 *grids for the function* $f(x, y) = (x - y)^m$.

is used to evaluate both the ξ and η derivatives in (2.4). The test function has been chosen so that a family of the optimal grids satisfying the system of equations (2.12) can be integrated analytically to yield

$$\begin{aligned} &y_{\text{opt}} = v(\eta) \\ (5.2) \qquad &x_{\text{opt}} = v(\eta) + \left\{ -v(\eta)^{(m+1)/3} + \left[(1 - v(\eta))^{(m+1)/3} + v(\eta)^{(m+1)/3} \right] \xi \right\}^{3/(m+1)}. \end{aligned}$$

Note that the above equations have been derived by assuming $(m + 1)/3$ is an odd number. The numerical optimal grid is generated along fixed coordinate lines $\eta =$ const by using (3.10). Figure 5.2 shows the analytical and numerical optimal grids. Grid lines are concentrated near strong gradients of the function $f(x, y)$, which are

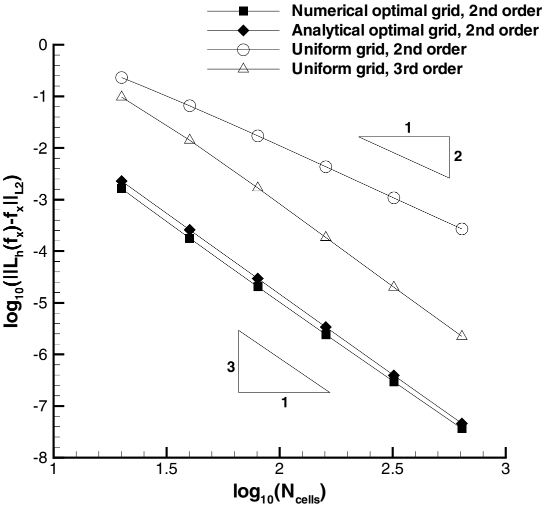


FIG. 5.3. *Error convergence for the second-order approximation of f_x of $f(x,y) = (x-y)^m$ calculated on* (1) *numerical optimal grid,* (2) *analytical optimal grid,* (3) *uniform grid, and* (4) *uniform grid with a third-order accurate discretization.*

located at points $(0,1)$ and $(1,0)$. At the same time, there are practically no grid points in the center of the domain so that very skewed grid cells are generated in this region. Despite these cell angles being very small, the L_2 truncation error norm computed in the computational space is superconvergent on both the analytical and numerical optimal grids, as evident in Figure 5.3. The comparison of the second-order approximation on the optimal grid and a third-order discretization on the uniform grid with the same number of grid points shows that the optimal grid results are about 10^2 times more accurate.

As follows from (5.2), the analytical optimal grid is not unique. Actually, any function $v(\eta)$ satisfying the conditions (3.20) should provide the optimal grid. Figure 5.4 shows the L_∞ norm of the truncation errors obtained on the analytical and numerical optimal grids corresponding to

$$v(\eta) = \begin{cases} 0.5(2\eta)^{0.25}, & 0 \le \eta \le 0.5, \\ 1 - 0.5[2(1-\eta)]^{0.25}, & 0.5 < \eta \le 1. \end{cases} \tag{5.3}$$

Similar to the optimal grid generated with $v(\eta) = \eta$, the results calculated on the numerical optimal grids are about four orders of magnitude more accurate than those obtained on the finest uniform grid with the same number of grid points.

The next test example is an approximation of the first derivatives f_x and f_y of the following function:

$$f(x,y) = x^{\frac{3-\gamma-2\theta}{\gamma-\theta}} y^{\frac{-3+2\gamma+\theta}{\gamma-\theta}}. \tag{5.4}$$

The physical domain considered is bounded by four curves, $y = \xi_0^{\theta-\gamma}x$, $y = (1+\xi_0)^{\theta-\gamma}x$, $y = x(x/\eta_0^{\phi})^{\theta/\gamma-1}$, $y = x[x/(1+\eta_0)^{\phi}]^{\theta/\gamma-1}$, where γ, θ, ϕ, ξ_0, and η_0 are parameters. The parameters θ and γ have been chosen as -5 and -1.25, respectively; this choice results in the function being singular along a line $y = 0$. Isolines of the

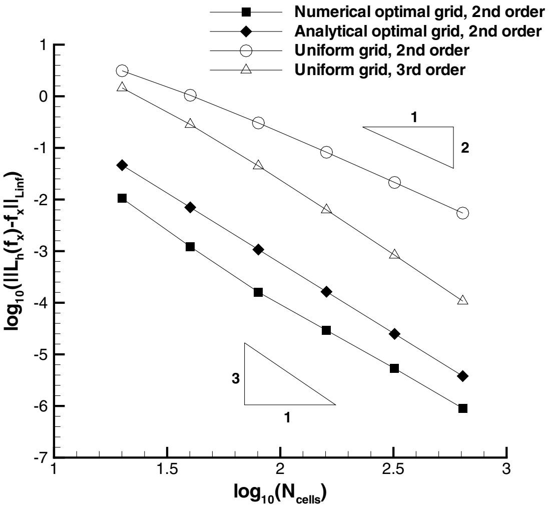


FIG. 5.4. *Error convergence for the second-order approximation of* f_x *of* $f(x, y) = (x - y)^m$ *calculated on* (1) *stretched numerical optimal grid,* (2) *stretched analytical optimal grid,* (3) *uniform grid, and* (4) *uniform grid with a third-order accurate discretization.*

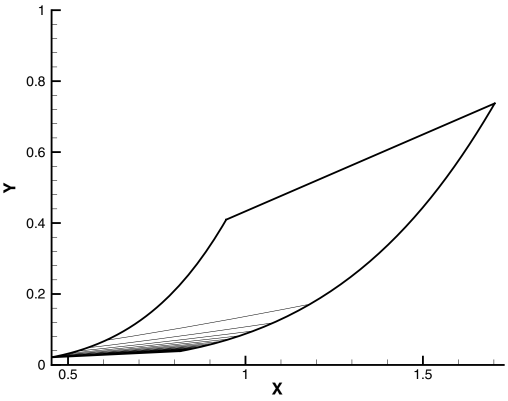


FIG. 5.5. *Isolines of the function* $f(x, y)$ *given by* (5.4).

function $f(x, y)$ are shown in Figure 5.5. In this test case, a family of the optimal grids satisfying equations (2.15) can be found analytically:

$$\begin{aligned} x_{\text{opt}} &= (\xi + \xi_0)^{\gamma}[v(\eta) + \eta_0]^{\phi}, \\ y_{\text{opt}} &= (\xi + \xi_0)^{\theta}[v(\eta) + \eta_0]^{\phi}, \end{aligned} \tag{5.5}$$

where v is an arbitrary smooth function satisfying the conditions (3.20). In numerical calculations, the parameters ξ_0 and η_0 were taken to be 1.25. A central second-order discretization is used for all the ξ and η derivatives in (2.4) and (2.13).

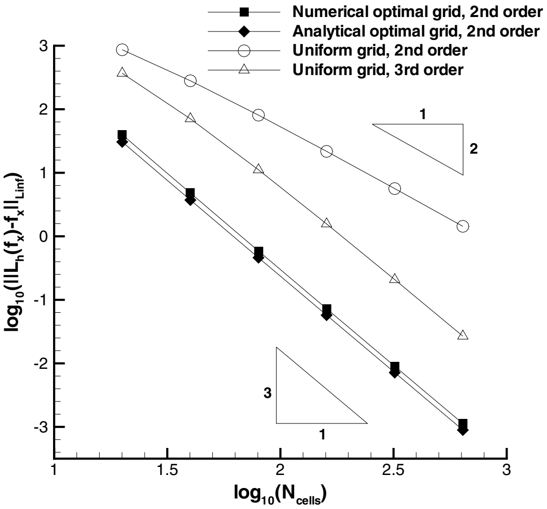


FIG. 5.6. *Error convergence for a central second-order approximation of* f_x *of* (5.4) *calculated on* (1) *numerical optimal grid,* (2) *analytical optimal grid,* (3) *uniform grid, and* (4) *uniform grid with a third-order accurate discretization.*

It should be noted that although the test function has the form $f = f_1(x)f_2(y)$, the analytical optimal grid (5.5) is not a tensor product grid of two 1D grids, one of which is optimal for $f_1(x)$ and the second one for $f_2(y)$. Using (3.12), it can easily be verified that the mapping (5.5) is optimal neither for $f_1(x) = x^{\frac{3-\gamma-2\theta}{\gamma-\theta}}$ nor for $f_2(y) = y^{\frac{-3+2\gamma+\theta}{\gamma-\theta}}$. At the same time, the mapping (5.5) is optimal for the function (5.4).

In Figure 5.6, error convergence plots of this approximation calculated on the analytical and numerical optimal grids corresponding to $v(\eta) = \eta$ are compared with results obtained by second- and third-order approximations on a uniform grid. The L_∞ norm of the truncation error of f_x calculated on the uniform grid exhibits the $O(\Delta\xi^2)$ convergence rate consistent with the second order of accuracy of the central differences. At the same time, the same second-order approximation of f_x on the optimal grid (5.5) exhibits a convergence rate of the order of $O(\Delta\xi^3)$. To generate the optimal grid numerically, a family of mesh lines $\eta =$ const is obtained by using (3.11). Then grid points are redistributed along these coordinate lines in accordance with (3.10). This grid point redistribution generates new lines $\eta =$ const and so on. The grid adaptation procedure is iterated until convergence. To reduce the residual by six orders of magnitude, 12 iterations were required. No attempt was made to optimize the iteration process. Figure 5.6 shows that the numerical optimal grid provides practically the same convergence rate as the analytical optimal grid (5.5). The truncation error norm calculated on the finest optimal grid is reduced by about two orders of magnitude as compared with the uniform grid results calculated with a third-order accurate formula.

As follows from (5.5), the optimal mapping is not unique. To demonstrate this property, the function $v(\eta)$ is chosen as

$$v(\eta) = \frac{k^\eta - 1}{k - 1}, \tag{5.6}$$

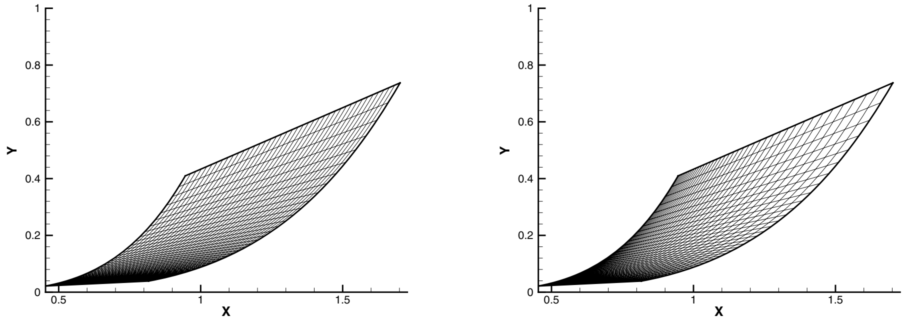


FIG. 5.7. *Analytical optimal* 40×40 *grids* (5.5) *corresponding to* $v(\eta) = \eta$ *(left) and* (5.6) *(right) for the function* (5.4).

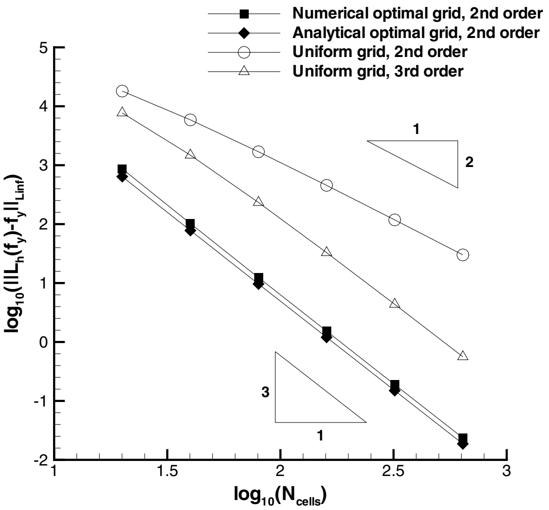


FIG. 5.8. *Error convergence for a central second-order approximation of* f_y *of* (5.4) *calculated on* (1) *numerical optimal grid,* (2) *analytical optimal grid,* (3) *uniform grid, and* (4) *uniform grid with a third-order accurate discretization.*

where the parameter k was chosen to be 10. The analytical optimal grids corresponding to the function $v(\eta) = \eta$ and the function (5.6) are shown in Figure 5.7. The optimal mapping (5.5), (5.6) results in grid lines $\eta = \text{const}$ being concentrated near the left boundary $\eta = 0$, as shown in Figure 5.7. Although this optimal mapping is different from that considered earlier, the error convergence plot for the grid (5.5), (5.6) is essentially the same as that shown in Figure 5.6 for the optimal grid corresponding to $v(\eta) = \eta$ and, therefore, is not presented here.

As shown in section 2, the optimal mapping (5.5) satisfying the system of equations (2.15) has to provide superconvergence not only for f_x but also for f_y. The L_∞ norm of the truncation error of f_y approximated by using the central second-order formulas is superconvergent on the same optimal grid (5.5), as is evident in Figure 5.8.

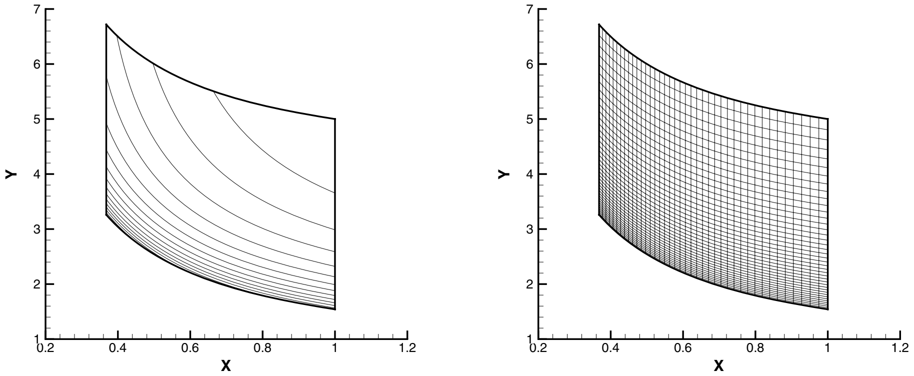


FIG. 5.9. *Isolines of the function* (5.7) *and the analytical optimal* 40×40 *grid* (5.8).

The third test function considered,

$$f(x, y) = \frac{1}{x} + \frac{mx}{xy - 1}, \tag{5.7}$$

has singularities along lines $x = 0$ and $y = 1/x$. For this function a particular solution of (2.15) can be found analytically:

$$\begin{aligned} x_{\text{opt}}(\xi, \eta) &= \mathrm{e}^{-\alpha\xi}, \\ y_{\text{opt}}(\xi, \eta) &= \mathrm{e}^{\alpha\xi} + m\mathrm{e}^{-\beta\eta}. \end{aligned} \tag{5.8}$$

As in the previous test example, this optimal grid satisfies both the grid adaptation criteria (3.1) and the optimal mapping (3.10) calculated with the function (5.7).

In the present test example, the parameters α, β, and m have been chosen as 1, 2, and 4, respectively. For such a choice of the parameters, the physical domain boundaries are $x = 1$, $x = \mathrm{e}^{-\alpha}$, $y = 1/x + 4$, and $y = 1/x + 4\mathrm{e}^{-2}$. Isolines of the function $f(x, y)$ and the optimal grid (5.8) are depicted in Figure 5.9. Notably, the optimal grid is orthogonal neither in the domain nor at the boundaries. Moreover, the grid lines are concentrated near strong gradients, and at the same time they are not strictly aligned with the isolines of $f(x, y)$.

As in the first test example, second-order accurate approximations of f_x and f_y are obtained by using the second-order approximation (5.1) for all the ξ and η derivatives in (2.4) and (2.13).

Similar to the previous test case, the optimal mapping (5.8) satisfies the system of equations (2.15), and, therefore, a grid generated by this transformation is optimal for the gradient of $f(x, y)$. Note that the numerical optimal grid is generated using formulas (3.10) and (3.11) so that grid points are redistributed in both ξ and η iteratively, as described in section 3. For this test problem, eight iterations were needed to reach convergence. A comparison of the truncation error convergences of ∇f obtained on the optimal and uniform grids is shown in Figure 5.10. The global order of the second-order central approximation in two dimensions is increased by one on the optimal grid. Furthermore, the L_∞ norm of the truncation error on the finest mesh is about four orders of magnitude less than that obtained on the corresponding uniform grid. As shown in Figure 5.10, the new grid adaptation criterion enables us to reach

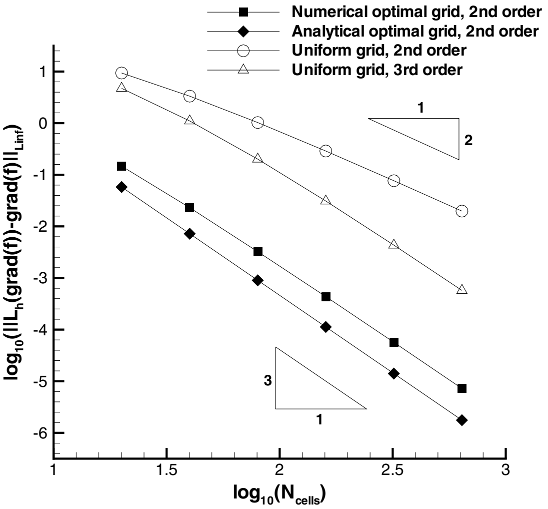


FIG. 5.10. *Error convergence for the second-order approximation of* ∇f *of* (5.7) *calculated on* (1) *numerical optimal grid,* (2) *analytical optimal grid,* (3) *uniform grid, and* (4) *uniform grid with a third-order accurate discretization.*

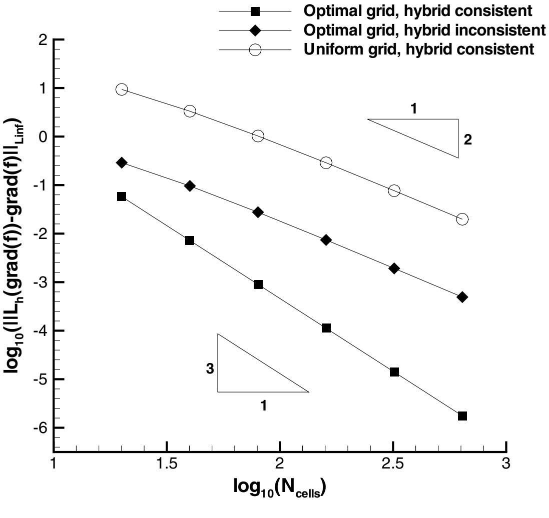


FIG. 5.11. *Error convergence for the second-order hybrid approximation* (5.9) *of* ∇f *of* (5.7) *calculated with the consistent and inconsistent discretization of the metric coefficients on the uniform and optimal grids.*

the asymptotic convergence rate on a coarse grid. At the same time, application of a third-order accurate discretization on the uniform grid does not provide such essential reduction in the truncation error as was obtained on the optimal grid.

Importance of the identical approximations of the first derivatives f_ξ, f_η and the corresponding metric coefficients x_ξ, y_ξ, x_η, and y_η is illustrated in Figure 5.11. The figure shows that if all the ξ and η derivatives in (2.4) and (2.13) are evaluated by hybrid discretization

$$\left(\frac{\partial f}{\partial \xi}\right)_i = \begin{cases} \frac{1}{2\Delta\xi}(f_{i+1} - f_{i-1}), & i \text{ even}, \\ \frac{1}{2\Delta\xi}(-3f_i + 4f_{i+1} - f_{i+2}), & i \text{ odd}, \end{cases} \tag{5.9}$$

the order of approximation of ∇f is increased from 2 to 3 when grid points are redistributed in accordance with (5.8). However, if the metric coefficients x_ξ, y_ξ, x_η, and y_η are evaluated by the two-point symmetric second-order difference expression in the entire computational domain, while both the hybrid approximation of f_ξ and f_η (5.9) and the optimal grid (5.8) remain the same, the order of the inconsistent approximation deteriorates to 2 and the L_∞ truncation error norm increases by a factor of $10^{2.5}$ on the finest mesh.

The last test example is a construction of the optimal grid that minimizes the leading truncation error term of $f_x + g_y$. Second-order central differences are used to evaluate both the ξ and η derivatives in (4.5) written in the computational space. The functions $f(x, y)$ and $g(x, y)$ have been chosen to be

$$\begin{aligned} f(x,y) &= x^{\frac{3\theta-2\phi}{\theta\phi(1-m)}} y^{\frac{\phi(2-\theta)-m\theta(3-\phi)}{\theta\phi(1-m)}}, \\ g(x,y) &= x^{\frac{-\phi(2-m\theta)+\theta(3-\phi)}{\theta\phi(1-m)}} y^{\frac{2\phi-3m\theta}{\theta\phi(1-m)}}. \end{aligned} \tag{5.10}$$

In this case a one-parameter family of the optimal grids satisfying (4.6) can be obtained analytically:

$$\begin{aligned} x_{\text{opt}}(\xi,\eta) &= (\xi+\xi_0)^{m\theta}[v(\eta)+\eta_0]^{\phi}, \\ y_{\text{opt}}(\xi,\eta) &= (\xi+\xi_0)^{\theta}[v(\eta)+\eta_0]^{\phi}, \end{aligned} \tag{5.11}$$

where the function $v(\eta)$ should satisfy the conditions (3.20). In this test example the parameters θ, ϕ, m, ξ_0, and η_0 were set at 1, 3, 0.5, 0.5, and 0.5, respectively. Such a choice provides that the physical domain is bounded by the following four curves: (1) $y = x/\sqrt{2}$, (2) $y = \sqrt{3/2}x$, (3) $y = 8x^2$, (4) $y = 8x^2/27$; and the functions $f(x, y)$ and $g(x, y)$ are singular along a line $x = 0$. As follows from (5.11), the optimal grid is not unique. To demonstrate this property, two functions, $v_1(\eta)$ and $v_2(\eta)$, have been chosen:

$$\begin{aligned} v_1(\eta) &= \eta, \\ v_2(\eta) &= \tfrac{k^\eta-1}{k-1}, \quad k = 10. \end{aligned} \tag{5.12}$$

The optimal grids (5.11) defined by the functions $v_1(\eta)$ and $v_2(\eta)$ are shown in Figure 5.12. The transformation of the η-coordinate caused by the function $v_2(\eta)$ results in stretching of grid lines to a boundary $\eta = 0$, which, like the other boundaries in the physical domain, remains unchanged. Similar to the previous example, the optimal grids are essentially nonorthogonal, and, at the same time, the grid nodes are concentrated near strong gradients of the f and g functions. In spite of the fact that the optimal grids shown in Figure 5.12 are quite different, the truncation error convergence rates obtained on these grids are practically identical, as depicted in Figure 5.13. This figure shows that the central second-order approximation of (4.5), calculated on the optimal grids (5.11) and (5.12), exhibits the $O(\Delta\xi^3)$ convergence rate, which allows us to reduce the L_∞ norm of the truncation error by 2.5 orders of magnitude

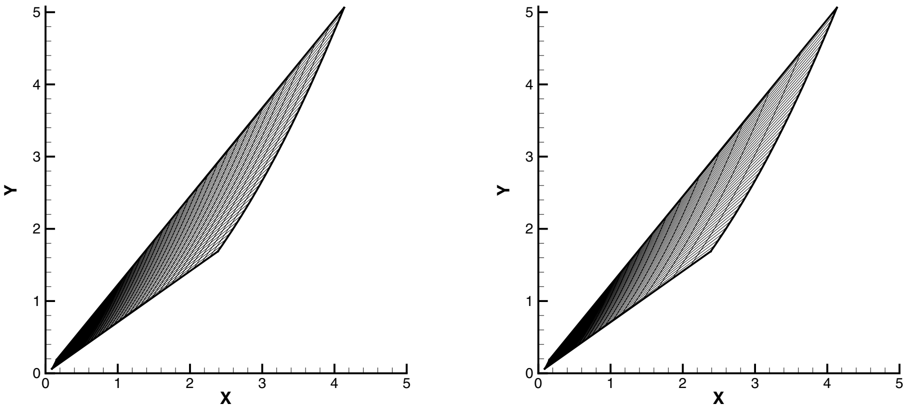


FIG. 5.12. *Analytical optimal* 40 × 40 *grids* (5.11) *corresponding to* $v(\eta) = \eta$ *(left) and* (5.6) *(right) for the function* (5.10).

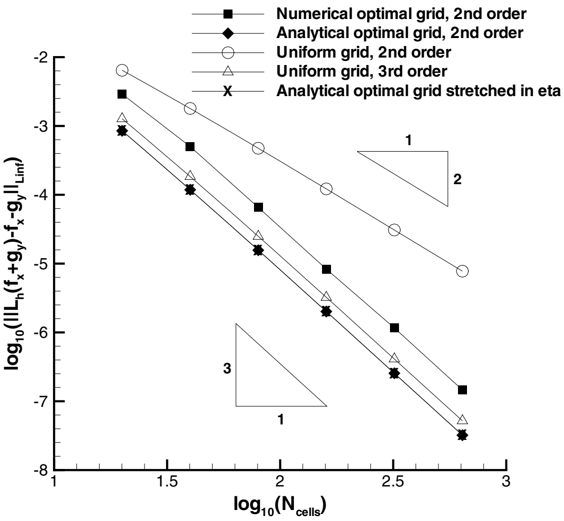


FIG. 5.13. *Error convergence for a central second-order approximation of* $f_x + g_y$ *of* (5.10) *calculated on* (1) *numerical optimal grid,* (2) *analytical optimal grid,* (3) *uniform grid, and* (4) *uniform grid with a third-order accurate discretization.*

as compared with the uniform grid results. The numerical optimal grid generated by using the iteration procedure described in the foregoing section also provides superconvergence. Note that the truncation error calculated on the numerical optimal grid is about two orders of magnitude more accurate than that obtained with the same second-order accurate approximation on the corresponding finest uniform grid.

6. Conclusion. A new multidimensional grid adaptation strategy based on the minimization of the leading truncation error term of an arbitrary order finite difference discretization has been developed. The main idea of the method is to redistribute grid points so that the leading truncation error terms resulting from the differential operator and the metric coefficients cancel each other. The result is that the design order of approximation on the optimal grid is increased by one in the entire computational domain. It has been shown that these grid adaptation criteria are stable under perturbations of the optimal grid. In contrast to most adaptive grid techniques, the present method does not explicitly require an a posteriori error estimate that makes it computationally inexpensive. Another very attractive characteristic of the new approach is its applicability to hybrid discretizations. We have proven that if the differential operator and the metric coefficients are evaluated identically, then the new grid adaptation criterion is generic in the sense that it remains valid in the entire computational domain regardless of points where the hybrid difference operator switches from one approximation to another. When a family of coordinate lines (e.g., $\eta = \text{const}$) is given a priori, the 2D optimal mapping that increases the order of approximation of ∇f in ξ from 2 to 3 has been derived analytically. For some functions, a family of the optimal grids can be constructed; this shows that the optimal mapping is not unique. One of the main advantages of the new method is that it can be extended directly to nonlinear PDEs and three dimensions. The numerical calculations show that the truncation error of both f_x and f_y obtained on the optimal grid is superconvergent and reduced by several orders of magnitude as compared with the second- and third-order uniform grid results for all the test examples considered.

REFERENCES

[1] J. F. Thompson and C. W. Mastin, *Order of difference expressions on curvilinear coordinate systems*, in Proceedings of the ASME Fluid Engineering Conference on Advances in Grid Generation, Houston, TX, 1983.

[2] J. D. Hoffman, *Relationship between the truncation errors of centered finite-difference approximation on uniform and nonuniform meshes*, J. Comput. Phys., 46 (1982), pp. 469–474.

[3] E. Turkel, *Accuracy of Schemes with Nonuniform Meshes for Compressible Fluid Flows*, ICASE Report 85-43, Hampton, VA, 1985.

[4] N. K. Yamaleev, *Minimization of the truncation error by grid adaptation*, J. Comput. Phys., 170 (2001), pp. 459–497.

[5] I. Babuška and W. Rheinboldt, *A-posteriori error estimates for the finite element method*, Internat. J. Numer. Methods Engrg., 12 (1978), pp. 1597–1615.

[6] H. A. Dwyer, *Grid adaptation for problems in fluid dynamics*, AIAA J., 22 (1984), 1705–1712.

[7] D. Catherall, *The adaptation of structured grids to numerical solutions for transonic flows*, Internat. J. Numer. Methods Engrg., 32 (1991), pp. 921–937.

[8] J. U. Brackbill and J. S. Saltzman, *Adaptive zoning for singular problems in two dimensions*, J. Comput. Phys., 46 (1982), pp. 342–368.

[9] W. Huang and D. M. Sloan, *A simple adaptive grid method in two dimensions*, SIAM J. Sci. Comput., 15 (1994), pp. 776–797.

[10] B. Pierson and P. Kutler, *Optimal nodal point distribution for improved accuracy in computational fluid dynamics*, AIAA J., 18 (1980), pp. 49–54.

[11] A. Mackenzie, D. F. Mayers, and A. J. Mayfield, *Error estimates and mesh adaptation for a cell vertex finite volume scheme*, Notes Numer. Fluid Mech., 44 (1993), pp. 290–310.

[12] D. Ait-Ali-Yahia, W. G. Habashi, A. Tam, M.-G. Vallet, and M. Fortin, *A directionally adaptive methodology using an edge-based error estimate on quadrilateral grids*, Internat. J. Numer. Methods Fluids, 23 (1996), pp. 673–690.

[13] D. S. McRae and K. R. Laflin, *Dynamic grid adaption and grid quality*, in Handbook of Grid Generation, J. F. Thompson, B. K. Soni, and N. P. Weatherill, eds., CRC Press, New York, 1999, pp. 34-1–34-33.

[14] J. Peraire, M. Vahdati, K. Morgan, and Zenkiewicz, *Adaptive remeshing for compressible flow computations*, J. Comput Phys., 72 (1987), pp. 449–466.

[15] E. F. D'Azevedo, *Optimal triangular mesh generation by coordinate transformation*, SIAM J. Sci. Statist. Comput., 12 (1991), pp. 755–786.

[16] I. N. Bronshtein and K. A. Semendyayev, *Handbook of Mathematics*, Van Nostrand Reinhold Company, New York, 1985.

SIAM J. SCI. COMPUT.
Vol. 23, No. 5, pp. 1731–1751

THE CONVERGENCE OF SPECTRAL AND FINITE DIFFERENCE METHODS FOR INITIAL-BOUNDARY VALUE PROBLEMS[*]

NATASHA FLYER[†][‡] AND PAUL N. SWARZTRAUBER[†]

Abstract. The general theory of compatibility conditions for the differentiability of solutions to initial-boundary value problems is well known. This paper introduces the application of that theory to numerical solutions of partial differential equations and its ramifications on the performance of high-order methods. Explicit application of boundary conditions (BCs) that are independent of the initial condition (IC) results in the compatibility conditions not being satisfied. Since this is the case in most science and engineering applications, it is shown that not only does the error in a spectral method, as measured in the maximum norm, converge algebraically, but the accuracy of finite differences is also reduced. For the heat equation with a parabolic IC and Dirichlet BCs, we prove that the Fourier method converges quadratically in the neighborhood of $t = 0$ and the boundaries and quartically for large t when the first-order compatibility conditions are violated. For the same problem, the Chebyshev method initially yields quartic convergence and exponential convergence for $t > 0$. In contrast, the wave equation subject to the same conditions results in inferior convergence rates with all spectral methods yielding quadratic convergence for all t. These results naturally direct attention to finite difference methods that are also algebraically convergent. In the case of the wave equation, we prove that a second-order finite difference method is reduced to 4/3-order convergence and numerically show that a fourth-order finite difference scheme is apparently reduced to 3/2-order. Finally, for the wave equation subject to general ICs and zero BCs, we give a conjecture on the error for a second-order finite difference scheme, showing that an $O(N^{-2} \log N)$ convergence is possible.

Key words. compatibility conditions, convergence theory, spectral and finite difference methods

AMS subject classifications. 65M06, 65M12, 65M70

PII. S1064827500374169

1. Introduction.

The regularity of solutions for initial-boundary value problems (IBVPs) is determined by the compatibility of the initial condition (IC) and the boundary conditions (BCs). IBVPs will have singular solutions at the corners of the space-time domain if the BCs do not satisfy the partial differential equation (PDE) or any of its higher-order derivatives, even if the IC is C^∞. The Cauchy–Kowalesky theorem [19] provides a solution in the neighborhood of $t = 0$ to the same problem without BCs. If compatibility exists, then the BCs can always be computed by the solution obtained from the Cauchy–Kowalesky theorem and the IC. In such cases, the IBVP could in fact be posed as an initial-value problem (IVP). The set of compatibility conditions are derived by equating the time derivatives of the BCs to those of the solution to the IVP given by the Cauchy–Kowalesky theorem.

Literature on the theory of compatibility conditions and the regularity of solutions for IBVPs is extensive, starting in the 1950s with the work of Ladyzenskaja [10], [11]. In the 1960s, Ladyzenskaja, Solonnikov, and Ural'ceva [12] and Friedman [3] discussed the regularity of the solution for linear and quasi-linear parabolic systems. During this same time, Kreiss [4] and Hersh [7] extended the theory for constant coefficient hyperbolic systems by providing the necessary and sufficient conditions for

[*] Received by the editors June 20, 2000; accepted for publication (in revised form) August 6, 2001; published electronically January 30, 2002.
http://www.siam.org/journals/sisc/23-5/37416.html

[†] National Center for Atmospheric Research, Advanced Study Program, P.O. Box 3000, Boulder, CO 80307-3000 (pauls@scd.ucar.edu). NCAR is sponsored by the National Science Foundation.

[‡] Current address: Department of Applied Mathematics, University of Colorado, Boulder, CO 80309 (flyer@colorado.edu).

the problem to be well-posed. It was further developed in the 1970s by Rauch and Massey [16] and Sakamoto [17] for hyperbolic systems with time-dependent coefficients. In the 1980s, Temam [20] extended the theory by obtaining the compatibility conditions for semilinear evolution equations while Smale [18] gave a rigorous proof of the compatibility conditions for the heat and wave equation on a bounded domain. These accomplishments have been confined to the realm of theoretical mathematics. Yet, not satisfying the compatibility conditions of a system can have a significant computational impact, particularly for high-order accurate methods, such as spectral methods, whose performance intrinsically depends on the smoothness of the solution. For nonsmooth initial data, there is a wealth of information on the convergence theory of numerical schemes in a variety of norms for parabolic and hyperbolic IVPs [13], [5], [21], [22]. These analyses can be applied to the IBVPs *if* the IBVP can be posed as an IVP with discontinuous ICs through periodic extensions of the ICs to the entire real axis [6], [9], [14]. However, such periodic extensions may induce singularities that do not exist in a smooth nonperiodic solution of an IBVP.

More recently, the impact of noncompatible BCs has been explored by Boyd and Flyer [2] in a computational framework. This paper appeared in a special issue on spectral methods. In the prologue, Karniadakis states "This is the first such effort in bringing the theory of compatibility conditions from the mathematical literature to the numerical community." This work illustrated the ubiquity of compatibility conditions and analyzed the connection between incompatibility and the rate of convergence of Chebyshev spectral series. However, the convergence of the error was not discussed or compared to alternatives such as the finite difference method. The focus of the paper, as has been in the mathematical literature, is on smoothing the initial condition so as to satisfy the compatibility conditions [2], [5], [18], [15]. The difficulty with this approach is that it leaves the fundamental question unanswered: namely, "What is the rate of convergence to the solution of the original unperturbed problem?" If the IC is smoothed, spectral convergence might be achieved *but* to a solution that differs algebraically from the original problem.

The focus of the current work is on the convergence of the approximate solution to the exact solution. As discussed in the first paragraph, the temporal derivatives of the solution as defined by the IC, differential operator, and Cauchy–Kowalesky theorem will not equal those determined by the independent BCs. The resulting singularities in the corners of the temporal-spatial domain, which are independent of the smoothness of the IC, disrupt the convergence of spectral and finite difference methods in a manner that differs significantly between parabolic and hyperbolic systems. A detailed analysis of the induced singularities for two model problems and their effect on the convergence of both numerical methods is determined. Since an incompatibility exists any time the BC is independent of the IC, algebraic convergence in the maximum norm is the expectation for IBVPs. In most science and engineering applications, this is the usual scenario.

In section 2, the infinite set of compatibility conditions for the generalized heat and wave equations with time-dependent BCs is presented. The purpose is to lay the groundwork for describing the inherent singular nature of IBVPs and how it affects the convergence rate of the error. In section 3, we examine the prototype parabolic case, namely the heat equation subject to Dirichlet BCs and a smooth parabolic IC. We use the Fourier method to determine an exact expression for the approximate solution, represented by a truncated Fourier series. Convergence rates are then developed for both finite difference and Chebyshev spectral methods, the latter of which is normally the basis set of choice for a problem on a finite domain. For the parabolic example in section 3, u_t is discontinuous at $t = 0$. Nevertheless, the Chebyshev method is shown to yield spectral convergence for $t > 0$, but only quartic convergence in the neighborhood of the discontinuity, which would be the relevant consideration for the computation of transient heat flow. Furthermore, the usual Fourier method is only quadratically convergent near the discontinuity and quartically convergent elsewhere. Thus, the convergence rate of the error is shown to be nonuniform and algebraic as measured in the maximum norm.

The hyperbolic case is discussed in section 4. Discontinuities that are induced by the incompatibilities now propagate throughout the region. It is shown that all spectral methods will yield algebraic convergence for $t > 0$. The Fourier method is used again to derive an exact expression for the error. Similar to the parabolic case, the results are compared with the centered finite difference and Chebyshev methods which yield algebraic convergence for smooth ICs. The performance of finite difference methods is also significantly diminished when the compatibility conditions are not satisfied. For this reason, the appendix contains the rates of convergence for a second-order finite difference method for a variety of smooth ICs and shows, for example, that second-order convergence is reduced to 4/3-order.

2. Compatibility conditions for the generalized heat and wave equations.

Here compatibility conditions are reviewed to illustrate their fundamental impact on the smoothness of the solutions to IBVPs. We simply state the necessary conditions for the solution to be C^∞ on the domain for the heat and wave equation with time-dependent BCs. The complete proofs of necessity and sufficiency are given for parabolic systems in [11], [12], [3], [18] and for hyperbolic systems in [10], [16], [17], [4].

THEOREM 2.1. *The domain is* $[0,T] \times \Omega$, *where* Ω *is a d-dimensional spatial domain with a boundary* $\partial\Omega$ *which is a* C^∞ *manifold of dimension* $(d-1)$. *Let L be an elliptic operator of the form*

$$L = \sum_{i=1}^{d}\sum_{j=1}^{d} A_{ij}(x)\frac{\partial}{\partial x_i}\frac{\partial}{\partial x_j} + \sum_{j=1}^{d} B_j(x)\frac{\partial}{\partial x_j} + \sum_{j=1}^{d} C_j(x). \tag{2.1}$$

The generalized linear diffusion problem is

$$u_t = Lu, \qquad u(x,0) = u_0(x) \in \Omega. \tag{2.2}$$

The generalized wave equation is

$$u_{tt} = Lu, \quad u(x,0) = u_0(x), \quad u_t(x,0) = v_0(x) \in \Omega. \tag{2.3}$$

Both are subject to the BC

$$u = f(t) \ \in \ \partial\Omega \ \ \forall t. \tag{2.4}$$

Then, the necessary and sufficient compatibility conditions for u(x,t) to be C^{2k} *for the heat equation are*

$$L^k u_0 = \frac{\partial^k f}{\partial t^k}, \qquad k = 0, 1, 2, \ldots \ \in [0,0] \times \partial\Omega \tag{2.5}$$

and for the wave equation are

$$L^k u_0 = \frac{\partial^{2k} f}{\partial t^{2k}} \quad \text{and} \quad L^k v_0 = \frac{\partial^{2k+1} f}{\partial t^{2k+1}}, \quad k = 0, 1, 2, \ldots \ \in [0,0] \times \partial\Omega. \tag{2.6}$$

In essence (2.5) and (2.6) state that at time $t = 0$, the temporal derivatives of the solution determined by the application of the differential operator to the IC must *equal* the temporal derivatives of the BCs. When these conditions are satisfied, we do *not* need to *impose BCs* because they can be derived from the Taylor series expansion of the IC. When (2.5) and (2.6) do not hold, we induce a singularity in the $2k$th-order derivative of the solution at $t = 0$ on $\partial\Omega$. Thus, imposing BCs that are independent of the IC places an infinite set of constraints on the solution which differ from the constraints enforced by the PDE, resulting in discontinuities in the corners of the temporal-spatial domain. The manner in which these incompatibilities affect the accuracy of the numerical method depends on whether the system is parabolic or hyperbolic. In the next section we will discuss the parabolic case, followed in section 4 by the hyperbolic case.

3. A parabolic example. Our goal is to study the convergence of the error for approximate spectral and finite difference solutions to IBVPs when the compatibility conditions of the system are not satisfied. With this in mind, we first study the effect on parabolic systems, taking as our example the one-dimensional heat equation

$$u_t = u_{xx} \tag{3.1}$$

subject to BCs

$$u(0,t) = u(\pi,t) = 0 \tag{3.2}$$

and IC

$$u_0 = u(x,0) = \frac{\pi}{8}x(\pi - x). \tag{3.3}$$

The first-order compatibility condition, ($k = 1$) in (2.5), is not satisfied because

$$Lu_0 = \left(\frac{\partial^2 u_0}{\partial x^2}\right) = -\frac{\pi}{4} \qquad \forall x \tag{3.4}$$

and

$$\frac{\partial u}{\partial t}|_{x=0,\pi} = 0 \qquad \forall t. \tag{3.5}$$

Thus from (3.1), $u_{xx}|_{x=0,\pi} = u_t|_{x=0,\pi} = 0$ yet $u_{xx} = -2$ in the interior. To see the impact of the jump discontinuity at the boundary in the second derivative at $t = 0$ on the decay rate of the error we will solve the problem exactly at time $t = 0$ via a truncated Fourier series, then compare this together with finite difference and Chebyshev methods against the exact solution for all time.

Time integration is assumed exact throughout, which yields the approximate spectral representation

$$u_N(x_i,t) = \sum_{n=1,\text{odd}}^{N-1} c_N(n)e^{-n^2 t}\sin nx_i \tag{3.6}$$

of the exact solution, given by

$$u(x_i,t) = \sum_{n=1,\text{odd}}^{\infty} \frac{e^{-n^2 t}}{n^3}\sin nx_i. \tag{3.7}$$

Gottlieb and Orszag [8] note that for the exact coefficients $c_N(n)$, $u_N(x,t)$ converges spectrally to $u(x,t)$. This is also true if $u(x,0)$ is known for all x because $c_N(n)$ can be computed to any accuracy. In this work, we assume, as with most scientific data, that $u(x,0)$ is given in tabular form $u_i = u(x_i,t)$, where $x_i = i\pi/N$. In science and engineering, we cannot assume we know the function exactly and therefore must approach the problem from a numerical standpoint. The coefficients $c_N(n)$ are then computed in the traditional manner, using trigonometric interpolation, as

$$c_N(n) = \frac{2}{N}\sum_{i=1}^{N-1} u_i \sin nx_i. \tag{3.8}$$

With this implementation of the spectral method, it will be shown that the approximate spectral representation (3.6) converges algebraically to the exact solution in the maximum norm, and, furthermore, convergence is nonuniform. This characteristic algebraic convergence is method-independent due to the inherent singularity induced by violating the compatibility conditions from explicit application of the BCs. In the next section, we derive an exact expression for $c_N(n)$ which will provide the initial error in the coefficients for the Fourier method.

3.1. The approximate spectral solution. In what follows, we derive an analytical representation of the coefficients $c_N(n)$ in (3.6). From (3.7)

$$u(x_i,0) = \sum_{n=1,\mathrm{odd}}^{\infty} \frac{1}{n^3}\sin\left(n\frac{i\pi}{N}\right), \tag{3.9}$$

$$\begin{aligned} u(x_i,0) = \sum_{n=1,\mathrm{odd}}^{N-1} \Bigg[& \frac{1}{n^3}\sin(n\frac{i\pi}{N}) \\ & + \sum_{m=1}^{\infty} \frac{1}{(2mN+n)^3}\sin\left\{(2mN+n)i\frac{i\pi}{N}\right\} \\ & + \frac{1}{(2mN-n)^3}\sin\left\{(2mN-n)i\frac{i\pi}{N}\right\}\Bigg], \end{aligned} \tag{3.10}$$

or

$$u(x_i,0) = \sum_{n=1,\mathrm{odd}}^{N-1} \left\{\frac{1}{n^3} + \sum_{m=1}^{\infty}\frac{1}{(2mN+n)^3} - \frac{1}{(2mN-n)^3}\right\}\sin\left(n\frac{i\pi}{N}\right) \tag{3.11}$$

and

$$u(x_i,0) = \sum_{n=1,\mathrm{odd}}^{N-1} \left\{\frac{1}{n^3} - \frac{n}{N^4}\Psi(n/N)\right\}\sin\left(n\frac{i\pi}{N}\right), \tag{3.12}$$

where

$$\Psi(s) = 2\sum_{m=1}^{\infty}\frac{3m^2+s^2}{[m^2-s^2]^3}. \tag{3.13}$$

$\Psi(s)$ can be called the alias function since it represents the error in the coefficients due to truncating the series after N terms. The initial error in the coefficients is due to the fact that a sine series cannot represent a nonperiodic polynomial with a finite number of terms. For use later, we observe that $\Psi(s)$ is positive and monotonically increasing on the interval $[0,1]$ with extremes $\Psi(0) \approx 0.4058712$ and $\Psi(1) = 1$.

Equating (3.6) and (3.12), we obtain the desired result

$$c_N(n) = \frac{1}{n^3} - \frac{n\Psi(n)}{N^4}. \tag{3.14}$$

We study convergence only as it relates to spatial discretization, and therefore temporal integrations are represented analytically. The spectral approximation is then given by

$$u_N(x_i,t) = \sum_{n=1,\text{odd}}^{N-1} e^{-n^2 t}\left[\frac{1}{n^3} - \frac{n\Psi(n/N)}{N^4}\right]\sin nx_i. \tag{3.15}$$

In the next two sections it will be proved that although the spectral series for both the exact and approximate solutions decay exponentially, the error decays algebraically and nonuniformly over the domain. Section 3.2 explores convergence in the neighborhood of the discontinuity induced by the incompatibility of imposing BCs. Section 3.3 looks at convergence in the interior of the domain, bounded away from the singularities at $(0,0)$ and $(0,\pi)$.

3.2. Quadratic convergence near $(0,0)$ and $(0,\pi)$. The discontinuity in u_t at $(0,0)$ induces an error in $\partial u_N/\partial t(\pi/N,0)$ that is bounded *away* from zero for *all* N. In what follows, we show that this produces an $O(N^{-2})$ error in the neighborhood of $(0,0)$ and likewise near $(\pi,0)$. With a derivation similar to that preceding (3.12), the exact solution is given by

$$u(x_i,t) = \sum_{n=1,\text{odd}}^{N-1}\left\{\frac{e^{-n^2t}}{n^3} - \sum_{m=2,even}^{\infty}\left[\frac{e^{-(mN+n)^2t}}{[mN+n]^3} - \frac{e^{-(mN-n)^2t}}{[mN-n]^3}\right]\right\}\sin nx_i. \tag{3.16}$$

Subtracting $u_N(x_i,t)$, given by (3.15), defines the error

$$e_N(x_i,t) = \sum_{n=1,\text{odd}}^{N-1}\left\{\sum_{m=2,even}^{\infty}\left[\frac{e^{-(mN+n)^2t} - e^{-n^2t}}{[mN+n]^3} - \frac{e^{-(mN-n)^2t} - e^{-n^2t}}{[mN-n]^3}\right]\right\}\sin nx_i. \tag{3.17}$$

Then, on the discrete trajectory $x_1 = \pi/N$ and $t_N = 1/N^2$

$$e_N\left(\frac{\pi}{N},\frac{1}{N^2}\right) = \frac{1}{N^2}I_N, \tag{3.18}$$

where

$$I_N = \frac{1}{N}\sum_{n=1,\text{odd}}^{N-1}\left\{\sum_{m=2,even}^{\infty}\left[\frac{e^{-(m+n/N)^2} - e^{-(n/N)^2}}{[m+n/N]^3} - \frac{e^{-(m-n/N)^2} - e^{-(n/N)^2}}{[m-n/N]^3}\right]\right\}\sin n\frac{\pi}{N}. \tag{3.19}$$

I_N defines the midpoint quadrature of the smooth positive function, which is the integrand in (3.20) below. Therefore,

$$I = \lim_{N\to\infty} I_N = \int_0^1 \sin\pi s \sum_{m=2,even}^{\infty}\left[\frac{e^{-(m+s)^2} - e^{-s^2}}{[m+s]^3} - \frac{e^{-(m-s)^2} - e^{-s^2}}{[m-s]^3}\right]ds. \tag{3.20}$$

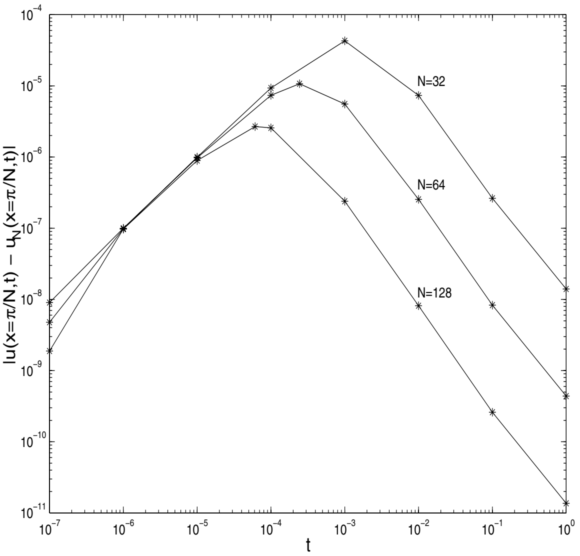


FIG. 3.1. $|e_N|$ *for the Fourier method on the interval* $t\epsilon(0,1]$ *for* $x=\pi/N$ *and fixed* N.

Numerical evaluation of the midpoint rule (3.20) with $N = 512$ provides the approximate value $I \approx .0876323127$. This completes the proof that convergence is at best quadratic. For each N there exists a point $(\pi/N, 1/N^2)$ at which the error is I/N^2. The error as a function of time for $x = \pi/N$ is plotted in Figure 3.1, where the maximum occurs at $t = 1/N^2$. Likewise, we can plot the error as a function of space for $t = 1/N^2$ and see that the maximum error occurs at the endpoints as shown in Figure 3.2(a). Numerical experiments, in the following sections, will confirm that indeed convergence *is* quadratic.

3.3. Quartic convergence for a fixed $t > \epsilon > 0$.

For any fixed $t > \epsilon > 0$ in the interior of the domain, bounded away from the singularities at $(0, 0)$ and $(0, \pi)$, $\|e_N(x_i,t)\|_2$ converges at best quartically. As was shown in (3.15), the coefficients of the approximate trigonometric series have an initial error of $O(1/N^4)$. This, in turn, induces quartic convergence for any fixed t.

The difference between (3.7) and (3.15) provides the error in the spectral approximation

$$e_N(x_i,t) = u(x_i,t) - u_N(x_i,t) = S_N(x_i,t) + E_N(x_i,t), \tag{3.21}$$

where

$$S_N(x_i,t) = \frac{1}{N^4} \sum_{n=1,\mathrm{odd}}^{N-1} n\Psi(n/N)e^{-n^2t} \sin n\frac{i\pi}{N} \tag{3.22}$$

and

$$E_N(x_i,t) = \sum_{n=N+1,odd}^{\infty} \frac{e^{-n^2t}}{n^3} \sin n\frac{i\pi}{N}. \tag{3.23}$$

We will rigorously establish quartic convergence for fixed $t > 0$ by demonstrating spectral convergence for $E_N(x_i, t)$ and, at best, quartic convergence for $S_N(x_i, t)$. Starting with (3.23),

$$E_N(x_i, t) = \sum_{n=1,odd}^{\infty} \frac{e^{-(N+n)^2 t}}{(N+n)^3} \sin(N+n)x_i \tag{3.24}$$

$$= e^{-N^2 t} \sum_{n=1,odd}^{\infty} \frac{e^{-(2N+n^2)t}}{(N+n)^3} \sin(N+n)x_i. \tag{3.25}$$

Therefore,

$$|E_N(x_i, t)| < e^{-N^2 t} \sum_{n=1,odd}^{\infty} \frac{1}{(N+n)^3} < e^{-N^2 t} \int_N^{\infty} \frac{ds}{s^3} \tag{3.26}$$

and finally

$$|E_N(x_i, t)| < \frac{e^{-N^2 t}}{2N^2}. \tag{3.27}$$

Now consider the asymptotic behavior of $S_N(x_i, t)$. From (3.22),

(3.28)

$$\|S_N(x_i, t)\|_2 = \frac{1}{N^4} \left\{ \sum_{n=1,odd}^{N-1} [n\Psi(n/N)e^{-n^2 t}]^2 \right\}^{1/2} > \Psi(0)\frac{e^{-t}}{N^4} \approx 0.4058712\frac{e^{-t}}{N^4},$$

where the last inequality is from the earlier observation that $\Psi(s)$ is monotone increasing on the interval [0,1] and consequently $\Psi(1/N) > \Psi(0) \approx 0.4058712$.

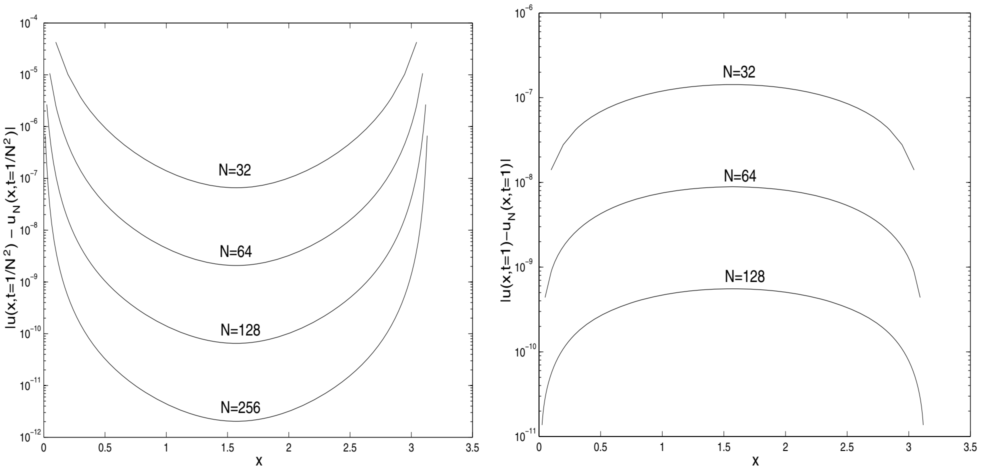
This demonstrates that convergence in the l_2 norm is at best quartic. In contrast to Figure 3.2(a), Figure 3.2(b) shows that for any fixed t, the maximum error, as measured by $|e_N|$, occurs in the middle of the domain at $x = \pi/2$ as opposed to the endpoint, $x = \pi/N$, for small t. The reason the error curves change as t becomes large is that the exponential term $e^{-n^2 t}$ dominates, and the error can be approximated by the lowest wavenumber, $n = 1$. Thus, for large t, the error is essentially proportional to $\sin x$ which has a maximum at $\pi/2$. As will be seen in the next section, numerical experiments demonstrate that convergence in the l_∞ norm is also quartic for t bounded away from $(0, 0)$ and $(0, \pi)$.

3.4. Comparison with finite difference methods. Given the nonuniform convergence of the Fourier method, as illustrated by quadratic convergence for small t and quartic convergence for large t, it is natural to compare its performance with fourth and second-order finite difference methods that require less computation. The finite difference approximations are derived by substituting

$$u_i = \sum_{n=1,\text{odd}}^{N-1} a_n(t) \sin nx_i \tag{3.29}$$

into

$$u_{i_t} = \frac{u_{i-1} - 2u_i + u_{i+1}}{\delta x^2}, \tag{3.30}$$



(a) (b)

FIG. 3.2. (a) $|e_N(x, N^{-2})|$ *on the interval* $x\epsilon(0,\pi)$ *for* $t = 1/N^2$. (b) $|e_N(x,1)|$ *on the interval* $x\epsilon(0,\pi)$ *for* $t = 1$.

where δx is π/N, yielding

$$a_n(t) = c_N(n)e^{\lambda_n^2 t}, \qquad \text{where} \qquad \lambda_n^2 = \frac{-4N^2\sin^2(\frac{n\pi}{2N})}{\pi^2}. \tag{3.31}$$

Likewise, the solution to the fourth-order finite difference approximation

$$u_{i_t} = \frac{-u_{i-2} + 16u_{i-1} - 30u_i + 16u_{i+1} - u_{i+2}}{12\delta x^2} \tag{3.32}$$

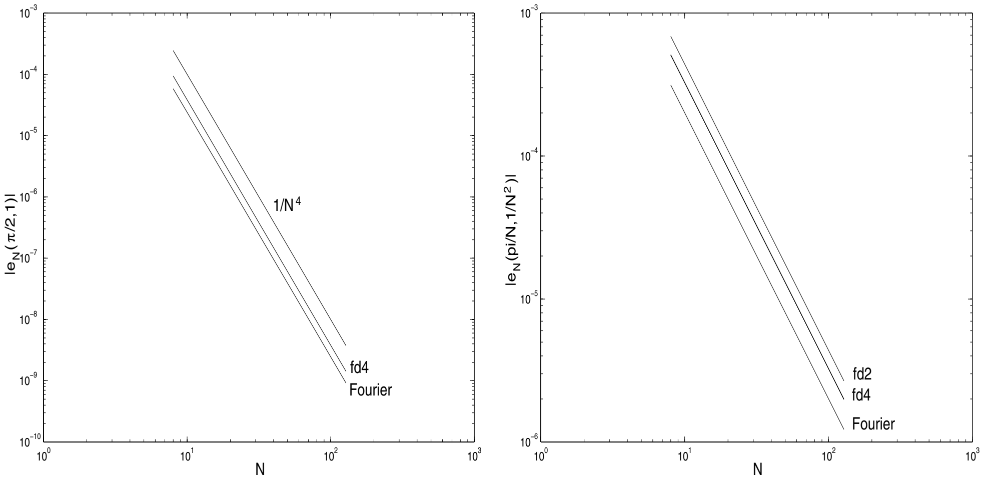
is given by (3.31), except now

$$\lambda_n^2 = \frac{-4N^2\sin^2(\frac{n\pi}{2N})}{3\pi^2}\left[3 + \sin^2\left(\frac{n\pi}{2N}\right)\right]. \tag{3.33}$$

The values of λ_n are the only difference between solutions to the finite difference and Fourier method for which $\lambda_n = n$.

However, if we compare the performance of the FD4 to the Fourier method for large t, it can be seen that there is little difference, as shown in Figure 3.3(a). For small t, a FD2 method performs just as well as the Fourier method and FD4 with only a slightly larger constant of proportionality, as seen in Figure 3.3(b). This marginal error reduction may not be sufficient to justify the expense of the Fourier method. However, it is reasonable to speculate that a FD4 scheme on a Chebyshev grid could give comparable results to the Chebyshev method that yields an $O(N^{-4})$ convergence error near the endpoints, as will be seen in the next section.

3.5. Comparison with the Chebyshev method. The method of choice for the interpolation of nonperiodic functions on a finite interval is the Chebyshev method. Unlike the Fourier method, the Chebyshev method does yield spectral convergence



(a) (b)

Fig. 3.3. (a) *The error of a fourth-order centered finite difference method versus the error in the spectral approximation for* $x = \pi/2$, $t = 1$. (b) *The error of a fourth-order and second-order centered finite difference method versus the error in the spectral approximation for* $x = \pi/N$ *and* $t = 1/N^2$.

for fixed t. The reason is simply that the cosine series representation of a smooth IC does not alias. Nevertheless, convergence in the neighborhood of $(0,0)$ and $(\pi,0)$ is algebraic. To see why this is the case, we implement the Chebyshev method by expanding the solution in terms of Chebyshev cardinal functions (see [1]).

The error for the Chebyshev method, as a function of space, is qualitatively similar to Figures 3.2(a) and 3.2(b), with the maximum error occurring at the endpoints for small t and in the middle of the domain for large t. The error as a function of time is similar to Figure 3.1 in that it reaches a maximum and then decays, but it differs in the fact that the decay becomes exponential after the maximum as seen in Figure 3.4. However, there exists a trajectory along which the max error converges algebraically as shown in Figure 3.5(a). The difference is that the max error is converging quartically as opposed to quadratically, as seen for the Fourier case. The increase in accuracy occurs because the Chebyshev grid is quadratically clustered near the endpoints where the singularities are occurring, giving an extra factor of $O(1/N^2)$. In contrast to the Fourier method for large t, the Chebyshev method does indeed give the expected exponential rate of convergence for the error, as indicated in Figure 3.5(b). This is a direct result of the Chebyshev method's ability to represent the IC exactly with only three polynomials.

Thus unlike the Fourier method, at time $t = 0$, there is no initial error in the coefficients due to truncation.

For the parabolic case, we can see that the convergence rate of the error in the maximum norm is nonuniform and algebraic. In general, as long as we are interested in solutions that are bounded away from singularities induced by violating the compatibility conditions, the spectral method *will* yield spectral convergence. However, if we are interested in heat transients such as those that occur on a microprocessor,

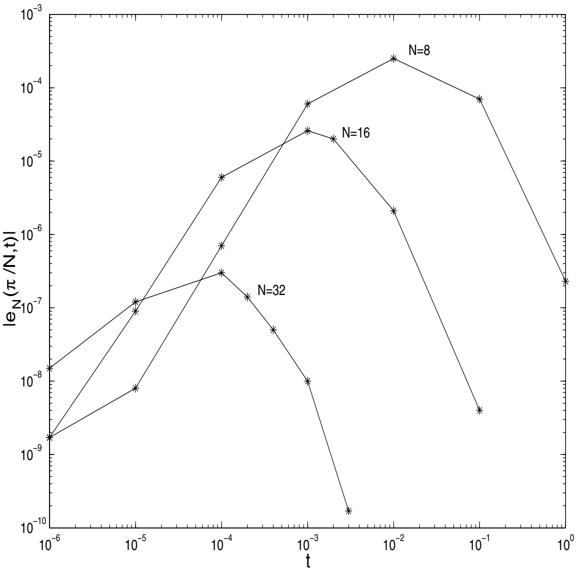
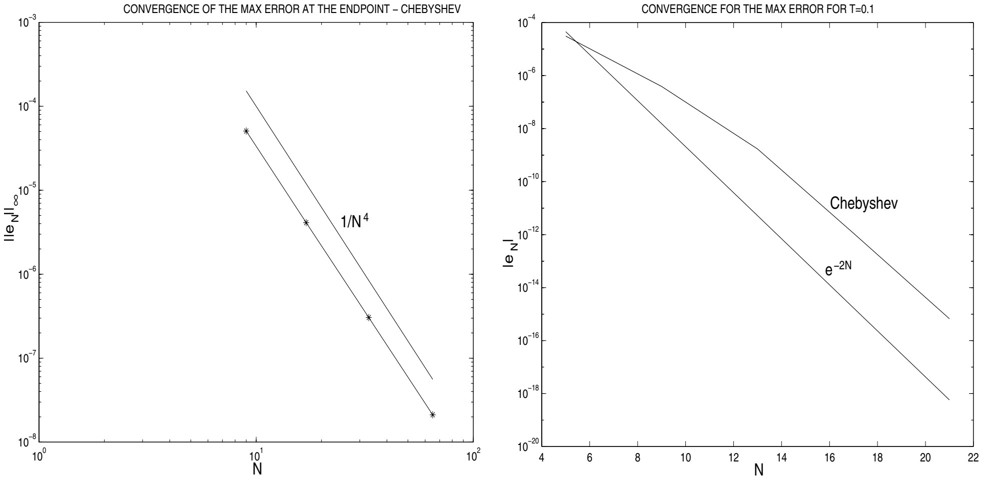


FIG. 3.4. $|e_N|$ *for the Chebyshev method on the interval* $t\epsilon(0,1]$ *for* $x = \pi/N$ *and fixed* N.



(a) (b)

FIG. 3.5. (a) *The max error in the Chebyshev approximation at the endpoint* $x = \pi/N$ *as a function of resolution* N. (b) *The error in the Chebyshev approximation for* $x = \pi/2$ *and* $t = 0.1$.

a computationally more efficient method with comparable algebraic convergence may be more attractive.

4. The hyperbolic case. In contrast to the parabolic case, a hyperbolic operator will propagate the solution through the time-space domain. As a result, singularities induced by incompatible BCs and the IC are not smoothed as t increases.

To observe the effect on the convergence of the error for spectral methods, we will consider the one-dimensional wave equation

$$u_{tt} = u_{xx} \tag{4.1}$$

subject to BCs

$$u(0, t) = u(\pi, t) = 0 \tag{4.2}$$

and ICs

$$u(x, 0) = \frac{\pi}{8} x(\pi - x), \qquad u_t(x, 0) = 0. \tag{4.3}$$

The first-order compatibility condition, (2.6) for u_0 and $k = 1$, is not satisfied. The difference in the analysis of section 3.1 is that the time dependence of the exact and approximate spectral solution is given by an oscillatory function. Thus,

$$u(x_i, t) = \sum_{n=1,\text{odd}}^{\infty} \frac{\cos(nt)}{n^3} \sin nx_i \tag{4.4}$$

with the analogue to (3.15) being

$$u_N(x_i, t) = \sum_{n=1,\text{odd}}^{N-1} \cos(nt) \left[\frac{1}{n^3} - \frac{n}{N^4} \Psi\left(\frac{n}{N}\right) \right] \sin nx_i. \tag{4.5}$$

4.1. Global quadratic convergence. The decay rate of the error for hyperbolic cases is more dramatic than for parabolic problems. The error induced by violating the first-order compatibility condition for a hyperbolic problem not only produces an $O(N^{-2})$ error for the Fourier method in the neighborhood of $(0, 0)$ and $(\pi, 0)$ but propagates throughout the domain, resulting in global quadratic convergence in the maximum norm. The propagation of the error through the x–t plane is along the characteristic $x = t$ and $x = \pi - t$, as shown in Figures 4.1(a) and 4.1(b) for $x \in [0, \pi]$ and $t \in [0, \pi]$.

The proof of quadratic convergence of the error for small t follows that given in section 3.2. For large t, the proof is the same except the error is defined along the discrete trajectory $x_{N/2} = \pi/2$ and $t_N = \pi/2 - 1/N$. Numerical experiments, given in Figure 4.2, show that convergence *is* quadratic along both of these trajectories, demonstrating the global nature of the error propagation.

4.2. The Chebyshev method. Next, we compare the performance of a Chebyshev method with the Fourier method that yields quadratic convergence. As noted in section 3.5, we use Chebyshev cardinal functions. Figure 4.3 shows the propagation of the error through the x–t plane for the Chebyshev method. As expected, we see that the error propagates along the characteristics $t = 1 \pm x$. The convergence of the error along the characteristics for any value of t differs significantly from the parabolic case, where spectral convergence was obtained for $t > 0$. The convergence of the error, as shown in Figure 4.4, is *quadratic*, no better than the Fourier series expansion and requiring $O(N^2)$ operations. At first, one might suspect that the convergence rate would be $O(N^4)$, the same as was observed for the heat equation in the neighborhood

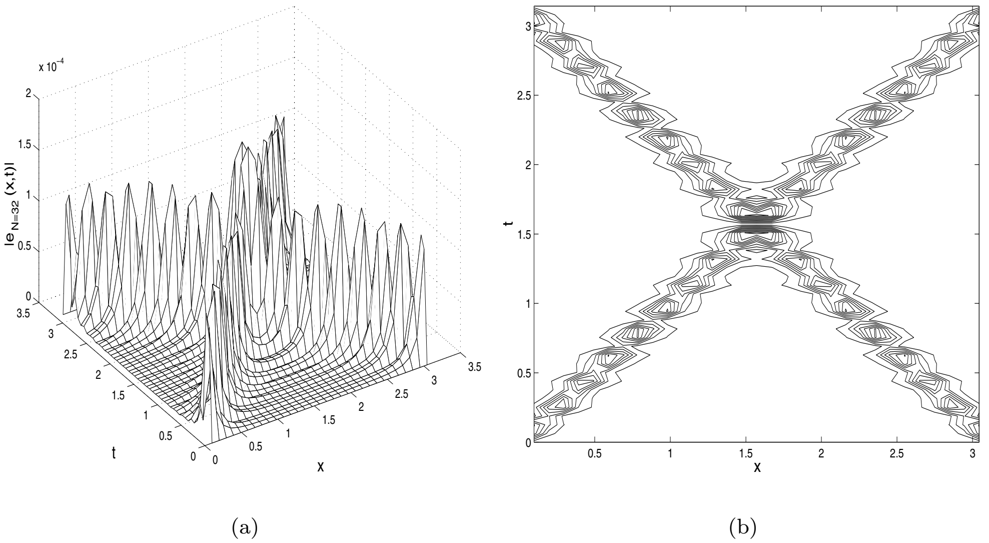


FIG. 4.1. (a) *Propagation of the error through the x–t plane.* (b) *Contour plot of* (a).

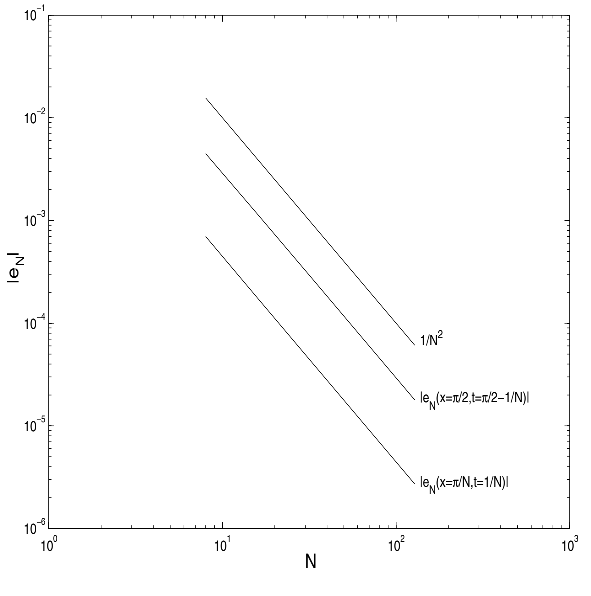


FIG. 4.2. *The convergence of the error in the Fourier method on the discrete trajectories $x = \pi/2$ and $t = \pi/2 - 1/N$, and $x = \pi/N$ and $t = 1/N$.*

of the discontinuities ($x = 0, t = 0$) and ($x = \pi, t = 0$). The difference, however, is that in the heat equation there is only a jump discontinuity in the second derivative of the solution at $t = 0$. For any $t > 0$, the discontinuity becomes smoothed out due to the diffusive operator. Thus, in the neighborhood of the singularities for $0 < t < \epsilon$, the original jump discontinuities become steep gradients that are analytically continuous.

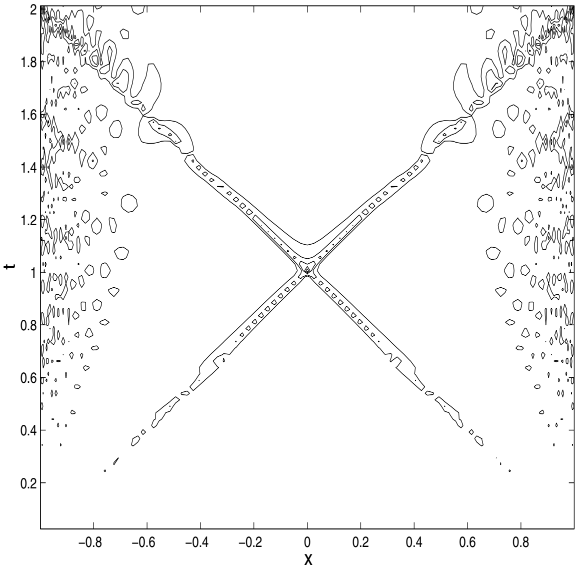


FIG. 4.3. *The propagation of the error in the Chebyshev approximation for $N = 128$.*

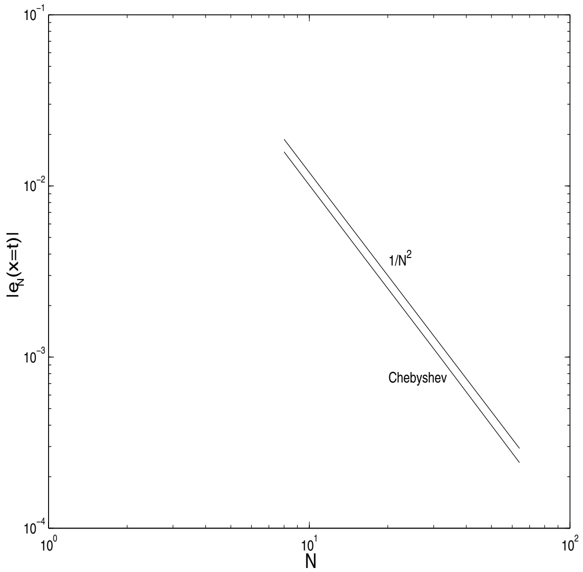


FIG. 4.4. *The convergence of the error in the Chebyshev approximation along the characteristic $t = x$.*

Spectral methods have difficulty resolving these gradients, but the Chebyshev method gives quartic convergence due to its quadratically clustered grid near the endpoints, as opposed to the uniform grid of the Fourier method. However, in the hyperbolic case, the discontinuities at $t = 0$ in the second derivative of the analytical solution are present for all time. Thus, the type of grid used is irrelevant, and *any* spectral method will converge quadratically.

4.3. Comparison with finite difference methods. The quadratic convergence of spectral methods naturally refocuses attention on second-order finite difference methods. The second-order centered finite difference method given by (3.30) yields the solution

$$u_N(x_i,t) = \sum_{n=1,odd}^{N} \cos(\lambda_n t)\left\{\frac{1}{n^3} - \frac{n}{N^4}\Psi(n/N)\right\}\sin nx_i, \tag{4.6}$$

where

$$\lambda_n = \frac{2N}{\pi}\sin\left(\frac{n\pi}{2N}\right) \tag{4.7}$$

is a second-order approximation to the eigenvalues n. The error is then given by

$$e_N(x_i,t) = u(x_i,t) - u_N(x_i,t) = T_N(x_i,t) + S_N(x_i,t) + E_N(x_i,t), \tag{4.8}$$

where

$$T_N(x_i,t) = \sum_{n=1,odd}^{N} \frac{\cos nt - \cos\lambda_n t}{n^3}\sin nx_i, \tag{4.9}$$

$$S_N(x_i,t) = \frac{1}{N^4}\sum_{n=1,\mathrm{odd}}^{N} n\Psi(n/N)\cos\lambda_n t \sin nx_i, \tag{4.10}$$

$$E_N(x_i,t) = \sum_{n=N+1,odd}^{\infty} \frac{\cos nt}{n^3}\sin nx_i. \tag{4.11}$$

Using these formulas, we will analytically prove as well as computationally show that not only does a second-order finite difference method *not* give quadratic convergence but rather converges only slightly better than linearly. The dominant error term is $T_N(x_i,t)$ which we demonstrate by first showing that

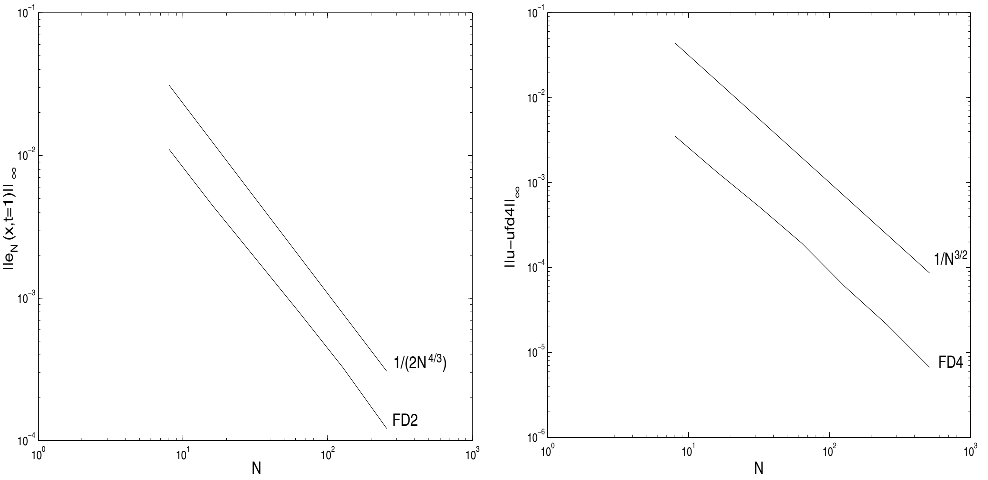
$$|S_N(x_i,t)| < O(N^{-2}) \qquad \text{and} \qquad |E_N(x_i,t)| < O(N^{-2}). \tag{4.12}$$

For $t = x$, we have

$$\begin{aligned} |S_N(t)| &= \frac{1}{2N^4}\Big|\sum_{n=1,\mathrm{odd}}^{N} n\Psi(n/N)\cos\lambda_n t \sin nt\Big| \\ &\le \frac{1}{2N^4}\sum_{n=1,\mathrm{odd}}^{N} |n\Psi(n/N)\cos\lambda_n t\sin nt| \\ &< \frac{1}{2N^2}. \end{aligned} \tag{4.13}$$

Similarly,

$$\begin{aligned} |E_N(t)| &= \frac{1}{2}\Big|\sum_{n=N+1,odd}^{\infty}\frac{\cos nt}{n^3}\sin 2nt\Big| \le \frac{1}{2}\sum_{n=N+1,odd}^{\infty}\Big|\frac{\cos nt}{n^3}\sin 2nt\Big| \\ &< \frac{1}{2}\int_N^{\infty}\frac{ds}{s^3} = \frac{1}{4N^2}. \end{aligned} \tag{4.14}$$



(a) (b)

FIG. 4.5. (a) *The max error in x for the second-order centered finite difference method at $t = 1$.* (b) *The max error in x for the fourth-order centered finite difference method at $t = 1$.*

Surprisingly, Figure 4.5 demonstrates that the error in a second-order finite difference method converges as $O(N^{-4/3})$. The proof for 4/3 convergence is rather detailed and deferred to Appendix A, where it is first shown that

$$|T_N(t)| \leq O(N^{-4/3}) \tag{4.15}$$

along the characteristic $t = x$ for $x\epsilon(0, \pi)$. To derive a lower bound is slightly more complicated, but nevertheless for any average value of t along the same characteristic,

$$\frac{1}{\pi}\int_0^\pi T_N(t)dt > \frac{C}{N^{4/3}}, \tag{4.16}$$

where C is a constant. These bounds hold for any second-order finite difference operator in which there is a discontinuity in the second derivative of the solution. The fact that the convergence of the second-order finite difference scheme is not quadratic may not be surprising since the approximation is based on Taylor series expansions that assume continuity of the function at least up to the fourth derivative. However, the fact that it is 4/3 is surprising.

For a variety of ICs that would correspond to violating different order compatibility conditions we propose the following conjecture for the upper bounds of the error term, $T_N(x_i, t)$, due to a second-order finite difference approximation.

Let

$$T_N(t) = \sum_{n=1,odd}^{N} \frac{\cos\lambda_{n,N}t - \cos nt}{n^3}\sin nt = \sum_{n=1,odd}^{N} b_{n,N}(t). \tag{4.17}$$

When we enforce BCs, we violate some order compatibility condition and the coefficients of the error, $|b_{n,N}(t)|$, will be algebraically decreasing due to the resulting discontinuity in a derivative of the Taylor expansion of the IC about the boundary. Thus, we can always impose the bound

$$|b_{n,N}(t)| \leq \frac{c}{n^{\beta}}, \qquad \beta \geq 1, \tag{4.18}$$

where c is a constant.

From (4.18) and generalizing the analysis in Appendix A we make the following conjecture whose derivation is relegated to Appendix B.

CONJECTURE. *The error term $T_N(x_i, t)$ is bounded from above by*

$$\begin{array}{lll} \sum_{n=1}^{N} |b_{n,N}(t)| & \leq & O(N^{(1-\beta)2/3}), \quad \beta < 4, \\ & \leq & O(N^{-2}\log N), \quad \beta = 4, \\ & \leq & O(N^{-2}), \qquad\quad \beta > 4. \end{array}$$

For $\beta < 4$, the error from the IC dominates, while for $\beta = 4$ both the error from the IC and second-order difference approximation contribute equally. In the last case the error from the finite difference operator dominates.

$\beta > 4$ implies that at least the fourth derivative of the IC is continuous. Thus, the best a second-order method can perform is $O(N^{-2})$. $\beta < 4$ implies that the function is not C^4, and therefore we would expect less than quadratic convergence since the error in a second-order method depends on the continuity of the fourth derivative of the function. However, the fact that $\beta = 4$ yields a convergence rate that is proportional to $\log(N)$ is not only new but quite surprising. The above conjecture has been supported by the numerical evidence.

For completeness, the convergence rate of the FD4 method that was used in section 3.4 is considered. However, we still see that convergence is less than quadratic and in fact appears to be only $O(N^{-3/2})$, as demonstrated in Figure 4.5(b). The reduction in the convergence rates of the finite difference methods demonstrates the significant impact of incompatibilities which occur anytime the BCs are independent of the IC. In general, violation of the compatibility conditions will likely lead to algebraic convergence of spectral methods as well as reduce the convergence rate of finite difference schemes.

5. Conclusions. In this paper, we have applied the theory of compatibility conditions to the numerical solutions of PDEs and have demonstrated how the subtle singularities inherent in IBVPs impact the convergence properties of spectral and finite difference methods. The temporal derivatives of the solution as defined by the initial condition, differential operator, and Cauchy–Kowalesky theorem will not equal those determined by the independent BCs. The resulting singularities in the corners of the temporal-spatial domain, which are independent of the smoothness of the IC, disrupt the spectral convergence of the error normally associated with spectral methods, in a manner that differs significantly between parabolic and hyperbolic systems.

For parabolic systems, the Chebyshev spectral method may be considered "self-healing," leading to a spectral rate of convergence for the error. However, this is not true for the Fourier method where aliasing is likely to induce an algebraic error in the initial Fourier representation. In the neighborhood of the singularities, all methods yield convergence rates that are algebraic. Thus, as measured in the maximum norm, the convergence of the approximate solution to the exact solution for spectral methods is both algebraic and nonuniform. Therefore, the usefulness of spectral methods may be diminished if we are interested in transient solutions for small t.

For hyperbolic systems, the situation is more interesting but worse because the singularities at $t = 0$ on the boundary propagate throughout the temporal-spatial domain along the characteristic lines $x \pm ct$. Thus, the algebraic rate of convergence for the error that existed in the neighborhood of the singularities for the parabolic case exists for all time in hyperbolic problems. As a result, the spectral method does not yield spectral convergence for hyperbolic IBVPs. This naturally draws attention to finite difference methods which require only $O(N)$ operations. However, it was shown that violation of the compatibility conditions can also reduce the order of convergence for finite difference schemes.

The above study details the impact of the theory of compatibility conditions on numerical calculations for IBVPs. Application is broad because many scientific problems are calculated on finite domains where independent BCs are imposed. As a result, singularities in the corners of the temporal-spatial domain are likely to disrupt the accuracy of numerical methods. It is in this regard that the above study has analyzed the convergence rate of the error for spectral and finite difference methods and explored their application to solving IBVPs.

Appendix A: Proof of 4/3 convergence. The term $T_N(x_i, t)$, which accounts for the error in the eigenvalue approximation, dominates the error of the second-order finite difference scheme. Our goal is therefore to bound the error from above and below to prove the 4/3 convergence rate that was computationally observed in section 4.3.

We will first prove the upper bound for the error term $T_N(x,t)$ along the characteristic $t = x$ for $x \epsilon (0, \pi)$

$$|T_N(t)| \leq O(N^{-4/3}). \tag{5.1}$$

Proof.

$$T_N(t) = \sum_{n=1,odd}^{N} \frac{\cos \lambda_{n,N} t - \cos nt}{n^3} \sin nt = \sum_{n=1,odd}^{N} b_{n,N}(t), \tag{5.2}$$

where

$$\lambda_{n,N} = n \frac{\sin(\frac{n\pi}{2N})}{\frac{n\pi}{2N}}. \tag{5.3}$$

Since the $\limsup_{n \to N} \cos(c(n)) \sin(d(n)) = 1$, we can immediately achieve an upper bound on the coefficients $b_{n,N}$

$$|b_{n,N}| \leq \frac{2}{n^3}. \tag{5.4}$$

This bound is due to the discontinuity in the second derivative of the IC from violating the compatibility conditions. However, because the series is decreasing slowly as $1/n^3$, this bound is only tight for the tail end of the series, when n is large. To describe the behavior for the initial terms (i.e., small n), we need to derive a secondary bound. This bound will come from the error in the second-order finite difference operator. Using trigonometric identities, we can rewrite

$$\cos \lambda_{n,N} t - \cos nt = 2 \sin \frac{n + \lambda_{n,N}}{2} t \sin \frac{n - \lambda_{n,N}}{2} t. \tag{5.5}$$

To find an upper bound on the product of the sine functions, we use the known result derived from the Taylor expansion of $\sin y/y$ for $y\epsilon[0,\pi]$ which gives

$$|\sin \frac{n-\lambda_{n,N}}{2}t| < \frac{n-\lambda_{n,N}}{2}t < \frac{\pi^2}{24}\frac{n^3}{N^2}t. \tag{5.6}$$

Therefore,

$$|b_{n,N}| \leq \frac{|(\cos\lambda_{n,N}t - \cos nt)\sin nt|}{n^3} \leq \frac{\pi^2}{24}\frac{t}{N^2}. \tag{5.7}$$

The trick now is to divide the summation in (5.2) into two parts,

$$\sum_{n=1,odd}^{N} b_{n,N}(t) = \sum_{n=1,odd}^{N^\gamma} b_{n,N}(t) + \sum_{n=N^\gamma+1,odd}^{N} b_{n,N}(t), \tag{5.8}$$

where $0<\gamma<1$. Equation (5.7) gives the tightest bound for the initial terms, while (5.4) gives a tighter bound for the tail end of the series. The bounds are minimized, giving equal contributions, for $\gamma = 2/3$. Therefore, we have

$$|T_N(t)| = |\sum_{n=1,odd}^{N} b_{n,N}(t)| \leq \sum_{n=1,odd}^{N^{2/3}} |b_{n,N}(t)| + \sum_{n=N^{2/3}+1,odd}^{N} |b_{n,N}(t)| \tag{5.9}$$

$$\leq N^{2/3}\frac{\pi^2}{24}\frac{t}{N^2} + \sum_{n=N^{2/3}+1,odd}^{N} \frac{2}{n^3} \tag{5.10}$$

$$\leq \frac{\pi^2}{24}\frac{t}{N^{4/3}} + \frac{1}{N^{4/3}} = \left(1+\frac{\pi^2}{24}t\right)N^{-4/3}. \qquad \square \tag{5.11}$$

For the second part of the proof, we show that $T_N(x,t)$ for any average value of t along the characteristic $t=x$ on $x\epsilon(0,\pi)$ is bounded from below by

$$\frac{1}{\pi}\int_0^\pi T_N(t)dt > \frac{C}{N^{4/3}}. \tag{5.12}$$

Proof. Beginning with

$$\frac{1}{\pi}\int_0^\pi T_N(t)dt = \frac{1}{\pi}\sum_{n=1,odd}^{N}\int_0^\pi \frac{\cos nt - \cos\lambda_n t}{n^3}\sin nt, \tag{5.13}$$

then by using trigonometric identities and integrating, we have

$$\frac{1}{\pi}\int_0^\pi T_N(t)dt = \frac{1}{2}\sum_{n=1,odd}^{N}\frac{f(\frac{n(1+\lambda_n/n)}{2}\pi) + f(\frac{n(1-\lambda_n/n)}{2}\pi)}{n^3} \tag{5.14}$$

$$\geq \frac{1}{2}\sum_{n=1,odd}^{N^{2/3}}\frac{f(\frac{n(1-\lambda_n/n)}{2}\pi)}{n^3}, \tag{5.15}$$

where

$$f(y) = \frac{1-\cos(y)}{y}. \tag{5.16}$$

Next, we will bound the argument of f which is of the form $1-\frac{\sin y}{y}$ as shown below. By inspection, we can show that on $y\epsilon(0,\pi)$, $y^2/12 < 1-\frac{\sin y}{y} < y^2/6$, which yields the following bounds:

$$\frac{n(1-\lambda_n/n)}{2}\pi = \frac{\pi}{2}n\left(1-\frac{\sin\frac{n\pi}{2N}}{\frac{n\pi}{2N}}\right) \tag{5.17}$$

$$< \frac{n\pi}{2}\frac{\frac{n\pi}{2N}}{6} = \frac{\pi^3}{48}\frac{n^3}{N^2} \tag{5.18}$$

$$> \frac{n\pi}{2}\frac{\frac{n\pi}{2N}}{12} = \frac{\pi^3}{96}\frac{n^3}{N^2}. \tag{5.19}$$

In the range $0 < y < 1$, $f(y)$ can be bounded by its argument $1/4y < f(y) < 1/2y$. Thus, for $n < N^{2/3}$

$$\frac{1}{4}\frac{n(1-\lambda_n/n)}{2}\pi < f\left(\frac{n(1-\lambda_n/n)}{2}\pi\right) \tag{5.20}$$

which from (5.19) implies

$$\frac{\pi^3 n^3}{384N^2} < f\left(\frac{n(1-\lambda_n/n)}{2}\pi\right). \tag{5.21}$$

Therefore, from (5.15)

$$\frac{1}{\pi}\int_0^\pi T_N(t)dt > \frac{\pi^3}{768N^2}\sum_{n=1,\text{odd}}^{N^{2/3}} 1 > \frac{\pi^3}{1536N^{4/3}}. \qquad \square \tag{5.22}$$

Appendix B: General upper bounds on the error term $T_N(t)$. In general, when we have to enforce BCs we will violate some order compatibility condition, and the coefficients of the error will be algebraically decreasing due to the resulting discontinuity in a derivative of the Taylor expansion of the IC about the boundary. Thus, we can always impose the bound

$$|b_{n,N}(t)| \le \frac{c}{n^\beta}, \qquad \beta \ge 1, \tag{5.23}$$

where c is a constant. From the analysis in Appendix A, we can generalize (5.7), the error bound induced by the second-order difference operator, for any IC as

$$|b_{n,N}(t)| \le \frac{c_2 tn(\frac{n}{N})^2}{n^\beta} = c_2 t\frac{n^{3-\beta}}{N^2}. \tag{5.24}$$

If we split the summation which would correspond to $T_N(x_i,t)$ into two parts as was done in (5.8), we achieve the following approximation:

$$\sum_{n=1}^{N^\gamma} c_2 t\frac{n^{3-\beta}}{N^2} \sim \begin{array}{ll} \frac{c_2 t}{4-\beta}N^{-2}N^{(4-\beta)\gamma}, & \beta < 4,\\ c_2 tN^{-2}\log N^\gamma, & \beta = 4,\\ c_3 tN^{-2}, & \beta > 4. \end{array}$$

Similarly, for the tail end of the series as N becomes large, we have, using (5.23),

$$\sum_{n=N^\gamma+1,odd}^{\infty} \sim \frac{c}{\beta-1}N^{\gamma(1-\beta)}. \tag{5.25}$$

From these approximations, the conjecture in section 4.3 follows for the upper bound on the error term, $T_N(x_i,t)$, due to the second-order finite difference scheme.

Acknowledgments. The authors gratefully thank Dr. Keith Lindsay at the National Center for Atmospheric Research for his discussions, help, and contributions, particularly with the convergence proofs in the appendices and the factorizations (5.8) and (5.9).

REFERENCES

[1] J.P. Boyd, *Chebyshev and Fourier Spectral Methods*, Springer-Verlag, New York, 1989.

[2] J.P. Boyd and N. Flyer, *Compatibility conditions for time-dependent partial differential equations and the rate of convergence of Chebyshev and Fourier spectral methods*, Comput. Methods Appl. Mech. Engrg., 175 (1999), pp. 281–309.

[3] A. Friedman, *Partial Differential Equations of Parabolic Type*, Prentice-Hall, Englewood Cliffs, NJ, 1964.

[4] H.O. Kreiss, *Initial boundary value problems for hyperbolic systems*, Comm. Pure Appl. Math, 23 (1970), pp. 277–298.

[5] H.O. Kreiss, V. Thomee, and O. Widlund, *Smoothing of the initial data and rates of convergence for parabolic difference equations*, Comm. Pure Appl. Math., 23 (1970), pp. 241–259.

[6] H.O. Kreiss and J. Lorenz, *Initial-Boundary Value Problems and the Navier-Stokes Equations*, Academic Press, Boston, 1989.

[7] R. Hersh, *Mixed problems in several variables*, J. Math. Mech., 12 (1963), pp. 317–334.

[8] D. Gottlieb and S.A. Orszag, *Numerical Analysis of Spectral Methods: Theory and Applications*, SIAM, Philadelphia, 1977.

[9] B. Gustafsson, H.O. Kriess, and J. Oliger, *Time-Dependent Problems and Difference Methods*, Wiley-Interscience, New York, 1995.

[10] O. Ladyzenskaja, *On the convergence of Fourier series defining a solution of a mixed problem for hyperbolic equations*, Dokl. Akad. Nauk SSSR, 85 (1952), pp. 481–484.

[11] O. Ladyzenskaja, *On the solvability of the fundamental boundary problems for equations of parabolic and hyperbolic type*, Dokl. Akad. Nauk SSSR, 87 (1954), pp. 395–398.

[12] O. Ladyzenskaja, V. Solonnikov, and N. Ural'ceva, *Linear and Quasi-Linear Equations of Parabolic Type*, Transl. Math. Monogr. 23, AMS, Providence, RI, 1968.

[13] A. Majda, J. McDonough, and S. Osher, *The Fourier method for nonsmooth initial data*, Math. Comp., 32 (1978), pp. 1041–1081.

[14] A. Majda and S. Osher, *Propagation of error into regions of smoothness for accurate difference approximations to hyperbolic equations*, Comm. Pure Appl. Math., 30 (1977), pp. 671–705.

[15] S. Osher, *Smoothing for spectral methods*, in Spectral Methods for Partial Differential Equations, SIAM, Philadelphia, 1984, pp. 209–216.

[16] J.B. Rauch and F.J. Massey, *Differentiability of solutions to hyperbolic initial-boundary value problems*, Trans. Amer. Math. Soc., 189 (1974), pp. 303–318.

[17] R. Sakamoto, *Hyperbolic Boundary Value Problems*, Cambridge University Press, Cambridge, UK, 1982.

[18] S. Smale, *Smooth solutions of the heat and diffusion equations*, Comment. Math. Helv., 55 (1980), pp. 1–12.

[19] V.I. Smirnov, *A Course of Higher Mathematics. Integral Equations and Partial Differential Equations*, Vol. 4, Pergamon Press, London, 1964.

[20] R. Temam, *Behaviour at time $t = 0$ of the solutions of semi-linear evolution equations*, J. Differential Equations, 43 (1982), pp. 73–92.

[21] V. Thomee, *Numerical Solution of Partial Differential Equations* II, Academic Press, 1970, pp. 585–622.

[22] V. Thomee and L. Wahlbin, *Convergence rates of parabolic difference schemes for non-smooth data*, Math. Comp., 28 (1974), pp. 1–13.

SIAM J. SCI. COMPUT.
Vol. 23, No. 5, pp. 1752–1776

DECOUPLING THREE-DIMENSIONAL MIXED PROBLEMS USING DIVERGENCE-FREE FINITE ELEMENTS[*]

ROBERT SCHEICHL[†]

Abstract. In this paper we describe an iterative method for indefinite saddle-point systems arising from mixed finite element discretizations of second-order elliptic boundary value problems subject to mixed boundary conditions and posed over polyhedral three-dimensional domains. The method is based on a decoupling of the vector of velocities in the saddle-point system from the vector of pressures, resulting in a symmetric positive definite velocity system and a triangular pressure system.

The crucial step in this approach is the construction of the divergence-free Raviart–Thomas–Nédélec elements from the curls of Nédélec's edge elements. Because of the large kernel of the curl-operator, this representation is not unique. To find a basis we consider the graph made up of the nodes and edges of the mesh and eliminate the edge elements associated with a spanning tree in this graph. To prove that this technique works in the general case considered here, we employ fundamental results from algebraic topology and graph theory.

We also include some numerical experiments, where we solve the (decoupled) velocity system by ILU-preconditioned conjugate gradients and the pressure system by simple back substitutions. We compare our method with a standard ILU-based block preconditioner for the original saddle-point system, and we find that our method is faster by a factor of at least 4.5 in all cases, with the greatest improvement occurring in the nonuniform mesh case.

Key words. mixed finite elements, second-order elliptic problems, mixed boundary conditions, divergence-free space, spanning trees, decoupled iterative method

AMS subject classifications. 65N22, 65N30, 65F10, 05C05, 55U10

PII. S1064827500375886

1. Introduction.

The problem we are going to consider in this paper is the following second-order elliptic problem in velocity-pressure formulation

$$\vec{u} + K\,\vec{\nabla} p = \vec{g}, \tag{1.1}$$

$$\vec{\nabla} \cdot \vec{u} = 0, \tag{1.2}$$

subject to mixed boundary conditions over a polyhedral three-dimensional domain Ω. Such a problem arises, for example, in groundwater flow or oil recovery simulations, where $\vec{u}$ corresponds to the velocity, p corresponds to the pressure, and K is permeability divided by dynamic viscosity.

The numerical treatment of (1.1), (1.2) involves the solution of usually very large indefinite linear equation systems. In this paper we describe a very efficient and practicable iterative method to solve these systems by decoupling the vector of velocities from the vector of pressures, resulting in a symmetric positive definite velocity system and a triangular pressure system. The crucial step in this approach is the construction of a basis for the divergence-free Raviart–Thomas–Nédélec elements. The proof that our algorithm for this construction works uses results from algebraic topology and graph theory and will also be presented in this paper.

[*] Received by the editors July 24, 2000; accepted for publication (in revised form) February 15, 2001; published electronically January 30, 2002. This work was supported by EPSRC CASE Award 97D00023 in collaboration with AEA Technology.

http://www.siam.org/journals/sisc/23-5/37588.html

[†] Institut Francais du Pétrole, DISMA, 1 et 4 ave. de Bois-Préau, 92852 Rueil-Malmaison, France (robert.scheichl@ifp.fr).

Because the variable of prime interest in (1.1), (1.2) (especially in the applications we have in mind) is the velocity $\vec{u}$, the discretization schemes of most interest are those which preserve conservation of mass (1.2) in an appropriate way, with the prime candidates being mixed finite element or finite volume techniques. In this paper we discretize (1.1), (1.2) using the lowest-order mixed Raviart–Thomas–Nédélec elements on tetrahedral meshes (Nédélec [23]). Here p is approximated in the space of piecewise constant functions and $\vec{u}$ is approximated in an appropriate subspace of the vector-valued piecewise linear functions, in which the normal component of $\vec{u}$ is required to be continuous across the element boundaries. The resulting discretization enforces mass conservation on each element of the mesh. Since the quality of the approximations is determined by the mesh width, it is usually necessary to work with very fine meshes.

As is well known this type of discretization yields symmetric indefinite systems of *saddle-point type*. Iterative methods for indefinite systems are less powerful and robust (w.r.t. a refinement of the mesh) than the methods available for definite systems. Therefore almost all approaches to solve this system efficiently contain at some point a reduction of the system to a symmetric positive definite system. At least two different strategies have been pursued.

The first strategy is to solve the saddle-point system by a preconditioned *minimum residual* (MINRES) method using a symmetric positive definite block preconditioner. The analysis for this strategy seems to be restricted to the two-dimensional case (e.g., [26, 4]), although the preconditioner described in Rusten and Winther [26] can be readily applied in three dimensions as well. There is also recent work by Wohlmuth, Toselli, and Widlund [29] on a domain decomposition preconditioner for Raviart–Thomas–Nédélec vector fields in three dimensions which can be used in the framework of Arnold, Falk, and Winther [4].

The second strategy is to decouple the vector of velocities in the saddle-point system from the vector of pressures. This can be done by (i) mixed hybridization via Lagrange multipliers [13, 11], (ii) block elimination of the velocity variable [24], or, as presented in this paper, (iii) a direct elimination of the divergence constraint (1.2) on the element level. This technique (iii) has two distinct advantages over (i) and (ii): first, no nonphysical variables are introduced, and, second, the velocity is obtained directly without (necessarily) computing the pressure, the latter advantage being particularly attractive in groundwater flow calculations. The method (iii) was first developed in the related but different case of the Stokes problem by Crouzeix and Thomasset [28] for two dimensions and by Hecht [17] for three dimensions. In connection with the solution of system (1.1), (1.2) it appears first in Chavent et al. [10]. Since the decoupling in this way leads to much smaller and nicer linear equation systems, it was subsequently possible to develop very competitive and efficient methods for the two-dimensional case (e.g., [15, 16, 21, 22, 12]). In this paper we present a very efficient iterative method for the solution of system (1.1), (1.2) that implements this idea in three dimensions.

Thus our solver is built on three essential steps. The first step decouples the velocity field in the saddle-point problem from the pressure field. This is done by writing the velocity as the curl of an appropriate discrete *vector potential*, automatically satisfying the discrete counterpart of the mass conservation law (1.2). The required discrete vector potential turns out to be a finite element approximation of the solution of a related symmetric positive *semidefinite* problem by *edge elements* (Nédélec [23]) which can be found independently of the pressure. Because of the large kernel of the curl-operator, this system is singular. In Hiptmair and Hoppe [19] a multilevel method is constructed that solves this singular problem approximately. In this paper we make the problem positive definite by eliminating the degrees of freedom associated with a *spanning tree* of the graph made up of the nodes and edges of the mesh. This also corresponds to finding a local basis for the divergence-free Raviart–Thomas–Nédélec elements. Such algebraic techniques have already been successfully used in other fields (e.g., [2, 20, 6] for Maxwell's equations, [17, 18, 14] for incompressible flow (Stokes)); in the context of (1.1), (1.2) there is only one paper by Cai et al. [8], and that is restricted to uniform rectangular meshes, in which case the special spanning tree can be written down a priori.

The second step in the solver is the application of a preconditioned *conjugate gradient* (CG) method to solve for the discrete velocity. Since the system is symmetric positive definite, it is easier to solve than the original saddle-point system. It also turns out to have the added advantage of being about three times smaller than the original system. In section 7 we have included some results using a simple ILU preconditioner to illustrate this advantage. We compare our method with the (also ILU-based) block preconditioner of Rusten and Winther [26] for the original saddle-point system, and we find that our method is faster by a factor of at least 4.5 in all cases, with the greatest improvement occurring in the nonuniform mesh case (where an improvement factor of 9.0 is observed on the finest mesh (36864 freedoms)).

The third and final step in the solver is the recovery of the pressure (if it is required). The decoupled pressure system turns out to be particularly simple. By an appropriate numbering of the freedoms it can be made triangular and solved in optimal time by simple *back substitutions*. We will prove this rigorously.

The layout of this paper is as follows. In section 2 we shall describe the mathematical setting for (1.1), (1.2) and its discretization by mixed finite elements. In section 3 we present the decoupling of the vector of velocities from the vector of pressures as a general algebraic procedure. In section 4 we construct the basis for the divergence-free Raviart–Thomas–Nédélec elements, and we show in section 5 how this basis is used to implement the decoupled velocity system. In section 6 we show how the pressure is recovered, and we finish the paper with some numerical results in section 7.

2. Mixed finite element discretization.

In this section we describe the mathematical setting for (1.1), (1.2) together with its discretization by mixed finite elements. Since this is a standard procedure (see, e.g., [7]), we shall be brief.

Let Ω denote an open *polyhedron*, i.e. a simply connected open domain without cavities in $\mathbb{R}^3$ with a connected boundary Γ composed of plane faces, which is assumed partitioned into $\Gamma_D \cup \Gamma_N$. Each of Γ_D and Γ_N is assumed to consist of a finite union of planar polygonal subsets of Γ, and Γ_D and Γ_N are both assumed to be connected. Additionally, Γ_N is assumed to be closed. Let $\vec{\nu}(\vec{x})$ denote the outward unit normal from Ω at $\vec{x} \in \Gamma$. In general we assume that K is a bounded, symmetric, and uniformly positive definite 3×3 matrix-valued function on Ω. The system (1.1), (1.2) is to be solved on Ω subject to mixed boundary conditions:

$$p = p_D \quad \text{on} \quad \Gamma_D \qquad \text{and} \quad \vec{u} \cdot \vec{\nu} = 0 \quad \text{on} \quad \Gamma_N \,. \tag{2.1}$$

Throughout, we shall assume that $\Gamma_D \neq \emptyset$, a condition which is generically satisfied in groundwater flow applications, where some inflow and outflow must occur. The extension to the case when $\Gamma_D = \emptyset$ (when p is nonunique) can easily be made by imposing an extra condition on p in the weak form below (e.g., that p should have a prescribed mean value; see, e.g., [15]).

To discretize (1.1), (1.2), (2.1) we put it in weak form. Let $(\cdot,\cdot)_{L_2(\Omega)^d}$ denote the usual inner product in $L_2(\Omega)^d$ for $d = 1, 2, 3$. Then introduce the Hilbert space

$$H(\mathrm{div}, \Omega) := \{\vec{v} \in L_2(\Omega)^3 : \mathrm{div}\, \vec{v} \in L_2(\Omega)\},$$

with the inner product

$$(\vec{u}, \vec{v})_{H(\mathrm{div},\Omega)} := (\vec{u}, \vec{v})_{L_2(\Omega)^3} + (\mathrm{div}\, \vec{u}, \mathrm{div}\, \vec{v})_{L_2(\Omega)},$$

and its subspace

$$H_{0,N}(\mathrm{div}, \Omega) := \{\vec{v} \in H(\mathrm{div}, \Omega) : \vec{v} \cdot \vec{\nu}|_{\Gamma_N} = 0\}$$

(see [7] for details). Introduce the bilinear forms

$$m(\vec{u}, \vec{v}) := (K^{-1}\vec{u}, \vec{v})_{L_2(\Omega)^3}, \qquad b(\vec{v}, w) := -(\mathrm{div}\, \vec{v}, w)_{L_2(\Omega)},$$

and the linear functional

$$G(\vec{v}) := (K^{-1}\vec{g}, \vec{v})_{L_2(\Omega)^3} - \int_{\Gamma_D} p_D\, \vec{v} \cdot \vec{\nu}\, dF.$$

Then the weak form of (1.1), (1.2), (2.1) is to find $(\vec{u}, p) \in H_{0,N}(\mathrm{div}, \Omega) \times L_2(\Omega)$ such that

$$(2.2) \qquad \left\{ \begin{array}{lcllll} m(\vec{u}, \vec{v}) & + & b(\vec{v}, p) & = & G(\vec{v}) & \text{for all } \vec{v} \in H_{0,N}(\mathrm{div}, \Omega), \\ b(\vec{u}, w) & & & = & 0 & \text{for all } w \in L_2(\Omega). \end{array} \right.$$

The mixed finite element discretization of (2.2) is obtained by choosing finite-dimensional subspaces $\mathcal{V} \subset H_{0,N}(\mathrm{div}, \Omega)$ and $\mathcal{W} \subset L_2(\Omega)$ and seeking $(\vec{U}, P) \in \mathcal{V} \times \mathcal{W}$ such that

$$(2.3) \qquad \left\{ \begin{array}{lcllll} m(\vec{U}, \vec{V}) & + & b(\vec{V}, P) & = & G(\vec{V}) & \text{for all } \vec{V} \in \mathcal{V}, \\ b(\vec{U}, W) & & & = & 0 & \text{for all } W \in \mathcal{W}. \end{array} \right.$$

In practice this is implemented by choosing bases $\{\vec{v}_i : i = 1, \ldots, n_{\mathcal{V}}\}$ and $\{w_j : j = 1, \ldots, n_{\mathcal{W}}\}$ for $\mathcal{V}$ and $\mathcal{W}$. By writing

$$\vec{U} = \sum_{i=1}^{n_{\mathcal{V}}} u_i \vec{v}_i, \qquad P = \sum_{j=1}^{n_{\mathcal{W}}} p_j w_j,$$

problem (2.3) is then reduced to the indefinite system of linear equations

$$(2.4) \qquad \begin{pmatrix} M & B \\ B^T & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{p} \end{pmatrix} = \begin{pmatrix} \boldsymbol{g} \\ \boldsymbol{0} \end{pmatrix} \qquad \text{in} \quad \mathbb{R}^{n_{\mathcal{V}}} \times \mathbb{R}^{n_{\mathcal{W}}},$$

where $M_{i,i'} := m(\vec{v}_i, \vec{v}_{i'})$ is the "mass matrix," $B_{i,j} := b(\vec{v}_i, w_j)$ is the "discrete gradient," and $g_i := G(\vec{v}_i)$.

In this paper we restrict attention to the (most practically important) case when $\mathcal{V}$ is the lowest-order Raviart–Thomas–Nédélec space on tetrahedra [23]. To define this, let $\mathcal{T}$ denote a triangulation of Ω into conforming tetrahedra $T \in \mathcal{T}$. We assume that all lines along which the boundary condition changes (i.e., the boundaries of the components of Γ_N) are edges of tetrahedra in $\mathcal{T}$. Let $\mathcal{F}$ denote the set of all faces of the tetrahedra in $\mathcal{T}$. It is convenient to think of these faces as open so that, for

$F \in \mathcal{F}$, $\overline{F}$ denotes the closure of F (including its boundary). For any $F \in \mathcal{F}$, we let $\vec{\nu}_F$ denote the unit normal to the face F which, for convenience, is assumed to be orientated so that

$$\vec{\nu}_F \in \{\vec{x} \in \mathbb{R}^3 : x_1 > 0\} \cup \{(0, x_2, x_3)^T \in \mathbb{R}^3 : x_2 > 0\} \cup \{(0,0,1)^T\}. \tag{2.5}$$

Let $\mathcal{F}_I$, $\mathcal{F}_D$, and $\mathcal{F}_N$ denote the faces $F \in \mathcal{F}$ which lie in Ω, Γ_D, and Γ_N, respectively.

The space $\mathcal{V}$ is defined to be the space of all functions $\vec{v} \in H_{0,N}(\text{div}, \Omega)$ such that for all $T \in \mathcal{T}$, there exist $\vec{\alpha}_T \in \mathbb{R}^3$ and $\gamma_T \in \mathbb{R}$ such that

$$\vec{v}(\vec{x}) = \vec{\alpha}_T + \gamma_T \vec{x} \qquad \text{for all} \quad \vec{x} \in T. \tag{2.6}$$

Equivalently, we can define $\mathcal{V}$ to be the space of all $\vec{v} : \Omega \to \mathbb{R}^3$ which satisfy (2.6) for each $T \in \mathcal{T}$, and also

$$\begin{array}{ll} \text{(i)} & \vec{v} \cdot \vec{\nu}_F \quad \text{is continuous across each face } F \in \mathcal{F}_I, \\ \text{(ii)} & \vec{v} \cdot \vec{\nu}_F = 0 \quad \text{for all} \quad F \in \mathcal{F}_N. \end{array} \tag{2.7}$$

Because of the special form of (2.6) it is easily shown that $\vec{v}(\vec{x}) \cdot \vec{\nu}_F$ is constant for $\vec{x} \in F$ on any face F of T. Thus $\vec{v} \in \mathcal{V}$ can be completely determined by specifying the constant value of $\vec{v} \cdot \vec{\nu}_F$ for each $F \in \mathcal{F}_I \cup \mathcal{F}_D$. This leads us to introduce the standard basis for $\mathcal{V}$ which is constructed by associating with each face $F \in \mathcal{F}_I \cup \mathcal{F}_D$, a function $\vec{v}_F \in \mathcal{V}$ with the property that

$$\vec{v}_F \cdot \vec{\nu}_{F'} = \delta_{F,F'}, \tag{2.8}$$

with δ denoting the Kronecker delta.

We also have to specify the space $\mathcal{W}$. To fulfill the discrete inf-sup condition which is necessary for existence and uniqueness (see, e.g., [25]), $\mathcal{W}$ is chosen as the space of piecewise constant functions on Ω, with the basis consisting of the characteristic functions w_T of each of the tetrahedra $T \in \mathcal{T}$. Thus

$$n_{\mathcal{V}} = (\#\mathcal{F}_I + \#\mathcal{F}_D), \qquad n_{\mathcal{W}} = (\#\mathcal{T}), \tag{2.9}$$

where, throughout, $\#A$ denotes the number of elements of a (finite) set A.

3. Decoupled iterative method for mixed problems. In this section we formulate our method for decoupling the vector of velocities $\boldsymbol{u}$ from the vector of pressures $\boldsymbol{p}$ in system (2.4). We have already presented this procedure for the two-dimensional case in [12]. Recall [7] that (2.4) has a unique solution $(\boldsymbol{u}, \boldsymbol{p}) \in \mathbb{R}^{n_{\mathcal{V}}} \times \mathbb{R}^{n_{\mathcal{W}}}$ for all $\boldsymbol{g} \in \mathbb{R}^{n_{\mathcal{V}}}$, and clearly $\boldsymbol{u}$ is in $\ker B^T$.

Remark 3.1. *The case of* $\boldsymbol{u} \notin \ker B^T$ *(or, equivalently, of a more general right-hand side* $(\boldsymbol{g}^T, \boldsymbol{h}^T)^T$ *of* (2.4)*) arises when* $\vec{\nabla} \cdot \vec{u} \neq 0$ *in* (1.2)*. This problem can be reduced to problem* (2.4) *following Ewing and Wang* [15] *(see also* [8, 19] *for three dimensions): in a preprocessing step based on domain decomposition (*static condensation*) a vector* $\boldsymbol{u}^*$ *is calculated such that* $B^T\boldsymbol{u}^* = \boldsymbol{h}$*; this can be done in* $O(n)$ *steps (where* $n = n_{\mathcal{V}} + n_{\mathcal{W}}$*); the remainder* $\tilde{\boldsymbol{u}} = \boldsymbol{u} - \boldsymbol{u}^*$ *fulfills* (2.4) *with right-hand side* $((\boldsymbol{g} - M\boldsymbol{u}^*)^T, \boldsymbol{0}^T)^T$ *and can be calculated with the method described in this paper. See* [27] *for details.*

To describe our decoupling procedure, first consider (2.4) as an abstract system. The decoupling of $\boldsymbol{u}$ from $\boldsymbol{p}$ can be achieved by finding

$$\text{a basis } \; \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{\mathring{n}}\} \; \text{ of } \; \ker B^T. \tag{3.1}$$

(Since B^T has full rank, $\mathring{n} = n_{\mathcal{V}} - n_{\mathcal{W}}$.) If we have such a basis, then the solution $\boldsymbol{u}$ of (2.4) can be written

$$\boldsymbol{u} = \sum_{j=1}^{\mathring{n}} \mathring{u}_j \boldsymbol{z}_j = Z^T \mathring{\boldsymbol{u}}, \tag{3.2}$$

for some $\mathring{\boldsymbol{u}} \in \mathbb{R}^{\mathring{n}}$, where Z denotes the $\mathring{n} \times n_{\mathcal{V}}$ matrix with rows $\boldsymbol{z}_1^T, \dots, \boldsymbol{z}_{\mathring{n}}^T$. Also, since $ZB = (B^T Z^T)^T = 0$, multiplying the first (block) row of (2.4) by Z shows that $\mathring{\boldsymbol{u}}$ is a solution of the linear system

$$\mathring{A}\mathring{\boldsymbol{u}} = \mathring{\boldsymbol{g}}, \tag{3.3}$$

where

$$\mathring{A} = ZMZ^T \quad \text{and} \quad \mathring{\boldsymbol{g}} = Z\boldsymbol{g}. \tag{3.4}$$

Since M is symmetric positive definite, so is $\mathring{A}$, and $\mathring{\boldsymbol{u}}$ is the unique solution of (3.3). Thus if the basis (3.1) can be found, then the velocity $\boldsymbol{u}$ in (2.4) can be computed by solving the decoupled positive definite system (3.3) rather than the indefinite coupled system (2.4).

In applications to groundwater flow, where one is primarily interested in the velocity $\vec{u}$ in (1.1), (1.2), the method described above is of great relevance. Even when the pressure p is also of interest our method may still be highly competitive, provided we can also compute a *complementary basis* $\{\boldsymbol{z}_{\mathring{n}+1}, \dots, \boldsymbol{z}_{n_{\mathcal{V}}}\}$ with the property that

$$\operatorname{span}\{\boldsymbol{z}_1, \dots, \boldsymbol{z}_{\mathring{n}}, \boldsymbol{z}_{\mathring{n}+1}, \dots, \boldsymbol{z}_{n_{\mathcal{V}}}\} = \mathbb{R}^{n_{\mathcal{V}}}. \tag{3.5}$$

If this is known and if Z' denotes the matrix with rows $\boldsymbol{z}_{\mathring{n}+1}^T, \dots, \boldsymbol{z}_{n_{\mathcal{V}}}^T$, then multiplying the first (block) row of (2.4) by Z' shows that $\boldsymbol{p}$ is the solution of the $n_{\mathcal{W}} \times n_{\mathcal{W}}$ system

$$(Z'B)\boldsymbol{p} = Z'(\boldsymbol{g} - M\boldsymbol{u}). \tag{3.6}$$

An elementary argument shows that $Z'B$ is nonsingular, and so the unique solution $\boldsymbol{p}$ of (3.6) also determines the pressure in (2.4) once the velocity $\boldsymbol{u}$ is known.

We show in the next three sections that in the particular case of the mixed finite element system (2.4),

(i) it is always easy to find the basis (3.1);

(ii) the resulting symmetric positive definite matrix $\mathring{A}$ in the reduced problem (3.3) can be obtained by simple algebraic techniques from the stiffness matrix of an associated symmetric positive semidefinite problem in the space $H(\overrightarrow{\text{curl}}, \Omega)$ discretized by Nédélec's edge elements;

(iii) the system (3.3) is about 3 times smaller than (2.4);

(iv) a simple choice of complementary basis can be made so that the coefficient matrix $Z'B$ in the system (3.6) is lower triangular.

To establish conclusions (i)–(iv) we need to exploit the particular properties of (2.4). In particular, note that finding the basis $\boldsymbol{z}_1, \dots, \boldsymbol{z}_{\mathring{n}}$ in (3.1) is equivalent to finding a basis $\mathring{\vec{v}}_1, \dots, \mathring{\vec{v}}_{\mathring{n}}$ of the finite element space

$$\mathring{\mathcal{V}} := \{\vec{V} \in \mathcal{V} : \; b(\vec{V}, W) = 0 \text{ for all } \; W \in \mathcal{W}\}.$$

To see why, suppose $\boldsymbol{z}_1, \dots, \boldsymbol{z}_{\mathring{n}}$ are known and let $Z = (Z_{i,j})$ be the matrix with rows $\boldsymbol{z}_1^T, \dots, \boldsymbol{z}_{\mathring{n}}^T$. Then the formulae

$$\mathring{\vec{v}}_i = \sum_{j=1}^{n_{\mathcal{V}}} Z_{i,j} \vec{v}_j \;, \qquad i = 1, \dots, \mathring{n}, \tag{3.7}$$

(where $\{\vec{v}_j\}$ is the basis of $\mathcal{V}$) determine the basis $\{\vec{\mathring{v}}_i\}$. Conversely, if the basis $\{\vec{\mathring{v}}_i\}$ of $\mathring{\mathcal{V}}$ is known, then the matrix Z (and hence the basis $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{\mathring{n}}$ of ker B^T) is determined by (3.7).

We now turn our attention to finding a basis of $\mathring{\mathcal{V}}$.

4. Construction of a divergence-free basis. As a first step, recall that $\mathcal{T}$ is the set of all tetrahedra in the mesh and that $\mathcal{F} = \mathcal{F}_I \cup \mathcal{F}_D \cup \mathcal{F}_N$ is the set of all faces of the mesh (assumed to be open triangles) which lie in Ω, Γ_D, and Γ_N, respectively. Analogously, we can write $\mathcal{E} = \mathcal{E}_I \cup \mathcal{E}_D \cup \mathcal{E}_N$, with $\mathcal{E}_I$, $\mathcal{E}_D$, and $\mathcal{E}_N$ denoting the edges in Ω, Γ_D, and Γ_N; and $\mathcal{N} = \mathcal{N}_I \cup \mathcal{N}_D \cup \mathcal{N}_N$, with $\mathcal{N}_I$, $\mathcal{N}_D$, and $\mathcal{N}_N$ denoting the nodes in Ω, Γ_D, and Γ_N. Recall that the boundaries of each of the components of Γ_N belong to Γ_N, and, since the lines between Neumann and Dirichlet boundaries are edges of the mesh, these edges lie in $\mathcal{E}_N$. For $E \in \mathcal{E}$, let $\vec{\tau}_E$ denote the unit tangent on edge E which, as in (2.5), is assumed to be orientated so that

$$\vec{\tau}_E \in \{\vec{x} \in \mathbb{R}^3 : x_1 > 0\} \cup \{(0, x_2, x_3)^T \in \mathbb{R}^3 : x_2 > 0\} \cup \{(0,0,1)^T\}. \tag{4.1}$$

Through this convention we associate an orientation with each edge of the mesh.

To construct a basis of $\mathring{\mathcal{V}}$ it is useful to introduce the following space of finite elements introduced by Nédélec in [23]. Let

$$H(\vec{\text{curl}}, \Omega) := \{\vec{\Phi} \in L_2(\Omega)^3 : \vec{\text{curl}}\, \vec{\Phi} \in L_2(\Omega)^3\}$$

and let $\mathcal{U}$ be the finite-dimensional space of all functions $\vec{\Phi} \in H(\vec{\text{curl}}, \Omega)$ such that for all $T \in \mathcal{T}$, there exist $\vec{\alpha}_T$, $\vec{\beta}_T \in \mathbb{R}^3$ such that

$$\vec{\Phi}(\vec{x}) = \vec{\alpha}_T + \vec{\beta}_T \times \vec{x} \qquad \text{for all } \vec{x} \in T. \tag{4.2}$$

In fact, $\mathcal{U}$ is the lowest-order member of the family of spaces introduced by Nédélec in [23]. The standard basis of $\mathcal{U}$ consists of the set of functions $\{\vec{\Phi}_E \in \mathcal{U} : E \in \mathcal{E}\}$ which are required to have the property

$$\int_{E'} \vec{\Phi}_E \cdot \vec{\tau}_{E'}\, ds = \delta_{E,E'} \qquad \text{for all } E' \in \mathcal{E}. \tag{4.3}$$

This choice of basis functions accounts for the widely used term *edge elements.*

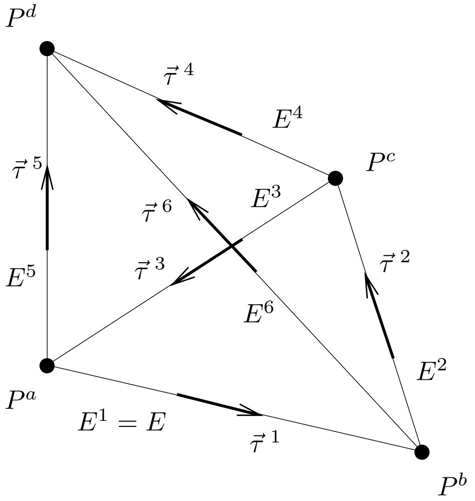
The basis for $\mathring{\mathcal{V}}$ will now be constructed from the fundamental functions $\vec{\Psi}_E$ defined by

$$\vec{\Psi}_E = \vec{\text{curl}} \vec{\Phi}_E \tag{4.4}$$

(so that $\vec{\Phi}_E$ is the *vector potential* of $\vec{\Psi}_E$). The functions (4.4) clearly satisfy $\text{div}\vec{\Psi}_E = 0$ on each tetrahedron of the mesh, and a subset of them lie in $\mathring{\mathcal{V}}$ as the following proposition shows.

PROPOSITION 4.1. *For each $E \in \mathcal{E}_I \cup \mathcal{E}_D$, $\vec{\Psi}_E \in \mathring{\mathcal{V}}$.*

Proof. Consider a general edge $E \in \mathcal{E}$. Conditions (4.2) and (4.3) clearly imply that supp $\vec{\Psi}_E$ consists only of the tetrahedra touching edge E. A typical such tetrahedron T with edges $E := E^1$, E^2, …, E^6, and nodes P^a, …, P^d, is depicted in the figure below:



with $\vec{\tau}^{\,\alpha}$, $\alpha = 1, \dots, 6$, denoting unit tangent vectors in the directions shown and $\vec{r}^{\,\beta}$ denoting the position vector of P^β, $\beta = a, b, c, d$. The faces are denoted by F^a, ..., F^d, where F^β is opposite P^β, $\beta = a, b, c, d$, and the unit outward normal on each face is denoted by $\vec{\nu}^{\,\beta}$.

Note first that for all $\vec{x} \in T$

$$\vec{\Psi}_E(\vec{x}) = \vec{\text{curl}}\vec{\Phi}_E(\vec{x}) = \vec{\nabla} \times \left(\vec{\beta}_T \times \vec{x}\right) = 2\vec{\beta}_T,$$

which is easily seen to be of the form (2.6) (in fact, with $\gamma_T = 0$).

Since $\vec{\Psi}_E(\vec{x})$ is constant on T, we can write for each $\vec{x} \in T$ and for each $\beta = a, b, c, d$,

$$\vec{\Psi}_E(\vec{x}) \cdot \vec{\nu}^{\,\beta} = \frac{1}{|F^\beta|} \int_{F^\beta} \vec{\Psi}_E(\vec{x}) \cdot \vec{\nu}^{\,\beta}\, dF = \frac{1}{|F^\beta|} \int_{F^\beta} \vec{\text{curl}}\vec{\Phi}_E(\vec{x}) \cdot \vec{\nu}^{\,\beta}\, dF,$$

and using Stokes's integral theorem we get

$$\vec{\Psi}_E(\vec{x}) \cdot \vec{\nu}^{\,\beta} = \frac{1}{|F^\beta|} \oint_{\partial F^\beta} \vec{\Phi}_E(\vec{x}) \cdot \vec{ds} = \begin{cases} \frac{1}{|F^\beta|} & \text{for } \beta = c, \\ -\frac{1}{|F^\beta|} & \text{for } \beta = d, \\ 0 & \text{otherwise,} \end{cases} \tag{4.5}$$

where in the last step we used (4.3) to evaluate the line integral (respecting the right-hand rule and the specific orientation of $\vec{\tau}^{\,1}$ and $\vec{\nu}^{\,\beta}$, $\beta = a, b, c, d$, as depicted in the figure above).

Now to obtain the result observe that, since $\text{div}\vec{\Psi}_E = 0$ on each tetrahedron, it is sufficient to show that

$$\vec{\Psi}_E \in \mathcal{V} \quad \text{for all } E \in \mathcal{E}_I \cup \mathcal{E}_D. \tag{4.6}$$

To show this we shall verify criterion (2.7). First consider $E \in \mathcal{E}_I$. Let $F \in \mathcal{F}_I$. If $F \not\subset \text{supp}\,\vec{\Psi}_E$, then we have trivially

$$\vec{\Psi}_E \cdot \vec{\nu}_F \quad \text{is continuous across } F. \tag{4.7}$$

Now take a general tetrahedron $T \subset \text{supp}\,\vec{\Psi}_E$, as pictured above. If $F = F^c$ or F^d, then performing the computation (4.5) in the other tetrahedron adjoining F and

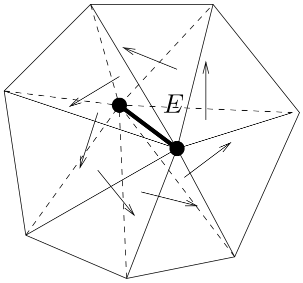


FIG. 1. *Divergence-free basis function* $\vec{\Psi}_E$.

combining with (4.5) establishes (4.7). On the other hand, when $F = F^a$ or F^b, (4.7) also holds since $\vec{\Psi}_E \cdot \vec{\nu}_F|_T = 0$ and since the other tetrahedron adjoining F lies outside $\operatorname{supp} \vec{\Psi}_E$. Altogether, we have established that $\vec{\Psi}_E$ satisfies criterion (2.7)(i).

To establish (2.7)(ii), let $F \in \mathcal{F}_N$. If $F \not\subset \operatorname{supp} \vec{\Psi}_E$, then $\vec{\Psi}_E \cdot \vec{\nu}_F = 0$ trivially. If $F \subset \overline{T} \subset \operatorname{supp} \vec{\Psi}_E$, then (since $E \in \mathcal{E}_I$) with the above notation F has to be either F^a or F^b and again $\vec{\Psi}_E \cdot \vec{\nu}_F = 0$, proving (2.7)(ii).

Thus we have shown that $\vec{\Psi}_E \in \mathcal{V}$ for all $E \in \mathcal{E}_I$. Similar arguments establish that $\vec{\Psi}_E \in \mathcal{V}$ for all $E \in \mathcal{E}_D$, proving (4.6). □

Note that each $\vec{\Psi}_E$ can be expressed as a local linear combination of the basis functions $\vec{v}_F$ of $\mathcal{V}$ satisfying (2.8); in fact, only those $\vec{v}_F$ corresponding to faces F that contain edge E appear in the expansion of $\vec{\Psi}_E$ (see Figure 1).

To find a basis for $\overset{\circ}{\mathcal{V}}$, let us first look at the pure Dirichlet case, $\Gamma_N = \emptyset$. The functions introduced in Proposition 4.1 are sufficient to span $\overset{\circ}{\mathcal{V}}$, but there are too many of them. The following theorem identifies a linearly independent subset of the functions in Proposition 4.1 that constitutes a basis of $\overset{\circ}{\mathcal{V}}$. A similar statement for the pure Neumann case, $\Gamma_D = \emptyset$, has already been proved by Dubois [14] (see Remark 4.7 for a more extensive literature survey).

The proof involves some fundamental notions and results from graph theory and algebraic topology (see Appendices A and B for a brief introduction). In particular we need the notion of a *spanning tree* of a graph (see Theorem A.4). Let $\mathbf{G} := (\mathcal{N}, \mathcal{E})$ be the graph formed by the nodes and (orientated) edges of the triangulation $\mathcal{T}$.

THEOREM 4.2. *Let* $\Gamma_N = \emptyset$ *and let* $\mathcal{H} \subset \mathcal{E}$ *be such that if* $\mathbf{H} := (\mathcal{N}, \mathcal{H})$ *is a spanning tree of* $\mathbf{G}$*, then*

$$\{\vec{\Psi}_E : E \in \mathcal{E} \backslash \mathcal{H}\} \quad \textit{is a basis of} \quad \overset{\circ}{\mathcal{V}}. \tag{4.8}$$

Before proving Theorem 4.2, we will first prove two lemmas. Let $\mathcal{V}(\mathbf{G})$ denote the vector space over $\mathbb{Z}$ generated by the cycles of $\mathbf{G}$ as defined in Definition A.1(e). Furthermore, for each face $F \in \mathcal{F}$ let μ^F be the *elementary cycle* of $\mathbf{G}$ formed by the edges E of F. We fix the orientation of this cycle w.r.t. $\vec{\nu}_F$ by applying the right-hand rule. The associated vector $\boldsymbol{\mu}^F := [\mu^F_E]_{E \in \mathcal{E}} \in \mathcal{V}(\mathbf{G})$ is given by

$$\mu^F_E = \begin{cases} 1 & \text{if } E \text{ is an edge of } F \text{ and } \vec{\tau}_E \text{ is positively orientated w.r.t. } \vec{\nu}_F, \\ -1 & \text{if } E \text{ is an edge of } F \text{ and } \vec{\tau}_E \text{ is negatively orientated w.r.t. } \vec{\nu}_F, \\ 0 & \text{otherwise.} \end{cases} \tag{4.9}$$

LEMMA 4.3. *Let $\boldsymbol{\mu} := [\mu_E]_{E\in\mathcal{E}} \in \mathcal{V}(\mathbf{G})$. Then there exist $\{\alpha_F \in \mathbb{Z} : F \in \mathcal{F}\}$ such that*

$$\boldsymbol{\mu} := \sum_{F\in\mathcal{F}} \alpha_F \boldsymbol{\mu}^F. \tag{4.10}$$

Proof. Let K be the simplicial complex underlying our simplicial triangulation $\mathcal{T}$. In the notation of algebraic topology (see Appendix B) the vector $\boldsymbol{\mu} \in \mathcal{V}(\mathbf{G})$ can be identified with the vector of coefficients of a cycle μ of K (with orientation of its edges defined by the tangent vectors $\vec{\tau}_E$).

Since $|K| = \overline{\Omega}$ is simply connected, we know from Corollary B.4 that each cycle of K is a bounding cycle and can therefore be written as a linear combination of the boundaries of the orientated triangles of K. In particular, there exist $\{\tilde{\alpha}_F \in \mathbb{Z} : F \in \mathcal{F}\}$ such that

$$\mu = \sum_{F\in\mathcal{F}} \tilde{\alpha}_F \partial F. \tag{4.11}$$

The boundary ∂F of an orientated triangle F of K is a special cycle $\tilde{\mu}^F$ of K. As above it can therefore be identified with a vector $\tilde{\boldsymbol{\mu}}^F \in \mathcal{V}(\mathbf{G})$. Depending on the orientation of $\vec{\nu}_F$ we either have $\tilde{\boldsymbol{\mu}}^F = \boldsymbol{\mu}^F$ or $\tilde{\boldsymbol{\mu}}^F = -\boldsymbol{\mu}^F$, and we can write (4.11) in vector notation:

$$\boldsymbol{\mu} = \sum_{F\in\mathcal{F}} \alpha_F \boldsymbol{\mu}^F \qquad \text{with} \quad \alpha_F = \begin{cases} \tilde{\alpha}_F & \text{if} \quad \tilde{\boldsymbol{\mu}}^F = \boldsymbol{\mu}^F, \\ -\tilde{\alpha}_F & \text{if} \quad \tilde{\boldsymbol{\mu}}^F = -\boldsymbol{\mu}^F. \end{cases} \qquad \square$$

LEMMA 4.4. *Let $\boldsymbol{\mu} \in \mathcal{V}(\mathbf{G})$ and let $\{\alpha_F \in \mathbb{Z} : F \in \mathcal{F}\}$ be such that $\boldsymbol{\mu} := \sum_{F\in\mathcal{F}} \alpha_F \boldsymbol{\mu}^F$. Then*

$$\sum_{F\in\mathcal{F}} \alpha_F \int_F \vec{\Psi}_E \cdot \vec{\nu}_F \, dF = \mu_E \qquad \textit{for all } E \in \mathcal{E}. \tag{4.12}$$

Proof. Let $F \in \mathcal{F}$. Using (4.5) we get

$$\int_F \vec{\Psi}_E \cdot \vec{\nu}_F \, dF = \begin{cases} 1 & \text{if } E \subset \overline{F} \text{ and } \vec{\tau}_E \text{ positively orientated w.r.t. } \vec{\nu}_F, \\ -1 & \text{if } E \subset \overline{F} \text{ and } \vec{\tau}_E \text{ negatively orientated w.r.t. } \vec{\nu}_F, \\ 0 & \text{otherwise,} \end{cases}$$

and, therefore, recalling the definition (4.9), we have $\int_F \vec{\Psi}_E \cdot \vec{\nu}_F \, dF = \mu_E^F$. Multiplying this by α_F and summing over $F \in \mathcal{F}$ we obtain (4.12). $\square$

We can now prove Theorem 4.2.

Proof. Let us first check that the number of basis functions in (4.8) coincides with $\mathring{n} = \dim \mathring{\mathcal{V}}$. Since Ω is simply connected without cavities, we can apply *Euler's polyhedron theorem* [9]

$$\#\mathcal{N} - \#\mathcal{E} + \#\mathcal{F} - \#\mathcal{T} \;=\; 1 \tag{4.13}$$

to the triangulation $\mathcal{T}$. Now observe that $\mathbf{H}$ is a tree, and therefore $\#\mathcal{H} = \#\mathcal{N} - 1$ (cf. Theorem A.3(iii)). Using this fact together with (4.13) we get

$$\#(\mathcal{E}\backslash\mathcal{H}) = \#\mathcal{E} - \#\mathcal{N} + 1 = \#\mathcal{F} - \#\mathcal{T}. \tag{4.14}$$

Now recalling that $\mathring{n} = n_{\mathcal{V}} - n_{\mathcal{W}} = \#\mathcal{F} - \#\mathcal{T}$, it follows from (4.14) that the number of functions in (4.8) is $\mathring{n}$, as required.

To establish linear independency of the functions in (4.8), suppose $\{\beta_{E'} : E' \in \mathcal{E}\backslash\mathcal{H}\}$ are scalars such that

$$\vec{0} = \sum_{E' \in \mathcal{E}\backslash\mathcal{H}} \beta_{E'} \vec{\Psi}_{E'}.$$

Now let $E \in \mathcal{E}\backslash\mathcal{H}$ and let $\boldsymbol{\mu}^E$ denote the vector associated with the unique cycle μ^E generated by taking edge E into the tree $\mathbf{H}$, which has the property that $\mu^E_{E'} := \delta_{E,E'}$ for all $E' \in \mathcal{E}\backslash\mathcal{H}$ (cf. Theorem A.6). Then using Lemma 4.3 we can find $\{\alpha_F \in \mathbb{Z} : F \in \mathcal{F}\}$ such that $\boldsymbol{\mu}^E := \sum_{F\in\mathcal{F}} \alpha_F \boldsymbol{\mu}^F$, and so by Lemma 4.4

$$0 = \sum_{F\in\mathcal{F}} \alpha_F \int_F \left(\sum_{E' \in \mathcal{E}\backslash\mathcal{H}} \beta_{E'}\, \vec{\Psi}_{E'} \right) \cdot \vec{\nu}_F \, dF = \sum_{E' \in \mathcal{E}\backslash\mathcal{H}} \beta_{E'} \mu^E_{E'} = \beta_E,$$

which establishes the linear independency of the functions in (4.8). □

Now let us look at mixed boundary conditions, $\Gamma_N \neq \emptyset$. In the following corollary we will see that the results of Theorem 4.2 extend to this case provided each component of Γ_N is simply connected. Our proof of this result makes use of the methods of Hecht [17] developed for the nonconforming P1-P0 elements for the approximation of solenoidal vector fields in $H^1(\Omega)^3$ (see Remark 4.7 for a more extensive discussion).

Therefore let $n_{\mathcal{C}}$ denote the number of connected components in Γ_N and write

$$\Gamma_N = \Gamma_N^1 \cup \Gamma_N^2 \cup \cdots \cup \Gamma_N^{n_{\mathcal{C}}}, \qquad \Gamma_N^\ell \cap \Gamma_N^{\ell'} = \emptyset \quad \text{for all } \ell \neq \ell' \in \{1, \ldots, n_{\mathcal{C}}\}.$$

For $\ell = 1, \ldots, n_{\mathcal{C}}$, let $\mathcal{N}_N^\ell \subset \mathcal{N}$, $\mathcal{E}_N^\ell \subset \mathcal{E}$, and $\mathcal{F}_N^\ell \subset \mathcal{F}$ denote the set of mesh nodes, edges, and faces on Γ_N^ℓ, respectively.

Corollary 4.5. *Suppose $n_{\mathcal{C}} \neq 0$ and suppose that Γ_N^ℓ is simply connected for each $\ell = 1, \ldots, n_{\mathcal{C}}$. Let $\mathcal{H} \subset \mathcal{E}$ such that $\mathbf{H} = (\mathcal{N}, \mathcal{H})$ is a spanning tree of $\mathbf{G}$ and such that for each $\ell = 1, \ldots, n_{\mathcal{C}}$, the restriction $\mathbf{H}_N^\ell := (\mathcal{N}_N^\ell, \mathcal{H} \cap \mathcal{E}_N^\ell)$ of $\mathbf{H}$ to Γ_N^ℓ is also a tree. Then*

$$\{\vec{\Psi}_E : E \in (\mathcal{E}_I \cup \mathcal{E}_D)\backslash\mathcal{H}\} \quad \textit{is a basis of} \quad \mathring{\mathcal{V}}. \tag{4.15}$$

Remark 4.6. *The general case, when Γ_N^ℓ is not simply connected for some ℓ, involves the introduction of a small number of additional nonlocal basis functions. In order not to complicate this paper, we omit the details for this case, but they will be given in* [27]. *Thus, from now on we will assume that Γ_N^ℓ is simply connected for all $\ell = 1, \ldots, n_{\mathcal{C}}$.*

Proof. Since $(\mathcal{E}_I \cup \mathcal{E}_D) \subset \mathcal{E}$, we also have $(\mathcal{E}_I \cup \mathcal{E}_D)\backslash\mathcal{H} \subset \mathcal{E}\backslash\mathcal{H}$, and therefore following the proof of Theorem 4.2 the functions $\vec{\Psi}_E$ in (4.15) are linearly independent.

We have only to check that the number of basis functions in (4.15) coincides with $\mathring{n} = \dim \mathring{\mathcal{V}}$. To do this, we need two elementary formulae (Cauchy [9]): *Euler's polyhedron theorem* (4.13) and the *Euler–Cauchy formula* for planar networks of polygons (4.16). Consider a typical Neumann boundary segment Γ_N^ℓ. Since Γ_N^ℓ is simply connected, we have

$$\#\mathcal{N}_N^\ell - \#\mathcal{E}_N^\ell + \#\mathcal{F}_N^\ell = 1. \tag{4.16}$$

Now observe that $\mathbf{H}_N^\ell$ is a tree, and therefore (again by virtue of Theorem A.3(iii)) $\#(\mathcal{H} \cap \mathcal{E}_N^\ell) = \#\mathcal{N}_N^\ell - 1$. Using (4.16) and summing over $\ell = 1, \dots, n_\mathcal{C}$, we obtain

$$\#(\mathcal{H} \cap \mathcal{E}_N) = \sum_{\ell=1}^{n_\mathcal{C}} (\#\mathcal{N}_N^\ell - 1) = \sum_{\ell=1}^{n_\mathcal{C}} (\#\mathcal{E}_N^\ell - \#\mathcal{F}_N^\ell) = (\#\mathcal{E}_N - \#\mathcal{F}_N). \tag{4.17}$$

Since the sets $\mathcal{E}_I$, $\mathcal{E}_D$, and $\mathcal{E}_N$ partition $\mathcal{E}$, we also have

$$(\mathcal{E}_I \cup \mathcal{E}_D)\backslash\mathcal{H} = (\mathcal{E}\backslash\mathcal{E}_N)\backslash\mathcal{H} = \mathcal{E}\backslash(\mathcal{E}_N \cup \mathcal{H}),$$

and, therefore, the number of functions in (4.15) is $\#\mathcal{E} - (\#\mathcal{E}_N + \#\mathcal{H} - \#(\mathcal{H} \cap \mathcal{E}_N))$. Combining this with (4.17), and using the fact that $\mathbf{H}$ is a tree (and therefore $\#\mathcal{H} = \#\mathcal{N} - 1$), we finally get

$$\#((\mathcal{E}_I \cup \mathcal{E}_D)\backslash\mathcal{H}) = (\#\mathcal{E} - \#\mathcal{N} + 1) - \#\mathcal{F}_N = (\#\mathcal{F} - \#\mathcal{T}) - \#\mathcal{F}_N, \tag{4.18}$$

where in the last step we have used Euler's polyhedron theorem (4.13). Now recalling that $\mathring{n} = n_\mathcal{V} - n_\mathcal{W}$ (and from section 3 we have $n_\mathcal{V} = \#\mathcal{F}_I + \#\mathcal{F}_D = \#\mathcal{F} - \#\mathcal{F}_N$ and $n_\mathcal{W} = \#\mathcal{T}$), it follows from (4.18) that the number of functions in (4.15) is $\mathring{n}$, as required. □

REMARK 4.7. *The idea of spanning trees in the context of finite element methods first appears in the context of the Stokes problem in a paper by Hecht* [17]*, where it is used in the same way as here to find a basis for the space of divergence-free nonconforming* $P1$-$P0$ *elements for the approximation of solenoidal vector fields in* $H^1(\Omega)^3$.

In an unpublished manuscript [18]*, Hecht extends these results to a wider family of finite elements in* $H^1(\Omega)^3$*, including the (nonconforming) Raviart–Thomas–Nédélec elements. The published literature on divergence-free Raviart–Thomas–Nédélec elements in* $H(\mathrm{div}, \Omega)$ *considered here is restricted to the pure Neumann case,* $\Gamma_D = \emptyset$*, in a paper by Dubois* [14]*, where he uses it to solve model incompressible flow problems with prescribed vorticity.*

In the context of the three-dimensional problem (1.1), (1.2) *considered in this paper, the only other work which we are aware of is the recent paper* [8]*, but this is restricted to uniform rectangular meshes and a special spanning tree which can be constructed a priori.*

Independently, spanning trees also appear as a technique for computing a discrete gauge condition in eddy-current calculations in computational electromagnetism (e.g., in Albanese and Rubinacci [2] *or Kettunen and Turner* [20]*). A thorough presentation of the theoretical foundation of those techniques using homology theory (related to our Appendix* B) *can be found in Bossavit* [6, Ch. 5].

5. Implementation. To implement the decoupled system (3.3) for determining $\mathring{\boldsymbol{u}}$ (and hence $\boldsymbol{u}$) we must work with the matrix $\mathring{A}$ and the right-hand side $\mathring{\boldsymbol{g}}$ specified in (3.4). We observe that these are formally defined in terms of multiplications with the matrix Z which, through (3.7), represents the basis $\{\vec{\mathring{v}}_i\}$ of $\mathring{\mathcal{V}}$ in terms of the basis $\{\vec{v}_j\}$ of $\mathcal{V}$.

In the specific system (2.4) the $\{\vec{v}_j\}$ are the Raviart–Thomas velocity basis functions $\{\vec{v}_F : F \in \mathcal{F}_I \cup \mathcal{F}_D\}$ given in section 2, whereas the $\{\vec{\mathring{v}}_i\}$ are the basis functions $\{\vec{\Psi}_E : E \in (\mathcal{E}_I \cup \mathcal{E}_D)\backslash\mathcal{H}\}$ specified in Corollary 4.5 (or Theorem 4.2, if $\Gamma_N = \emptyset$). Thus we can identify the columns of Z with the indices $F \in \mathcal{F}_I \cup \mathcal{F}_D$, whereas the rows of Z correspond to $E \in (\mathcal{E}_I \cup \mathcal{E}_D)\backslash\mathcal{H}$.

Using this identification of Z we can rewrite (3.7) as

$$\vec{\Psi}_E = \sum_{F \in \mathcal{F}_I \cup \mathcal{F}_D} Z_{E,F}\, \vec{v}_F, \qquad E \in (\mathcal{E}_I \cup \mathcal{E}_D) \backslash \mathcal{H}. \tag{5.1}$$

Note that the matrix Z is sparse; in fact, $Z_{E,F} \neq 0$ only when edge E is an edge of the face F.

The set $\mathcal{H} \subset \mathcal{E}$ of edges that form a spanning tree in the graph $\mathbf{G} = (\mathcal{N}, \mathcal{E})$ can be found in optimal time (proportional to the number of edges) using Algorithm A.7 presented in Appendix A. The introduction of the special spanning tree for the mixed boundary problem in Corollary 4.5 does not pose an extra problem to our method. We need only to modify Algorithm A.7 slightly: We choose $x_1 \in \mathcal{N}_N$ and at first consider only nodes $y_1 \in \mathcal{N}_N$ in the function "*recursive*(.)" to find a spanning tree $\mathbf{H}_N^\ell$ for each Γ_N^ℓ; then (without resetting the array "*mark*[.]") we call the function "*recursive*($\tilde{x}$)" with argument $\tilde{x} \in \mathcal{N}_I \cup \mathcal{N}_D$ to find the rest of the spanning tree.

With the same convention as above we can write the elements of the matrix M appearing in (2.4) as

$$M_{F,F'} = m(\vec{v}_F, \vec{v}_{F'}), \qquad F, F' \in \mathcal{F}_I \cup \mathcal{F}_D. \tag{5.2}$$

With these observations, it is simple to write $\mathring{A}$ as a sum of element matrices. To be precise, recalling that $\mathcal{T}$ is the set of tetrahedral elements, we can write

$$M = \sum_{T \in \mathcal{T}} M_T, \qquad \text{where } (M_T)_{F,F'} = \int_T K^{-1} \vec{v}_F \cdot \vec{v}_{F'}\, d\vec{x}.$$

Then we can similarly write $\mathring{A}$ as

$$\mathring{A} = \sum_{T \in \mathcal{T}} \mathring{A}_T, \tag{5.3}$$

where $\mathring{A}_T = Z_T M_T Z_T^T$, and Z_T denotes the matrix whose entries equal the entries of Z for columns and rows corresponding to T (i.e., faces $F \subset \overline{T}$ and edges $E \subset \overline{T}$) and are zero elsewhere. The representation (5.3) may be important if iterative methods are being used to solve (3.3). A similar elementwise representation can be given for the computation of $\mathring{\boldsymbol{g}}$ in (3.4).

Alternatively, $\mathring{A}$ can be determined (elementwise or globally) from an approximation of a related bilinear form by Nédélec's edge elements, without the assembly of any Raviart–Thomas stiffness matrix entries, as the following calculation shows.

Introduce the bilinear form

$$a(\vec{\Phi}, \vec{\Phi}') := (K^{-1}\, \vec{\text{curl}}\vec{\Phi}, \vec{\text{curl}}\vec{\Phi}')_{L^2(\Omega)^3} \qquad \text{for all } \vec{\Phi}, \vec{\Phi}' \in H(\vec{\text{curl}}, \Omega), \tag{5.4}$$

and, for $E, E' \in \mathcal{E}$, set

$$\mathcal{A}_{E,E'} := a(\vec{\Phi}_E, \vec{\Phi}_{E'}),$$

where $\{\vec{\Phi}_E\}$ are the basis functions of the piecewise linear Nédélec's edge elements defined in (4.2) and (4.3). Thus (after specifying an ordering of the edges in $\mathcal{E}$), $\mathcal{A}$ is the stiffness matrix corresponding to the bilinear form $a(\cdot, \cdot)$ discretized by Nédélec's edge elements, with a natural boundary condition on all of Γ. Because of the nontrivial kernel of $a(\cdot,\cdot)$, this bilinear form is degenerate and therefore not elliptic on $H(\vec{\text{curl}},\Omega)$. In fact, let $v \in H^1(\Omega)$; then $a(\vec{\nabla}v, \vec{\Phi}') = 0$ for all $\vec{\Phi}' \in H(\vec{\text{curl}},\Omega)$. Consequently, $\mathcal{A}$ is singular.

The following result shows that the minor of this matrix obtained by restricting to $E, E' \in (\mathcal{E}_I \cup \mathcal{E}_D)\backslash\mathcal{H}$, where $\mathcal{H} \subset \mathcal{E}$ as defined in Corollary 4.5, determines the matrix $\mathring{A}$ in (3.3). (This corresponds to imposing an essential boundary condition on Γ_N and restricting to the orthogonal complement of the kernel of $a(\cdot,\cdot)$.)

Theorem 5.1. *Let $\mathcal{H} \subset \mathcal{E}$ as defined in Corollary* 4.5 *(or Theorem* 4.2*, if $\Gamma_N = \emptyset$). Then*

$$\mathring{A}_{E,E'} = \mathcal{A}_{E,E'} \qquad \text{for all } E, E' \in (\mathcal{E}_I \cup \mathcal{E}_D)\backslash\mathcal{H}.$$

Proof. Let $\mathcal{H} \subset \mathcal{E}$ as defined in Corollary 4.5 (or Theorem 4.2, if $\Gamma_N = \emptyset$) and let $E, E' \in (\mathcal{E}_I \cup \mathcal{E}_D)\backslash\mathcal{H}$. Then using the definition (3.4) of $\mathring{A}$ together with (5.2) and (5.1) we get

$$\mathring{A}_{E,E'} = \sum_{F,F' \in \mathcal{F}_I \cup \mathcal{F}_D} Z_{E,F} M_{F,F'} Z_{E',F'} = m\left(\sum_F Z_{E,F}\vec{v}_F, \sum_{F'} Z_{E',F'}\vec{v}_{F'}\right) = m(\vec{\Psi}_E, \vec{\Psi}_{E'}).$$

Now using the definitions of $m(\cdot,\cdot)$ and $\vec{\Psi}_E$, we finally get

$$\begin{aligned}\mathring{A}_{E,E'} &= (K^{-1}\,\vec{\Psi}_E, \vec{\Psi}_{E'})_{L^2(\Omega)^3} = (K^{-1}\,\vec{\text{curl}}\vec{\Phi}_E, \vec{\text{curl}}\vec{\Phi}_{E'})_{L^2(\Omega)^3} \\ &= a(\vec{\Phi}_E, \vec{\Phi}_{E'}) = \mathcal{A}_{E,E'} \qquad \square\end{aligned}$$

Remark 5.2. *Hiptmair and Hoppe* [19] *solve the singular symmetric positive semidefinite system with stiffness matrix $\mathcal{A}$ by multilevel preconditioned CG without explicitly eliminating columns and rows corresponding to edges $E \in \mathcal{H}$. In their multilevel splitting they eliminate the kernel of $a(\cdot,\cdot)$ only approximately by relaxing the orthogonality condition and thus avoid the construction of a basis. Here we eliminate the kernel a priori, which allows us to then apply the CG algorithm with a range of possible preconditioners.*

Remark 5.3. *Observe that the decoupled system* (3.3) *is about three times smaller than the original indefinite system* (2.4)*. More precisely, the dimension of* (3.3) *is smaller than that of* (2.4) *by a factor*

$$C := \frac{\#\mathcal{F}_I + \#\mathcal{F}_D + \#\mathcal{T}}{\#\mathcal{F}_I + \#\mathcal{F}_D - \#\mathcal{T}}\,.$$

Since $4(\#\mathcal{T}) = 2(\#\mathcal{F}_I) + \#\mathcal{F}_D + \#\mathcal{F}_N$ *we have*

$$C = 3\left\{\frac{\#\mathcal{F}_I + \frac{5}{6}(\#\mathcal{F}_D) + \frac{1}{6}(\#\mathcal{F}_N)}{\#\mathcal{F}_I + \frac{3}{2}(\#\mathcal{F}_D) - \frac{1}{2}\#\mathcal{F}_N}\right\}.$$

Under reasonable mesh regularity assumptions $\#\mathcal{F}_I$ is the dominant part of $\#\mathcal{F}$ as $\#\mathcal{T} \to \infty$, and so $C \to 3$ as $\#\mathcal{T} \to \infty$.

6. Pressure computations. In this section we present a procedure for the efficient recovery of the pressure $\boldsymbol{p}$ from the decoupled system (3.6).

In the general situation described in section 3, the assembly of (3.6) requires the computation of a complementary basis $\{\boldsymbol{z}_{\mathring{n}+1}, \ldots, \boldsymbol{z}_{n_\mathcal{V}}\}$ satisfying (3.5). This is again equivalent to finding a complementary basis $\{\vec{v}^c_{\mathring{n}+1}, \ldots, \vec{v}^c_{n_\mathcal{V}}\}$ to $\{\vec{\mathring{v}}_1, \ldots, \vec{\mathring{v}}_{\mathring{n}}\}$ such that

$$\text{span}\left\{\vec{\mathring{v}}_1, \ldots, \vec{\mathring{v}}_{\mathring{n}}, \vec{v}^c_{\mathring{n}+1}, \ldots, \vec{v}^c_{n_\mathcal{V}}\right\} = \mathcal{V}\,. \tag{6.1}$$

In the context of the specific system (2.4) (using lowest-order Raviart–Thomas–Nédélec elements) this can be done by finding a distinguished subset of faces

$$\mathcal{F}^c \subset \mathcal{F}_I \cup \mathcal{F}_D$$

such that the corresponding subset of Raviart–Thomas–Nédélec basis functions, $\{\vec{v}_F : F \in \mathcal{F}^c\}$, constitutes a complementary basis. Note that this set must contain $n_{\mathcal{W}} := n_{\mathcal{V}} - \mathring{n} = \#\mathcal{T}$ elements. The following simple Algorithm 6.1 chooses $n_{\mathcal{W}}$ appropriate faces, yielding a complementary basis, in such a way that the system (3.6) has a particularly simple form. We have already presented this algorithm to find a subset of faces $\mathcal{F}^c \subset \mathcal{F}_I \cup \mathcal{F}_D$ for the two-dimensional case in [12], but here we will give a rigorous proof that the corresponding subset of Raviart–Thomas–Nédélec basis functions, $\{\vec{v}_F : F \in \mathcal{F}^c\}$, constitutes a complementary basis to (4.15) (or to (4.8), if $\Gamma_N = \emptyset$) in $\mathcal{V}$.

Algorithm 6.1.

```
1. Choose $T_1 \in \mathcal{T}$ to be any tetrahedron with a face $F_1 \in \mathcal{F}_D$ and set $\mathcal{F}^c = \{F_1\}$.
2. For $j = 2, \ldots, n_{\mathcal{W}}$,
    • choose $T_j \in \mathcal{T} \backslash \{T_\ell : \ell = 1, \ldots, j-1\}$ with the property that there exists
      $F_j \in \mathcal{F}_I$ such that
      (6.2)    $F_j \subset \overline{T}_j \cap \left\{ \bigcup_{\ell=1}^{j-1} \overline{T}_\ell \right\}$
    • update $\mathcal{F}^c = \mathcal{F}^c \cup \{F_j\}$.
   End of loop over $j$.
3. Assemble $Z'B$ as
      $(Z'B)_{i,j} = b(\vec{v}_{F_i}, w_{T_j}), \qquad i, j = 1, \ldots, n_{\mathcal{W}}$ .
```

Theorem 6.2. *Algorithm* 6.1 *is well defined and the matrix $Z'B$ given in step* 3 *is lower triangular.*

Proof. Since $\Gamma_D \neq \emptyset$, there exists a $T \in \mathcal{T}$ with a face $F \in \mathcal{F}_D$. Let T be T_1. Now, assume we have found $j - 1 < n_{\mathcal{W}} = \#\mathcal{T}$ tetrahedra T_ℓ in step 2 that fulfill property (6.2). Since Ω is connected, there exists a tetrahedron $T \in \mathcal{T}$ that has a face in common with $\bigcup_{\ell=1}^{j-1} \overline{T}_\ell$. Let T be T_j. The existence of a set $\mathcal{F}^c$ therefore follows by an inductive argument.

Second, let $i, j = 1, \ldots, n_{\mathcal{W}}$ with $i < j$. By the algorithm $F_i \subset \overline{T}_i \cap \{\bigcup_{\ell=1}^{i-1} \overline{T}_\ell\}$ and therefore $F_i \not\subset \overline{T}_j$. Since T_j is not in supp $\vec{v}_{F_i}$, it follows that $(Z'B)_{i,j} = b(\vec{v}_{F_i}, w_{T_j}) = 0$, and the matrix $Z'B$ given in step 3 is lower triangular. □

To show that Algorithm 6.1 yields a complementary basis, let us first consider the pure Dirichlet case, $\Gamma_N = \emptyset$.

Theorem 6.3. *Let $\Gamma_N = \emptyset$. The functions*

$$\{\vec{v}_F : F \in \mathcal{F}^c\} \tag{6.3}$$

form a complementary basis to (4.8) *in $\mathcal{V}$.*

Before proving this theorem we will first prove two lemmas again.

Lemma 6.4. *Let $\boldsymbol{\mu} := (\mu_E)_{E \in \mathcal{E}} \in \mathcal{V}(\mathbf{G})$. Then there exist $\{\tilde{\alpha}_F \in \mathbb{Z} : F \in \mathcal{F} \backslash \mathcal{F}^c\}$ such that*

$$\boldsymbol{\mu} := \sum_{F \in \mathcal{F} \backslash \mathcal{F}^c} \tilde{\alpha}_F \boldsymbol{\mu}^F . \tag{6.4}$$

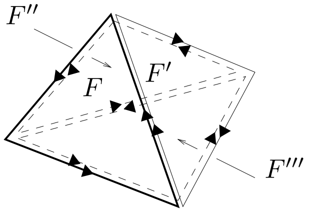
Proof. Let $F \in \mathcal{F}^c$ and let $\boldsymbol{\mu}^F$ be the vector associated with the elementary cycle μ^F formed by the edges E of F in the graph $\mathbf{G} := (\mathcal{N}, \mathcal{E})$. We will first show that there exist $\{\alpha^F_{F'} \in \mathbb{Z} : F' \in \mathcal{F}\backslash\mathcal{F}^c\}$ such that

$$\boldsymbol{\mu}^F = \sum_{F' \in \mathcal{F}\backslash\mathcal{F}^c} \alpha^F_{F'} \boldsymbol{\mu}^{F'}. \tag{6.5}$$

Let $j \in \{1, \dots, n_{\mathcal{W}}\}$ be such that $F = F_j$ in Algorithm 6.1 and let F', F'', and F''' be the other faces of T_j; then there exist $\alpha^F_{F'}, \alpha^F_{F''}, \alpha^F_{F'''} \in \{-1, 1\}$ such that

$$\boldsymbol{\mu}^F = \alpha^F_{F'} \boldsymbol{\mu}^{F'} + \alpha^F_{F''} \boldsymbol{\mu}^{F''} + \alpha^F_{F'''} \boldsymbol{\mu}^{F'''}$$

as depicted below:



If $F', F'', F''' \in \mathcal{F}\backslash\mathcal{F}^c$, the proof of (6.5) is complete. Otherwise assume, without loss of generality, that $F' \in \mathcal{F}^c$. By construction there has to be a $j' \in \{j+1, \dots, n_{\mathcal{W}}\}$ such that $F' = F_{j'}$. Let $\tilde{F}'$, $\tilde{F}''$, and $\tilde{F}'''$ be the other faces of $T_{j'}$. As before, there exist $\alpha^{F'}_{\tilde{F}'}, \alpha^{F'}_{\tilde{F}''}, \alpha^{F'}_{\tilde{F}'''} \in \{-1, 1\}$ such that $\boldsymbol{\mu}^{F'} = \alpha^{F'}_{\tilde{F}'} \boldsymbol{\mu}^{\tilde{F}'} + \alpha^{F'}_{\tilde{F}''} \boldsymbol{\mu}^{\tilde{F}''} + \alpha^{F'}_{\tilde{F}'''} \boldsymbol{\mu}^{\tilde{F}'''}$ and therefore

$$\boldsymbol{\mu}^F = \alpha^F_{F'} (\alpha^{F'}_{\tilde{F}'} \boldsymbol{\mu}^{\tilde{F}'} + \alpha^{F'}_{\tilde{F}''} \boldsymbol{\mu}^{\tilde{F}''} + \alpha^{F'}_{\tilde{F}'''} \boldsymbol{\mu}^{\tilde{F}'''}) + \alpha^F_{F''} \boldsymbol{\mu}^{F''} + \alpha^F_{F'''} \boldsymbol{\mu}^{F'''}.$$

If $F'', F''', \tilde{F}', \tilde{F}'', \tilde{F}''' \in \mathcal{F}\backslash\mathcal{F}^c$, the proof of (6.5) is complete. Otherwise, we can repeat the above procedure for the faces $F'', F''', \tilde{F}', \tilde{F}''$, and $\tilde{F}'''$, and since the set $\{1, \dots, n_{\mathcal{W}}\}$ is finite, the procedure will terminate in a finite number of steps. Altogether, we have established that there exist $\{\alpha^F_{F'} \in \mathbb{Z} : F' \in \mathcal{F}\backslash\mathcal{F}^c\}$ such that (6.5) holds.

Now let $\boldsymbol{\mu} \in \mathcal{V}(\mathbf{G})$. Substituting (6.5) into (4.10), we find that there exist $\{\tilde{\alpha}_F \in \mathbb{Z} : F \in \mathcal{F}\backslash\mathcal{F}^c\}$ such that $\boldsymbol{\mu} = \sum_{F \in \mathcal{F}\backslash\mathcal{F}^c} \tilde{\alpha}_F \boldsymbol{\mu}^F$. □

Lemma 6.5. *Let $\boldsymbol{\mu} \in \mathcal{V}(\mathbf{G})$ and $\{\tilde{\alpha}_F \in \mathbb{Z} : F \in \mathcal{F}\backslash\mathcal{F}^c\}$ be such that $\boldsymbol{\mu} := \sum_{F \in \mathcal{F}\backslash\mathcal{F}^c} \tilde{\alpha}_F \boldsymbol{\mu}^F$. Then*

$$\sum_{F \in \mathcal{F}\backslash\mathcal{F}^c} \tilde{\alpha}_F \int_F \vec{v}_{F'} \cdot \vec{\nu}_F \, dF = 0 \qquad \textit{for all } F' \in \mathcal{F}^c. \tag{6.6}$$

Proof. Let $F' \in \mathcal{F}^c$; then $\vec{v}_{F'} \cdot \vec{\nu}_F = 0$ for all $F \in \mathcal{F}\backslash\mathcal{F}^c$, which implies (6.6). □

We can now prove Theorem 6.3.

Proof. Since $\#\mathcal{F}^c = n_{\mathcal{W}}$, we merely need to show that the union of the sets of functions (6.3) and (4.8) is a linearly independent set. Therefore suppose $\{\beta_{E'} : E' \in \mathcal{E}\backslash\mathcal{H}\}$ and $\{\gamma_F : F \in \mathcal{F}^c\}$ are scalars such that

$$\vec{0} = \sum_{E' \in \mathcal{E}\backslash\mathcal{H}} \beta_{E'} \vec{\Psi}_{E'} + \sum_{F \in \mathcal{F}^c} \gamma_F \vec{v}_F.$$

Let $E \in \mathcal{E}\backslash\mathcal{H}$ and let $\boldsymbol{\mu}^E$ denote the vector associated with the unique cycle μ^E generated by taking edge E into the tree $\mathbf{H} = (\mathcal{N}, \mathcal{H})$, which has the property that $\mu^E_{E'} := \delta_{E,E'}$ for all $E' \in \mathcal{E}\backslash\mathcal{H}$ (cf. Theorem A.6 and the proof to Theorem 4.2). Now, using Lemma 6.4 we can find $\{\tilde{\alpha}_F \in \mathbb{Z} : F \in \mathcal{F}\backslash\mathcal{F}^c\}$ such that $\boldsymbol{\mu} := \sum_{F\in\mathcal{F}\backslash\mathcal{F}^c} \tilde{\alpha}_F \boldsymbol{\mu}^F$, and so by Lemmas 4.4 and 6.5

$$\begin{aligned} 0 &= \sum_{F\in\mathcal{F}\backslash\mathcal{F}^c} \tilde{\alpha}_F \int_F \left(\sum_{E\in\mathcal{E}\backslash\mathcal{H}} \beta_E \,\vec{\Psi}_E + \sum_{F'\in\mathcal{F}^c} \gamma_{F'}\,\vec{v}_{F'} \right) \cdot \vec{\nu}_F \, dF \\ &= \sum_{E'\in\mathcal{E}\backslash\mathcal{H}} \beta_{E'} \sum_{F\in\mathcal{F}\backslash\mathcal{F}^c} \tilde{\alpha}_F \int_F \vec{\Psi}_E \cdot \vec{\nu}_F \, dF \; + \sum_{F'\in\mathcal{F}^c} \gamma_{F'} \sum_{F\in\mathcal{F}\backslash\mathcal{F}^c} \tilde{\alpha}_F \int_F \vec{v}_{F'} \cdot \vec{\nu}_F \, dF \\ &= \sum_{E'\in\mathcal{E}\backslash\mathcal{H}} \beta_{E'} \mu^E_{E'} \;=\; \beta_E \,. \end{aligned}$$

Since the functions $\{\vec{v}_F : F \in \mathcal{F}^c\}$ form a subset of the Raviart–Thomas–Nédélec basis functions, they have to be linearly independent. Therefore we also have $\gamma_F = 0$, for all $F \in \mathcal{F}^c$, which establishes the linear independence of the functions in (6.3) and (4.8). □

COROLLARY 6.6. *Let* $n_{\mathcal{C}} \neq 0$ *and let* Γ^ℓ_N *be simply connected for each* $\ell = 1, \dots, n_{\mathcal{C}}$. *The functions*

$$\{\vec{v}_F : F \in \mathcal{F}^c\} \tag{6.7}$$

form a complementary basis to (4.15) *in* $\mathcal{V}$.

Proof. Since $\#\mathcal{F}^c = n_{\mathcal{W}}$, the result follows directly from Corollary 4.5 and Theorem 6.3. □

Using this complementary basis (6.7) and applying the general theory presented in section 3, we can therefore find the unique solution $\boldsymbol{p}$ from (3.6) by simple *back substitutions.*

The matrix $Z'B$ is obtained from the original matrix B in (2.4) by deleting some rows and reordering the rows and columns. Equivalently, the right-hand side $Z'(\boldsymbol{g} - M\boldsymbol{u})$ of (3.6) is obtained from $\boldsymbol{g} - M\boldsymbol{u}$ by deleting some rows and reordering the rows.

7. Numerical results.

In this section we want to demonstrate the performance of the proposed method in two very simple test cases. Let Ω be the unit cube $(0,1)^3$. We will consider only the constant coefficient case $K \equiv 1$, of system (1.1), (1.2), with zero right-hand side $\vec{g} = \vec{0}$. The two experiments are induced by different choices of boundary conditions. In Figure 2 we illustrate the Neumann boundary Γ_N in each case.

In the *first experiment* (Figure 2 (left)) we choose

$$\begin{aligned} p_D(x,y,z) &= 1 - x \qquad &&\text{on} \quad \Gamma_D = \{0,1\}\times(0,1)\times(0,1) \;\cup\; [0,1]\times(0,1)\times\{1\}, \\ \vec{u}\cdot\vec{\nu} &= 0 &&\text{on} \quad \Gamma_N = \Gamma\backslash\Gamma_D, \end{aligned}$$

where $\vec{\nu}(\vec{x})$ denotes the outward unit normal from Ω at $\vec{x} \in \Gamma$ as before.

In the *second experiment* (Figure 2 (right)) we choose

$$\begin{aligned} p_D(x,y,z) &= 1 - x \qquad &&\text{on} \quad \Gamma_D = (0,1)\times(0,1)\times\{1\}, \\ \vec{u}\cdot\vec{\nu} &= 0 &&\text{on} \quad \Gamma_N = \Gamma\backslash\Gamma_D. \end{aligned}$$

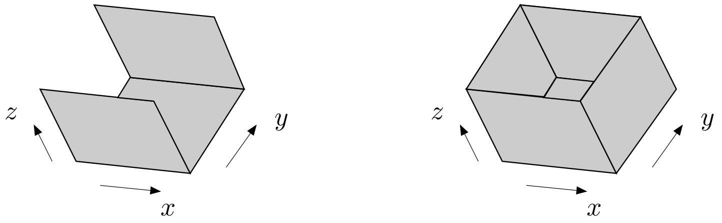


Fig. 2. *The "no-flux" boundary* Γ_N *for Experiments* 1 *and* 2*, respectively.*

We discretize these problems using the mixed finite element discretization (2.3) with lowest-order Raviart–Thomas–Nédélec elements on a sequence of uniform and nonuniform meshes of different refinement levels L. The uniform mesh is constructed from a uniform rectangular mesh of $L * L * L$ cubes which are each subdivided themselves into six tetrahedra. The nonuniform mesh is constructed from a nonuniform hexahedral mesh of $L * L * L$ hexahedra which are each subdivided themselves into 24 tetrahedra.

To solve the resulting saddle-point system (2.4) we use the decoupled iterative method described above: The construction of the matrix $\mathring{A}$ in the decoupled velocity system (3.3) is carried out in an elementwise fashion as presented in (5.3); the resulting symmetric positive definite system (3.3) is solved with ILU(0)-preconditioned conjugate gradients (PCG); the matrix $Z'B$ in the decoupled pressure system (3.6) is obtained from the original matrix B in (2.4) by deleting some rows and reordering the rows and columns (as mentioned at the end of section 6); the resulting triangular system (3.6) is solved by simple back substitutions. The convergence criterion in the PCG method is the relative reduction of the residual by a factor of 10^{-5}.

Tables 1 and 2 show the performance of our method for Experiments 1 and 2, respectively. First of all we observe in both cases the reduction in size from the full mixed system (2.4) to the decoupled velocity system (3.3). (Compare rows 3 and 4 in Tables 1 and 2.) It is approximately 3 as claimed in Remark 5.3.

Second, let us look at the number of *floating point operations* (Flops) needed in the decoupling process (row 7). This process is asymptotically optimal (# Flops $= O($# Freedoms$)$), since the matrix $\mathring{A}$ and the right-hand side in (3.3) are constructed in an elementwise fashion and since the matrix $Z'B$ and the right-hand side in (3.6) are obtained from the original system (2.4) by simple reordering. While this decoupling process is still the dominant part in the total Flop count (row 9) on very small problems (e.g., column 2), its effect gets less important and will eventually vanish for larger problems (e.g., columns 5 and 8).

The core part of the calculation is the solution of the decoupled velocity system (3.3). In the uniform mesh case (columns 2–5) the condition number of $\mathring{A}$ (row 5) grows as expected with $O(\mathring{n}^{2/3})$, where $\mathring{n}$ is the dimension of $\mathring{A}$ (row 4). The growth in the nonuniform mesh case (columns 6–8) is (naturally) slightly worse. This growth of the condition number is reflected in the iteration and Flop count for the PCG method to solve (3.3). The number of iterations (row 6) grows with the square root of the condition number of $\mathring{A}$; in the uniform mesh case (columns 2–5) this is a factor of about 2 from one refinement level to the next.

Remark 7.1. *Although the effect of the ILU(0) preconditioner deteriorates as the grid size decreases, it is extremely cheap to invert and remains a cost effective way of preconditioning this system. Diagonal scaling, for example, leads to a similar*

Table 1
Performance of the decoupled iterative method for Experiment 1.

	Uniform mesh				Nonuniform mesh		
Refinement level L	2	4	8	16	2	4	8
Freedoms (full mixed)	144	1152	9216	73728	576	4608	36864
Freedoms (decoupled)	48	384	3072	24576	192	1536	12288
Condition # (decoupled)	150	610	2370	9230	1240	10670	89700
PCG-iterations	14	26	45	97	33	90	232
MFlops (decoupling)	0.032	0.34	3.0	25	0.17	1.5	13
MFlops (PCG)	0.031	0.53	8.0	142	0.35	8.0	171
MFlops (total)	0.065	0.89	11.2	169	0.53	9.6	185

Table 2
Performance of the decoupled iterative method for Experiment 2.

	Uniform mesh				Nonuniform mesh		
Refinement level L	2	4	8	16	2	4	8
Freedoms (full mixed)	128	1088	8960	72704	544	4480	36352
Freedoms (decoupled)	32	320	2816	23552	160	1408	11776
Condition # (decoupled)	87	450	2080	8740	360	2780	21300
PCG-iterations	9	18	35	75	24	80	187
MFlops (decoupling)	0.024	0.29	2.8	25	0.15	1.4	12
MFlops (PCG)	0.012	0.31	5.8	106	0.20	6.5	132
MFlops (total)	0.037	0.61	8.7	132	0.36	8.0	145

asymptotic behavior but is more expensive in terms of iterations as well as Flops. For Experiment 1 *the number of iterations using diagonal scaling as the preconditioner in the PCG method in the uniform mesh case is* 38*,* 109*,* 245*, and* 494 *for* $L = 2, 4, 8,$ *and* 16*, respectively. The number of MFlops is* 0.086*,* 1.73*,* 29.7*, and* 47*. Future work will involve a more detailed investigation of various preconditioners like algebraic multigrid (AMG).*

Finally, let us look at the solution of the decoupled pressure system (3.6). Since the matrix $Z'B$ in (3.6) is triangular (as shown in Theorem 6.2) it can be solved in an asymptotically optimal number of operations by simple back substitutions. Therefore this part of the calculation does not affect the overall cost of the method significantly. The number of Flops is included in the total Flop count (row 9) reported in Tables 1 and 2, and it accounts for less than 1% of the total cost of the method for larger problems (e.g., columns 5 and 8).

In Tables 3 and 4 we compare the performance of our method with the performance of a preconditioned MINRES method for the original (full mixed) saddle-point system (2.4) for Experiments 1 and 2, respectively. The convergence criterion for MINRES is again the relative reduction of the residual by a factor of 10^{-5}. To precondition this MINRES method we take an optimal symmetric positive definite block diagonal preconditioner presented and analyzed in Rusten and Winther [26] (using an ILU(0) factorization of B^TB for the pressure block).

Remark 7.2. *As for the decoupled system it would be possible to employ other, more efficient preconditioners like AMG for the pressure block. However, in order to evaluate the performance of the decoupling procedure, our prime incentive, we wanted to compare "similar" iterative methods for the full mixed and the decoupled system.*

Comparing columns 6 and 7 in Table 3 we observe that for the first experiment our decoupled method is about 4.5 times faster than preconditioned MINRES on the uniform meshes (rows 3–6), and about 6 times faster on the nonuniform meshes

Table 3
Comparison of the decoupled iterative method with a full mixed method [26] *for Experiment* 1.

		Freedoms		Iterations		MFlops	
Mesh	L	Mixed	Decoupled	Mixed	Decoupled	Mixed	Decoupled
Uniform	2	144	48	37	14	0.29	0.065
	4	1152	384	56	26	3.8	0.89
	8	9216	3072	89	45	49	11
	16	73728	24576	175	97	790	169
Nonuniform	2	576	192	100	33	3.4	0.53
	4	4608	1536	204	90	57	9.6
	8	36864	12288	456	232	1030	185

Table 4
Comparison of the decoupled iterative method with a full mixed method [26] *for Experiment* 2.

		# Freedoms		Iterations		MFlops	
Mesh	L	Mixed	Decoupled	Mixed	Decoupled	Mixed	Decoupled
Uniform	2	128	32	37	9	0.25	0.038
	4	1088	320	57	18	3.6	0.62
	8	8960	2816	109	35	59	8.7
	16	72704	23552	217	75	960	132
Nonuniform	2	544	160	110	24	3.5	0.35
	4	4480	1408	254	80	68	8.0
	8	36352	11776	582	187	1300	145

(rows 7–9). The advantage of our method over preconditioned MINRES is even more impressive for the second experiment (columns 6 and 7 in Table 4). On the uniform meshes (rows 3–6) it is about 7 times faster, on the nonuniform meshes (rows 7–9) about 9 times faster.

In conclusion we have found a very competitive and practicable iterative method to solve saddle-point problems of the form (2.4). The decoupling procedure and the recovery of the pressure are asymptotically optimal. The decoupled velocity system, on the other hand, is symmetric positive definite and of second order, and there may still be room for improvement in solving this system by employing the more sophisticated preconditioning techniques which are available for such problems.

Appendix A. Some results from graph theory.

Definition A.1 (Berge [5]).

(a) *A* graph *(or more precisely a* 1-graph*)* $\mathbf{G}$ *is defined to be a pair* $(\mathcal{X},\mathcal{U})$*, where* $\mathcal{X}$ *is a set* $\{x_1, x_2, \dots, x_n\}$ *of elements called* vertices *(or nodes), and* $\mathcal{U}$ *is a subset* $\{u_1, u_2, \dots, u_m\}$ *of* $\mathcal{X} \times \mathcal{X}$ *of elements called* arcs *(or* orientated edges*). For an arc* $u = (x, y) \in \mathcal{X} \times \mathcal{X}$*, the vertex* x *is called its* initial endpoint*, and the vertex* y *is called its* terminal endpoint*. A vertex* $y \in \mathcal{X}$ *is called a* neighbor *of* $x \in \mathcal{X}$ *if either* $(x, y) \in \mathcal{U}$ *or* $(y, x) \in \mathcal{U}$*. The set of all neighbors of a vertex* x *in the graph* $\mathbf{G}$ *will be denoted by* $\Gamma_{\mathbf{G}}(x)$*.*

(b) *A* partial graph *of a graph* $\mathbf{G} = (\mathcal{X},\mathcal{U})$ *is a graph* $\mathbf{H} = (\mathcal{X}, \mathcal{V})$ *with* $\mathcal{V} \subset \mathcal{U}$*.*

(c) *A* chain *is a sequence* $\mu = (u_{i_1}, u_{i_2}, \dots, u_{i_q})$ *of arcs of a graph* $\mathbf{G}$ *such that each arc in the sequence has one endpoint in common with its predecessor and its other endpoint in common with its successor. A chain that does not encounter the same vertex twice is called* elementary*. A chain that does not use the same arc twice is called* simple*.*

(d) *For two vertices x and y of a graph* $\mathbf{G}$ *let us define the equivalence relation* $x \equiv y$ *by*

$$[x = y, \text{ or } x \neq y \text{ and there exists a chain in } \mathbf{G} \text{ connecting } x \text{ and } y].$$

The equivalence classes of $\equiv$ *are called the* connected components *of* $\mathbf{G}$. *A* connected graph *is a graph that consists only of one connected component.*

(e) *A* cycle *is a simple chain whose terminal endpoint coincides with its initial endpoint. Let m be the number of arcs in* $\mathbf{G}$. *With each cycle μ of* $\mathbf{G}$ *we can associate a vector* $\boldsymbol{\mu} \in \mathbb{Z}^m$ *with*

$$\mu_i = \begin{cases} 0 & \text{if } u_i \text{ is not in } \mu, \\ 1 & \text{if } u_i \text{ is in } \mu \text{ and shares initial endpoint with its predecessor}, \\ -1 & \text{if } u_i \text{ is in } \mu \text{ and shares terminal endpoint with its predecessor}. \end{cases}$$

The set of all those vectors $\boldsymbol{\mu} \in \mathbb{Z}^m$ *generates a vector space over* $\mathbb{Z}$. *We denote this vector space by* $\mathcal{V}(\mathbf{G})$.

(f) *A* forest *is defined to be a graph without cycles. A* tree *is defined to be a connected graph without cycles.*

Theorem A.2. *Let* $\mathbf{G}$ *be a graph with n vertices, m arcs, and p connected components. The dimension of* $\mathcal{V}(\mathbf{G})$ *is* $m - n + p$.

Proof. The proof is seen in Berge [5, p. 16]. □

Theorem A.3. *Let* $\mathbf{H} = (\mathcal{X}, \mathcal{U})$ *be a graph with* $n > 2$ *vertices. The following properties are equivalent and each characterizes a tree:*

(i) $\mathbf{H}$ *is connected and has no cycles.*

(ii) $\mathbf{H}$ *has* $n - 1$ *arcs and has no cycles.*

(iii) $\mathbf{H}$ *is connected and contains* $n - 1$ *arcs.*

(iv) $\mathbf{H}$ *has no cycles and adding an arc creates a unique cycle.*

(v) $\mathbf{H}$ *is connected and removing an arc leaves the remaining graph disconnected.*

(vi) *Every pair of vertices* x, y *of* $\mathbf{H}$ *is connected by a unique chain.*

Proof. The proof is seen in Berge [5, p. 24]. □

Theorem A.4. *Let* $\mathbf{G} = (\mathcal{X}, \mathcal{U})$ *be a connected graph. There exists a partial graph* $\mathbf{H} = (\mathcal{X}, \mathcal{V})$ *such that* $\mathbf{H}$ *is a tree.*

Proof. The proof is seen in Berge [5, p. 25]. □

The tree $\mathbf{H}$ obtained from $\mathbf{G}$ as above is called a *spanning tree.* An optimal algorithm to find a spanning tree $\mathbf{H}$ of a connected graph $\mathbf{G}$ is presented in Algorithm A.7.

Theorem A.5. *Let* $\mathbf{G}$ *be a graph with n vertices and* $m \geq n$ *arcs. The time spent on Algorithm* A.7 *is proportional to the number of arcs, i.e.,* $O(m)$.

Proof. The proof is seen in Aho, Hopcroft, and Ullman [1]. □

Theorem A.6. *Let* $\mathbf{G} = (\mathcal{X}, \mathcal{U})$ *be a connected graph with n vertices and m arcs, let* $\mathbf{H} = (\mathcal{X}, \mathcal{V})$ *be a spanning tree of* $\mathbf{G}$, *and let* $u_i \in \mathcal{U}$ *be an arc of* $\mathbf{G}$ *not in tree* $\mathbf{H}$, *i.e.,* $u_i \notin \mathcal{V}$. *Adding* u_i *to* $\mathbf{H}$ *creates a unique cycle* μ^i, *and its associated vector* $\boldsymbol{\mu}^i$ *satisfies* $\mu^i_i = 1$. *The set* $\{\boldsymbol{\mu}^i : u_i \in \mathcal{U} \backslash \mathcal{V}\}$ *forms a basis of* $\mathcal{V}(\mathbf{G})$.

Proof. The existence of $\boldsymbol{\mu}^i$ for all $u_i \in \mathcal{U} \backslash \mathcal{V}$ is guaranteed by virtue of Theorem A.3(iv). The vectors are linearly independent, since $\mu^j_i = \delta_{i,j}$, for all $u_i, u_j \in \mathcal{U} \backslash \mathcal{V}$. Moreover,

$$\dim\{\boldsymbol{\mu}^i : u_i \in \mathcal{U} \backslash \mathcal{V}\} = \#\mathcal{U} - \#\mathcal{V} = m - (n - 1) = \dim \mathcal{V}(\mathbf{G}),$$

where in the last step we used Theorem A.2 with $p = 1$. □

Algorithm A.7.

```
variables
    n – number of vertices;
    mark[1:n] – array of flags;
begin
    $\mathcal{V} = \emptyset$;
    for i := 1 to n do mark[$x_i$] := unvisited;
    recursive($x_1$);
end;

procedure recursive( x – vertex );
    variables
        y – vertex;
    begin
        mark[x] := visited;
        for each vertex $y \in \Gamma_{\mathbf{G}}(x)$ do
            if mark[y] = unvisited then
                $\mathcal{V} = \mathcal{V} \cup \{u\}$  – where $u \in \mathcal{U}$ with endpoints x and y;
                recursive(y);
    end;
```

Appendix B. A topological result on simplicial triangulations.

Definition B.1 (the fundamental group (Armstrong [3, Ch. 5])).

(a) *A* topological space *is a set S together with a collection $\mathcal{U}$ of subsets of S satisfying the following conditions:*

(1) $\emptyset \in \mathcal{U}$, $S \in \mathcal{U}$.

(2) *If* $U_1, \dots, U_n \in \mathcal{U}$, *then* $\bigcap_{i=1}^{n} U_i \in \mathcal{U}$.

(3) *If* $\tilde{\mathcal{U}} \subset \mathcal{U}$, *then* $\bigcup_{U \in \tilde{\mathcal{U}}} U \in \mathcal{U}$.

The elements of $\mathcal{U}$ are called open sets *in S. $\mathcal{U}$ is called a* topology *on S.*

(b) *Let X be a topological space. A* path *in X from x_0 to x_1 (with origin x_0 and end x_1) is a continuous map $\alpha : [0,1] \to X$ such that $\alpha(0) = x_0$ and $\alpha(1) = x_1$. Let α be a path in X from x_0 to x_1 and let β be a path in X from x_1 to x_2. The* product *of α and β is the path $\alpha\beta$ from x_0 to x_2 defined by*

$$\alpha\beta(t) = \begin{cases} \alpha(2t) & \text{for } t \in [0, 1/2], \\ \beta(2t-1) & \text{for } t \in [1/2, 1]. \end{cases}$$

The inverse *of α is the path α^{-1} from x_1 to x_0 defined by $\alpha^{-1}(t) = \alpha(1-t)$.*

(c) *Two paths α and β from x_0 to x_1 are* homotopic *(written $\alpha \simeq \beta$) if there exists a continuous map $F : [0,1] \times [0,1] \to X$ such that*

$$\begin{array}{llll} F(0,t) = x_0 & \text{and} & F(1,t) = x_1 & \text{for all } t \in [0,1], \\ F(s,0) = \alpha(s) & \text{and} & F(s,1) = \beta(s) & \text{for all } s \in [0,1]. \end{array}$$

(d) *Let X be a topological space and let $x_0 \in X$. The set of $\simeq$ equivalence classes of paths with origin x_0 and end x_0 forms a group under the operations of multiplication and inverse as defined above. This group is denoted $\pi_1(X, x_0)$ and is called the* fundamental group *of the pair (X, x_0). X is called* simply connected *if its fundamental group is trivial.*

DEFINITION B.2 (the first homology group (Armstrong [3, Ch. 8])).

(a) *Let V be a vector space over $\mathbb{R}$ and let $\{v_0, v_1, \ldots, v_k\} \subset V$ such that the set $\{v_1 - v_0, \ldots, v_k - v_0\}$ is linearly independent. The smallest convex set containing $\{v_0, v_1, \ldots, v_k\}$, i.e., the convex hull*

$$\left\{ v := \sum_{i=0}^{k} \lambda_i v_i : \lambda_i \geq 0 \text{ and } \sum_{i=0}^{k} \lambda_i = 1 \right\},$$

is called a simplex of dimension k *(or a k-*simplex*). The points $v_0, v_1, \ldots, v_k$ are called the* vertices *(or nodes) of the simplex. The simplices formed by the subsets of $\{v_0, v_1, \ldots, v_k\}$ are called the* faces *of the simplex.*

(b) *A* simplicial complex *K is a finite set of simplices in V such that*
  (1) *if $A \in K$, then the faces of A are also in K;*
  (2) *if $A, B \in K$ and $A \cap B \neq \emptyset$, then $A \cap B \in K$.*

The dimension *of K is the maximum dimension of the simplices of K. The point set union of all simplices in K is denoted by $|K|$.*

(c) *Let K be a simplicial complex. An* orientated edge *in K is an ordered pair (u, v) such that u and v lie in some simplex of K. An* orientated triangle *in K is an ordered triple (u, v, w) such that u, v, w lie in some simplex of K. Note that $(u, v, w) = (v, w, u) = (w, u, v)$. A change of orientation is denoted by a minus sign, thus $(v, u) = -(u, v)$ and $(v, u, w) = -(u, v, w)$. The* boundary *of the orientated edge (u, v) is defined to be*

$$\partial(u, v) = v - u.$$

The boundary of the orientated triangle (u, v, w) is

$$\partial(u, v, w) = (v, w) + (w, u) + (u, v).$$

Let n be the number of all edges in K. A linear combination of orientated edges

$$\sum_{i=1}^{n} \lambda_i (u_i, v_i) \quad \textit{with the property that} \quad \sum_{i=1}^{n} \lambda_i \partial(u_i, v_i) = 0$$

and $\lambda_i \in \mathbb{Z}$ for all $i = 1, \ldots, n$ is called a (one-dimensional) cycle *of K. A cycle β is called a* bounding cycle *if we can find a linear combination*

$$\sum_{j=1}^{k} \alpha_j (u_j, v_j, w_j)$$

of orientated triangles in K such that

$$\beta = \sum_{j=1}^{k} \alpha_j \partial(u_j, v_j, w_j).$$

(d) *The set of all cycles of K forms an abelian group under the addition*

$$\sum_{i=1}^{n} \lambda_i (u_i, v_i) + \sum_{i=1}^{n} \mu_i (u_i, v_i) = \sum_{i=1}^{n} (\lambda_i + \mu_i)(u_i, v_i).$$

We denote this group by $Z_1(K)$. The bounding cycles form a subgroup $B_1(K)$ of $Z_1(K)$. The quotient group

$$H_1(K) = Z_1(K)\backslash B_1(K)$$

is called the first homology group *of K.*

We will need only the following fundamental theorem which is a corollary to the simplicial approximation theorem (Armstrong [3, p. 128]).

THEOREM B.3. *Let K be a simplicial complex and let v be a vertex of K. If $|K|$ is connected, abelianizing $\pi_1(|K|, v)$ gives the first homology group $H_1(K)$.*

Proof. The proof is seen in Armstrong [3, p. 182]. □

COROLLARY B.4. *If $|K|$ is simply connected, then each cycle of K is a bounding cycle.*

Proof. From Definition B.1(d) we know that if $|K|$ is simply connected, then $\pi_1(|K|, v)$ is trivial for any vertex v of K. As a consequence of Theorem B.3 this also implies that $H_1(K)$ is trivial (since abelianizing the trivial group has to result in the trivial group again). Therefore $B_1(K) = Z_1(K)$. □

Acknowledgment. I would like to thank Professor Thomas Russell for kindly sending me a copy of Hecht's unpublished manuscript [18] and particularly thank Professor Ivan Graham and Professor Andrew Swann for useful discussions and comments.

REFERENCES

[1] A.V. AHO, J.E. HOPCROFT, AND J.D. ULLMAN, *Data Structures and Algorithms*, Addison-Wesley, Reading, MA, 1983.

[2] R. ALBANESE AND G. RUBINACCI, *Integral formulation for 3D eddy-current computation using edge-elements*, IEE Proceedings A, 135 (1988), pp. 457–462.

[3] M.A. ARMSTRONG, *Basic Topology*, Springer, New York, 1983.

[4] D.N. ARNOLD, R.S. FALK, AND R. WINTHER, *Preconditioning in $H(\text{div})$ and applications*, Math. Comp., 66 (1997), pp. 957–984.

[5] C. BERGE, *Graphs and Hypergraphs*, North-Holland, Amsterdam, 1973.

[6] A. BOSSAVIT, *Computational Electromagnetism: Variational Formulation, Complementarity, Edge Elements*, Academic Press Electromagnetism Series 2, Academic Press, San Diego, 1998.

[7] F. BREZZI AND M. FORTIN, *Mixed and Hybrid Finite Element Methods*, Springer, New York, 1991.

[8] Z. CAI, R.R. PARASHKEVOV, T.F. RUSSELL, AND X. YE, *Domain decomposition for a mixed finite element method in three dimensions*, SIAM J. Numer. Anal., submitted.

[9] A.-L. CAUCHY, *Recherches sur les polyèdres – premier mémoire*, J. de l'École Polytechnique 9, (1813), pp. 68–86.

[10] G. CHAVENT, G. COHEN, J. JAFFRE, M. DUPUY, AND I. RIBERA, *Simulation of two-dimensional water flooding by using mixed finite elements*, Soc. Petroleum Eng. J, 24 (1984), pp. 382–390.

[11] Z. CHEN, R.E. EWING, R.D. LAZAROV, S. MALIASSOV, AND Y.A. KUZNETSOV, *Multilevel preconditioners for mixed methods for second order elliptic problems*, Numer. Linear Algebra Appl., 3 (1996), pp. 427–453.

[12] K.A. CLIFFE, I.G. GRAHAM, R. SCHEICHL, AND L. STALS, *Parallel computation of flow in heterogeneous media modelled by mixed finite elements*, J. Comput. Phys., 164 (2000), pp. 258–282.

[13] L.C. COWSAR, J. MANDEL, AND M.F. WHEELER, *Balancing domain decomposition for mixed finite elements*, Math. Comp., 211 (1995), pp. 989–1015.

[14] F. DUBOIS, *Discrete vector potential representation of a divergence-free vector field in three-dimensional domains: Numerical analysis of a model problem*, SIAM J. Numer. Anal., 27 (1990), pp. 1103–1141.

[15] R.E. EWING AND J. WANG, *Analysis of the Schwarz algorithm for mixed finite element methods*, RAIRO Modél. Math. Anal. Numér., 26 (1992), pp. 739–756.

[16] R.E. Ewing and J. Wang, *Analysis of multilevel decomposition iterative methods for mixed finite element methods*, RAIRO Modél. Math. Anal. Numér., 28 (1994), pp. 377–398.

[17] F. Hecht, *Construction d'une base de fonctions $P1$ non-conformes à divergence nulle dans* $\mathbb{R}^3$, RAIRO Anal. Numér., 15 (1981), pp. 119–150.

[18] F. Hecht, *Construction d'une base pour des éléments finis mixtes à divergence faiblement nulle*, manuscript, 1988.

[19] R. Hiptmair and R.H.W. Hoppe, *Multilevel methods for mixed finite elements in three dimensions*, Numer. Math., 82 (1999), pp. 253–279.

[20] L. Kettunen and L.R. Turner, *How to define the minimum set of equations for edge elements*, Int. J Appl. Electromagnetics Mater., 3 (1992), pp. 47–53.

[21] T.P. Mathew, *Schwarz alternating and iterative refinement methods for mixed formulations of elliptic problems.* I., Numer. Math., 65 (1993), pp. 445–468.

[22] T.P. Mathew, *Schwarz alternating and iterative refinement methods for mixed formulations of elliptic problems.* II., Numer. Math., 65 (1993), pp. 469–492.

[23] J.C. Nédélec, *Mixed finite elements in* $\mathbb{R}^3$, Numer. Math., 35 (1980), pp. 315–341.

[24] L.F. Pavarino and M. Ramé, *Numerical experiments with an overlapping additive Schwarz solver for $3D$ parallel reservoir simulation*, Int. J. Supercomp. Appl., 1 (1995), pp. 3–17.

[25] J.E. Roberts and J.M. Thomas, *Mixed and hybrid methods*, in Handbook of Numerical Analysis, Vol. 2, P.G. Ciarlet and J.L. Lions, eds., North-Holland, Amsterdam, 1991, pp. 523–639.

[26] T. Rusten and R. Winther, *A preconditioned iterative method for saddlepoint problems*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 887–904.

[27] R. Scheichl, *Iterative Solution of Saddle-Point Problems Using Divergence-Free Finite Elements with Applications to Groundwater Flow*, Ph.D. Thesis, University of Bath, Bath, UK, 2000.

[28] F. Thomasset, *Implementation of Finite Element Methods for Navier-Stokes Equations*, Springer, New York, 1981.

[29] B.I. Wohlmuth, A. Toselli, and O.B. Widlund, *An iterative substructuring method for Raviart-Thomas vector fields in three dimensions*, SIAM J. Numer. Anal., 37 (2000), pp. 1657–1676.

SIAM J. SCI. COMPUT.
Vol. 23, No. 5, pp. 1777–1798

ADAPTIVE ERROR CONTROL FOR STEADY STATE SOLUTIONS OF INVISCID FLOW*

LARS FERM† AND PER LÖTSTEDT†

Abstract. The steady state solution of the Euler equations of inviscid flow is computed by an adaptive method. The grid is structured and is refined and coarsened in predefined blocks. The equations are discretized by a finite volume method. Error equations, satisfied by the solution errors, are derived with the discretization error as the driving right-hand side. An algorithm based on the error equations is developed for errors propagated along streamlines. Numerical examples from two-dimensional compressible and incompressible flow illustrate the method.

Key words. finite volume method, discretization error, error control, error equation, Euler equations

AMS subject classifications. 65N15, 65N50

PII. S1064827500367452

1. Introduction.

Inadequate grid resolution is the most important contributor to inaccuracy in computational fluid dynamics (CFD) calculations [31]. This is a serious obstacle in the industrial use of CFD methods. Grid refinement studies reveal the sensitivity of the solution to the grid spacing. Such studies are expensive and time-consuming to make for an applications engineer, and usually only one grid is generated based on his or her experience. There is a need for solution-adaptive methods.

The errors in the numerical solution of the partial differential equations (PDEs) governing inviscid flow can be controlled by an adaptive method. By changing the grid spacing so that the solution errors fulfill given tolerances, the quality of the solution is guaranteed. Furthermore, computational work and memory is saved since the grid is not made unnecessarily fine. Two important ingredients in an adaptive algorithm are a way of estimating the solution error and a data structure for the grid that allows for modifications of the original grid. We combine computing the solution errors from the error equations with estimated discretization errors as source terms with refinement and coarsening in predetermined blocks of the grid.

The equations are discretized by finite volume methods in this paper. They are of second order on Cartesian grids. The refinement and the coarsening of the grid is determined by estimates of the solution error. Grid adaptation is necessary not only at discontinuities in the solution but also in the smooth parts. Differential equations are derived which are satisfied approximately by the errors. The right-hand side of the error equations is the discretization error in the finite volume method. This error is estimated by comparing the space discretization on two different grids. Then the error equations are solved numerically for the error in the solution. For certain variables, these equations are particularly simple and the error is propagated along the streamlines of the flow. An algorithm is devised for such errors. It is relatively easily implemented in existing codes and takes the sign of the error into account when the grid is adjusted so that a tolerance on the maximum error is fulfilled. The

*Received by the editors February 9, 2000; accepted for publication (in revised form) September 7, 2001; published electronically January 30, 2002. Financial support was obtained from the Swedish Research Council for Engineering Science and the Swedish Foundation for Strategic Research.

http://www.siam.org/journals/sisc/23-5/36745.html

†Department of Information Technology, Scientific Computing, Uppsala University, SE-75104 Uppsala, Sweden (ferm@tdb.uu.se, perl@tdb.uu.se).

discretization error generated at a point may propagate and influence the solution error far downstream. The large error in the solution in a cell is not necessarily caused by too coarse a grid there.

In [3], [4], [5], [30], [33], [40] the discretization or truncation error is estimated by comparing the discretization on different grids as in Richardson extrapolation. The adaptivity is based on the discretization error, but there is no direct control of the error in the solution. The error equations are solved in one-dimensional (1D) examples in [43] and the solution is computed on adapted grids based on error estimates. Three different ways of assessing the solution error are compared in [25]. The error is computed by comparing the solutions on different grids, by comparing the solution with a higher-order reconstruction, and by solving the error equations. Three other techniques are compared in [34]. The estimation of a local discretization error is the standard procedure in the time step regulation of the numerical solution of ordinary differential equations (ODEs) [23], [39].

Bounds on the errors in finite element methods are computed a posteriori using the solution to the adjoint problem in [17]. Then another linear system of differential equations has to be solved for the adjoint solution. The adjoint equation is solved for 1D problems in [41] and is used in an adaptive procedure to control the error in a finite volume discretization of the Euler equations. While a full control of the solution errors is offered by the adjoint solution, it is an expensive method in terms of computing time and storage, e.g., for time-dependent problems in three dimensions, and may not always be worth the effort. The idea in the numerical solution of ODEs controlling only the discretization error is probably a satisfactory procedure also for many PDEs. A general discussion of numerical errors and their estimation in CFD is found in [38].

Another possibility is to base the refinement on sensors such as pressure gradients or vorticity; see, e.g., [9], [12], [13], [35]. This approach yields good results in practice, but it is difficult to relate the sensor values to quantitative information on the solution errors. Wavelets are used in [21] to detect sharp features in the solution.

The computational domain is partitioned into a fixed number of blocks. The grid in each block is structured and all cells in a block are refined or coarsened following the error estimate. The data structure is simple since the neighbors of a block are the same in the original grid and all the adapted grids. Inside a block the organization of the data is also simple since the size of all cells in a block are changed when the grid there is refined or coarsened as in [18] and [29]. There will be jumps in the grid size only at the block boundaries, and interpolation is needed for the cell variables at the boundary. The interpolation of the variables at block boundaries is such that second-order accuracy is preserved in the cells adjacent to the boundary. Although we use structured grids the principles are applicable also to unstructured grids and hybrid grids with a mixture of blocks with structured and unstructured grids. Block partitioning is also suitable for parallelization of a code. Each processor solves the equations in a subset of the blocks with communication limited to the boundaries of blocks with a neighbor in another subset.

The flow equations are solved on structured grids with patches of refinement in [3], [4], [5], [12], [33], [40]. The refined parts follow the grid lines but can be positioned arbitrarily in the domain. The data structure is more complicated than ours but has greater flexibility since the patches can surround the points where refinement is needed with fewer extra cells. Parallel implementations are found in [14], [42]. Other equations related to the flow equations are solved with similar techniques and adaptivity in [10], [24], [36]. Individual cells are split for refinement in [13], [35]. The parental and adjacent cells are kept track of by a tree data structure. The idea here is the same for both structured and unstructured grids (cf. [9] and [35]). The price to be paid for the possibility of introducing cells precisely where they are needed is the administration of the advanced data structure and the indirect addressing of the data. For structured grids it is also complicated to obtain second-order accuracy in the discretization. Instead of introducing more cells, the grid points in one dimension are moved in [6] and [27] and the grid lines are moved in two dimensions in [28] to adapt the original grid. The advantage is that the available cells are clustered in areas where high resolution is required and fewer cells are used in smooth areas. A disadvantage is the distortion of the grid which may be severe in three dimensions around complicated geometries. Moreover, instabilities with moving grid algorithms are discussed in [27] and there is no guarantee that spurious solutions do not appear [6].

In the next section, the errors in finite volume methods are investigated and the estimate of the discretization error is justified. The error equations for compressible and incompressible inviscid flow problems are derived in section 3 from the results in section 2. The equations are particularly simple for the total pressure in incompressible flow and the entropy and the enthalpy in compressible flow. The adaptation algorithm is found in section 4. Finally, numerical solutions of the incompressible flow around a half-cylinder and compressible flow around an airfoil at transonic and subsonic speeds are computed with the algorithm.

2. Numerical errors in finite volume methods.

The PDE

$$\frac{\partial u}{\partial t} + \nabla \cdot \mathcal{F}(u) = 0, \qquad L(u) = 0 \text{ on } \partial\Omega, \quad u = u_0 \text{ at } t = 0, \tag{1}$$

with the flux vector $\mathcal{F}$ is discretized on $\Omega \times [0, t_{end}]$ by the finite volume method (FVM) in space and a linear multistep method or a Runge–Kutta method in time. The domain Ω with boundary $\partial\Omega$ is covered by m cells ω_j with a normal $\hat{n}$, boundary $\partial\omega$, and volume or area A_j. This section is devoted to the analysis of the solution errors in the discrete approximation of (1). The adaptive algorithm is developed only for steady state problems in this paper, but we include a short discussion of temporal errors in this section and the next for completeness. Adaptive methods in space and time are developed in a separate paper [29].

2.1. Space discretization.

Integrate (1) over ω_j and use Gauss's theorem to obtain

$$\int_{\omega_j} u_t d\omega + \int_{\partial\omega_j} \mathcal{F}(u) \cdot \hat{n} ds = 0.$$

In what follows, a coordinate as a subscript of a variable denotes differentiation with respect to that coordinate. Let $\bar{u}_j$ be the average of u in cell j and let G be defined by

$$G_j(u) = \bar{u}_{jt} + A_j^{-1} \int_{\partial\omega_j} \mathcal{F}(u) \cdot \hat{n} ds = \bar{u}_{jt} + F_j(u). \tag{2}$$

The solution u of (1) satisfies

$$G(u) = 0. \tag{3}$$

At time t_n, $\bar{u}_j$ is approximated by the numerical solution $\bar{v}_j^n$. By reconstruction from $\bar{v}_j^n$ (see, e.g., [1], [7]), v_j^n is created in Ω such that $v_j^n \in C^q(\Omega)$, $q \geq 0$, and

$$\bar{v}_j^n = A_j^{-1} \int_{\omega_j} v^n d\omega.$$

Let $\Psi_j^i(v)$ be the discrete flux in cell j on edge i, l be the number of edges, and $\hat{n}_i$ be the normal on edge i. Then the discretization of the integral in (2) is

$$\Phi_j(\bar{v}) = A_j^{-1} \sum_{i=1}^{l} \Psi_j^i(v) \cdot \hat{n}_i. \tag{4}$$

The discretization of the boundary condition on $\partial\Omega$ in (1) is

$$\Lambda_i(\bar{v}) = 0, \quad i = 1 : m_b, \tag{5}$$

at m_b boundary cells. Since the numerical scheme may need more boundary conditions than the analytical solution, $\Lambda(u)$ is usually not a direct approximation of $L(u)$.

2.2. Time discretization. Let us choose a linear multistep method, defined by its coefficients $\alpha_i, \beta_i, i = 0 : k$ (see [23]), for the approximation of (2) in time. Let Δt be the time step from t_{n-1} to t_n and introduce

$$\Gamma_\ell(\bar{v}^n) = \Delta t^{-1} \sum_{i=0}^{k} \alpha_i \bar{v}_\ell^{n-i} + \sum_{i=0}^{k} \beta_i \Phi_\ell(\bar{v}^{n-i}). \tag{6}$$

Then $\bar{v}^n$ satisfies

$$\Gamma(\bar{v}^n) = 0. \tag{7}$$

An explicit Runge–Kutta method with s stages is defined by the coefficients a_{ij}, $i = 2 : s$, $j = 1 : i - 1$, and b_i, $i = 1 : s$ (see [23]). In this case we have

$$\begin{aligned} k_1 &= \Phi_\ell(\bar{v}^{n-1}), \\ k_i &= \Phi_\ell(\bar{v}^{n-1} + \Delta t \textstyle\sum_{j=1}^{i-1} a_{ij} k_j), \quad i = 2 : s, \\ \Gamma_\ell(\bar{v}^n) &= \Delta t^{-1}(\bar{v}_\ell^n - \bar{v}_\ell^{n-1}) + \textstyle\sum_{i=1}^{s} b_i k_i. \end{aligned} \tag{8}$$

Also here $\bar{v}^n$ satisfies (7).

2.3. The error equation. As a measure of how well Γ_j approximates G_j in Ω take

$$\tau_j(w) = G_j(w) - \Gamma_j(\bar{w}) \tag{9}$$

for $w \in C^q$. If u is the solution of (3), then the truncation error of Γ_j is

$$\Gamma_j(\bar{u}) = -\tau_j(u). \tag{10}$$

In finite element analysis (see [17]), the finite element method solution v is inserted into the differential equation G to compute the *residual* $r(v) = G(v)$ used in the error control. Here, we have the reconstruction v from the average solution satisfying $\tau(v) = G(v)$ since $\Gamma(\bar{v}) = 0$ in (9). The numerical error δv^n in v^n satisfying (7) is

$$\delta v^n = v^n - u(t_n),$$

where $u(t_n)$ solves (3) at t_n. We have from (9) and (3)

$$G_j(u(t_n) + \delta v^n) - G_j(u(t_n)) = \tau_j(v^n), \quad j = 1 : m. \tag{11}$$

The coefficients of the linearized error equation and the right-hand side are evaluated at the numerical solution.

Since the numerical solution often needs more boundary conditions than the analytical solution, L is extended to $\tilde{L}$ on $\partial\Omega$ so that

$$\chi_i(w) = \tilde{L}_i(w) - \Lambda_i(\bar{w}). \tag{12}$$

Here, χ measures the discretization error in the boundary conditions. Consequently,

$$\tilde{L}_i(u(t_n) + \delta v^n) = \chi_i(v^n), \tag{13}$$

which is the boundary condition of δv^n on $\partial\Omega$. The initial condition is assumed to be exact.

We have derived an error equation for δv^n with accompanying boundary condition. The average error in a cell is

$$\delta \bar{v}_j^n = A_j^{-1} \int_{\omega_j} \delta v^n d\omega = \bar{v}_j^n - \bar{u}_j(t_n).$$

The numerical approximation fulfills a modification of the original equation. If G is linear, then from (11)

$$G_j \delta v^n = \tau_j(v^n). \tag{14}$$

There are two components of τ_j in (9). Subtract Γ_j in (6) or (8) from G_j in (2) and split the difference into two parts: one due to the temporal discretization (τ_j^t) and one due to the spatial discretization (τ_j^s). The result is

$$G_j(w) - \Gamma_j(\bar{w}) = \tau_j^t(w) + \tau_j^s(w),$$

where we are interested only in the spatial error in this paper.

Suppose that we wish to compute the error in a linear functional $\Xi(u)$. Introduce an inner product $(\cdot, \cdot)$ and a corresponding norm $\|\cdot\|$. Then by Riesz' representation theorem there is a unique ξ such that the error is

$$\delta\Xi = \Xi(u + \delta v) - \Xi(u) = (\xi, \delta v).$$

If we solve the error equation (14), then (at least formally)

$$\delta\Xi = (\xi, G^{-1}\tau).$$

In a posteriori error analysis [17], [41], the adjoint equation

$$G^* \varphi = \xi \tag{15}$$

is solved for φ. Then the discretization is chosen so that the errors satisfy $|(\varphi, \tau)| \leq \epsilon$. Since

$$\delta\Xi = (\xi, G^{-1}\tau) = (G^*\varphi, G^{-1}\tau) = (\varphi, \tau),$$

the estimate of $\delta\Xi$ is the same with the error equation (14) and the adjoint equation (15). The advantage with the adjoint equation is that weights are determined for the influence of different parts of τ on $\delta\Xi$. A disadvantage is that the solution of (15) is required. Another problem is if $\delta\Xi = \|\delta v\|^2$. Then $\xi = \delta v$, which is unknown in (15). The error equation is also a system of PDEs to solve, but our algorithm in section 4 only computes errors in variables that are particularly easy to calculate.

2.4. FVM error estimate. Assume that Ω is covered by triangles or quadrangles in two dimensions and by tetrahedrons or hexahedrons in three dimensions. Then the discretization error in space can be estimated asymptotically by comparing the residual Φ_j on different grids, as follows.

PROPOSITION 2.1. *Let cell ω_0 consist of the cells $\omega_j,\ j = 1 : p$, and let h be a measure of the cell size so that size$(\omega_0) = h$ and size$(\omega_j) = h/r,\ j = 1 : p$. Assume that $u \in C^1(\Omega)$ and that the space discretization in cell j satisfies*

$$\Phi_j(\bar{u}) = F_j(u) - \tau(u_{\omega_j}), \tag{16}$$

where u_{ω_j} is determined at a point in ω_j. Furthermore, assume that the discretization error satisfies

$$\tau(u) = h^\nu c(u) + h^{\nu+1} d(u), \tag{17}$$

where c is a linear operator

$$c(a_1u_1 + a_2u_2) = a_1c(u_1) + a_2c(u_2), \tag{18}$$

and that $\gamma_j = A_j/A_0,\ j = 1 : p$. Then in $\omega_j,\ j = 0 : p$,

$$\tau(u_{\omega_j}) = \tfrac{b_j}{r^\nu - 1}\left(\sum_{j=1}^{p}\gamma_j\Phi_j(\bar{u}_j) - \Phi_0\left(\sum_{j=1}^{p}\gamma_j\bar{u}_j\right)\right) + O(h^{\nu+1}),$$
$$b_0 = r^\nu,\ b_j = 1,\ j > 1.$$

Proof. For u in ω_0 we have

$$\Phi_0(\bar{u}_0) = \Phi_0\left(\sum_{j=1}^{p}\gamma_j\bar{u}_j\right) = F_0(u) - \tau(u_{\omega_0}). \tag{19}$$

Since $u \in C^1(\Omega)$ and by (17) and (18)

$$\tau(u_{\omega_0}) = h^\nu c(u_{\omega_0}) + h^{\nu+1}d(u_{\omega_0}) = h^\nu\sum_{j=1}^{p}\gamma_j c(u_{\omega_j}) + O(h^{\nu+1}). \tag{20}$$

A summation over ω_0 gives

$$\begin{aligned}\sum_{j=1}^{p}\gamma_j\Phi_j(\bar{u}) &= \sum_{j=1}^{p}\gamma_jF_j(u) - \sum_{j=1}^{p}\gamma_j\tau(u_{\omega_j})\\ &= F_0(u) - (h/r)^\nu\sum_{j=1}^{p}\gamma_jc(u_{\omega_j}) + O(h^{\nu+1}).\end{aligned} \tag{21}$$

We infer from (19), (20), and (21) that

$$h^{\nu}\sum_{j=1}^{p}\gamma_j c(u_{\omega_j}) = \frac{r^{\nu}}{r^{\nu}-1}\left(\sum_{j=1}^{p}\gamma_j \Phi_j(\bar{u}_j) - \Phi_0\left(\sum_{j=1}^{p}\gamma_j \bar{u}_j\right)\right).$$

By this expression the proposition follows.

The discretization error in space can be estimated by comparing the expression for the space derivatives on a fine and a coarse grid using the fine grid solution as in Proposition 2.1. The result is similar to what is usually referred to as a Richardson estimate [4], [33], [38]. In [43] the error equations are solved in 1D examples with a driving right-hand side approximating the discretization error for the particular spatial operator. Our evaluation of τ in the proposition is more general and estimates explicitly what happens when the grid size is changed.

3. Errors in the equations of inviscid flow. The equations satisfied by the reconstruction of the numerical solution in (11) are derived for compressible and incompressible flow. The error in the solution can then be estimated by numerical solution of these equations. An estimate of the effect of the discretization error, which can be controlled by changing the space discretization, is obtained on the solution error, which is desirable to keep within given bounds. We write the equations in the form (1) instead of the integrated form (2), since if the solution is smooth and satisfies (2), then it also satisfies (1). If the analytical and the numerical solutions are smooth, then the solution error satisfies PDEs similar to linearized versions of the original PDE.

Under certain assumptions, approximate error equations have constant coefficients and can be solved analytically. It follows from these equations that the solution error caused by a discretization error behaves differently depending on the component of the solution, the flow speed, and the dimension.

3.1. Error equations. The analytical solution of a variable u is denoted by $\hat{u}$ and the numerical error by δu. The reconstructed numerical solution is $u = \hat{u} + \delta u$. Let ρ be the density, $\mathbf{U} = (u, v, w)^T$ the velocity vector, p the pressure, E the total energy, and H the total enthalpy. Then the error equations for compressible flow on conservation form are [2]

$$\rho_t + (\rho u)_x + (\rho v)_y + (\rho w)_z = \tau_1, \tag{22a}$$

$$(\rho u)_t + (\rho u^2)_x + (\rho uv)_y + (\rho uw)_z + p_x = \tau_2, \tag{22b}$$

$$(\rho v)_t + (\rho uv)_x + (\rho v^2)_y + (\rho vw)_z + p_y = \tau_3, \tag{22c}$$

$$(\rho w)_t + (\rho uw)_x + (\rho vw)_y + (\rho w^2)_z + p_z = \tau_4, \tag{22d}$$

$$(\rho E)_t + (\rho uH)_x + (\rho vH)_y + (\rho wH)_z = \tau_5. \tag{22e}$$

These equations conserve mass, momentum, and energy. The system is closed by the internal energy e, the gas constant γ, and the relations

$$e = \frac{p}{(\gamma-1)\rho}, \quad q^2 = u^2 + v^2 + w^2, \quad E = e + \frac{1}{2}q^2, \quad H = E + \frac{p}{\rho}.$$

The equations in (22) can be written in nonconservation form using the substantial derivative D/Dt [2]:

$$\frac{D\rho}{Dt} + \rho\nabla\cdot\mathbf{U} = \tilde{\tau}_1, \tag{23a}$$

$$\rho\frac{Du}{Dt} + p_x = \tilde{\tau}_2, \tag{23b}$$

$$\rho\frac{Dv}{Dt} + p_y = \tilde{\tau}_3, \tag{23c}$$

$$\rho\frac{Dw}{Dt} + p_z = \tilde{\tau}_4, \tag{23d}$$

$$\rho\frac{De}{Dt} + p\nabla\cdot\mathbf{U} = \tilde{\tau}_5. \tag{23e}$$

The relations between τ in (22) and $\tilde{\tau}$ in (23) are

$$\tilde{\tau}_1 = \tau_1, \tag{24a}$$

$$\tilde{\tau}_2 = \tau_2 - u\tau_1, \tag{24b}$$

$$\tilde{\tau}_3 = \tau_3 - v\tau_1, \tag{24c}$$

$$\tilde{\tau}_4 = \tau_4 - w\tau_1, \tag{24d}$$

$$\tilde{\tau}_5 = \tau_1(q^2 - E) - (u\tau_2 + v\tau_3 + w\tau_4) + \tau_5. \tag{24e}$$

The entropy S of the flow is defined by

$$S = c_v \log(p/\rho^\gamma). \tag{25}$$

Here c_v is the specific heat at constant volume. By [2] and (23) we derive the differential equation for the computed entropy S:

$$\frac{DS}{Dt} = c_v(\gamma - 1)\left(\frac{\tilde{\tau}_5}{p} - \frac{\tau_1}{\rho}\right) = \tau_S. \tag{26}$$

In the numerical experiments in section 5, $c_v = 1$ and $\gamma = 1.4$.

Another simple equation is fulfilled by the enthalpy H. From (22e) and the definition of H we arrive at (see also [2])

$$\rho\frac{DH}{Dt} - \frac{\partial p}{\partial t} = \tau_5 - H\tau_1. \tag{27}$$

In incompressible flow, we let $\rho = 1$ and ignore the last equation in (23) to obtain the nonconservation form with $\tilde{\tau}$ from (24):

$$\nabla\cdot\mathbf{U} = \tilde{\tau}_1, \tag{28a}$$

$$\frac{Du}{Dt} + p_x = \tilde{\tau}_2, \tag{28b}$$

$$\frac{Dv}{Dt} + p_y = \tilde{\tau}_3, \tag{28c}$$

$$\frac{Dw}{Dt} + p_z = \tilde{\tau}_4. \tag{28d}$$

The total pressure P for an incompressible fluid is defined by

$$P = p + 0.5q^2. \tag{29}$$

Multiply the momentum equations in (28) by the velocity vector and use the definition of the substantial derivative to obtain

$$\frac{DP}{Dt} - \frac{\partial p}{\partial t} = u\tilde{\tau}_2 + v\tilde{\tau}_3 + w\tilde{\tau}_4. \tag{30}$$

Hence, if $\hat{P}$ is constant in Ω in a steady state solution, the error in P fulfills the linear equation

$$\mathbf{U} \cdot \nabla \delta P = u\tilde{\tau}_2 + v\tilde{\tau}_3 + w\tilde{\tau}_4, \tag{31}$$

where $\mathbf{U}$ is the computed solution.

3.2. Linearized equations. It is possible to simplify (23) if we make a few additional assumptions. For steady, compressible flow assume that

1. all derivatives of the analytical solution are small;
2. u, ρ, and p are of $O(1)$ but v and w are small;
3. the errors in the solution are small.

The first two assumptions are common in linearized flow analysis [2].

By (22) and the assumptions, we have, after ignoring small terms,

$$\begin{aligned} \hat{\rho}\nabla \cdot \delta\mathbf{U} + \hat{u}\delta\rho_x &= \tau_1, \\ \hat{\rho}\hat{u}\delta u_x + \delta p_x &= \tau_2 - \hat{u}\tau_1 = \tau_2', \\ \hat{\rho}\hat{u}\delta v_x + \delta p_y &= \tau_3, \\ \hat{\rho}\hat{u}\delta w_x + \delta p_z &= \tau_4, \\ \hat{\rho}\hat{H}\nabla \cdot \delta\mathbf{U} + \hat{H}\hat{u}\delta\rho_x + \hat{\rho}\hat{u}\delta H_x &= \tau_5. \end{aligned} \tag{32}$$

The error components $\delta\rho, \delta\mathbf{U}, \delta p$, and δH satisfy (32).

Using the definitions of the enthalpy, the speed of sound $a^2 = \gamma\hat{p}/\hat{\rho}$, and the Mach number $M = \hat{u}/a$ in (32), we arrive at

$$\begin{aligned} \nabla \cdot \delta\mathbf{U} - \tfrac{\hat{\rho}}{\gamma\hat{p}}\hat{u}^2\delta u_x &= (1 - M^2)\delta u_x + \delta v_y + \delta w_z \\ &= -\hat{u}\tau_2'/\hat{p} + (\gamma - 1)(\tau_5 - 0.5\hat{u}^2\tau_1)/(\gamma\hat{p}) = \tau_5'. \end{aligned} \tag{33}$$

We have obtained an approximate error equation for the velocity components. For incompressible flow, $M = 0$, and this is the error equation derived from the continuity equation.

The error in the pressure fulfills an equation similar to (33). Take the derivatives of the three momentum equations in (32) with respect to x, y, and z, multiply the first equation by $1 - M^2$, and add them together. Then we have from (33)

$$\begin{aligned} &(1 - M^2)\delta p_{xx} + \delta p_{yy} + \delta p_{zz} \\ &= (1 - M^2)\tau_{2x}' + \tau_{3y} + \tau_{4z} - \hat{\rho}\hat{u}\tau_{5x}' = \tau_p. \end{aligned} \tag{34}$$

The equation for the numerical error in the pressure is elliptic for subsonic flow ($M < 1$) and hyperbolic for supersonic flow ($M > 1$).

If we ignore the boundaries of the computational domain, we can write the solution of (34) for $M < 1$ by means of the fundamental solution g of the Laplace operator [37]. First, introduce the change of variables

$$x = \beta x_1, \quad y = y_1, \quad z = z_1, \quad \beta^2 = 1 - M^2, \tag{35}$$

and then let $\mathbf{r}$ be the vector

$$\mathbf{r} = (x_1 - \xi, y_1 - \eta, z_1 - \zeta)^T, \quad r = |\mathbf{r}|.$$

In (34), τ_p has the form of a divergence $\tau_p = \nabla \cdot \tau_p'$. Suppose that τ_p' stays bounded as $r \to \infty$. The error in the pressure can now be written with $g = -1/(4\pi r)$,

$$\delta p(x_1, y_1, z_1) = \int_\Omega \tau_p(\xi, \eta, \zeta) g \, d\Omega = -\int_\Omega \tau_p' \cdot \nabla g \, d\Omega + \int_{\partial\Omega} g\tau_p' \cdot \hat{n} \, dA, \tag{36}$$

using Gauss's theorem. Let $\partial\Omega$ expand so that the last term in (36) vanishes and $\Omega = \mathbf{R}^3$. Then we obtain

$$\delta p(x/\beta, y, z) = \frac{1}{4\pi} \int \frac{\tau_p' \cdot \mathbf{r}}{r^3} d\Omega. \tag{37}$$

The conclusion from (37) is that the influence of a discretization error $\tau_p'(\xi, \eta, \zeta)$ decays rapidly as $1/r^2$ when r grows.

In the hyperbolic case when $M > 1$ let $\beta^2 = M^2 - 1$ and use the same coordinate transformation as above. Then from (34)

$$\beta^2 \delta p_{xx} - \delta p_{yy} - \delta p_{zz} = -\tau_p. \tag{38}$$

With the fundamental solution to the wave equation (38) (see [37]) the pressure error is

$$\delta p(x/\beta, y, z) = -\frac{1}{2\pi} \int_\Omega \frac{\tau_p(\xi, \eta, \zeta)}{\sqrt{(x_1 - \xi)^2 - ((y_1 - \eta)^2 + (z_1 - \zeta)^2)}} d\Omega, \tag{39}$$

where $\Omega = \{(\xi, \eta, \zeta) | \; x_1 - \xi > \sqrt{(y_1 - \eta)^2 + (z_1 - \zeta)^2}\}$. Thus, for fixed y and z the contribution from $\tau_p(\xi, \eta, \zeta)$ vanishes when x grows.

The analysis can be extended to include also boundaries by introducing Green's function for both $M < 1$ and $M > 1$ [19].

3.3. Natural coordinates. The natural coordinates (s, n) in two dimensions are such that one coordinate s follows the streamlines and the other n is orthogonal to s. The transformation between the (x, y) and (s, n) systems is

$$\begin{aligned} dx &= u \, ds - v \, dn, \\ dy &= v \, ds + u \, dn. \end{aligned}$$

Note that $\mathbf{U} \cdot \nabla = \partial/\partial s$. Define the angle $\theta = \arctan(v/u)$. The speed of sound a is here given by $dp = a^2 \, d\rho$ and the Mach number is $M = q/a$. Then the momentum equations in (23) and (28) can be replaced by two equations in the (s, n) coordinate system. In two dimensions, the steady, compressible flow is governed by

$$(1 - M^2) p_s - \rho q^2 \theta_n = u\tilde{\tau}_2 + v\tilde{\tau}_3 - q^2 \tau_1 = \tilde{\tau}^s, \tag{40a}$$

$$p_n + \rho q^2 \theta_s = -v\tilde{\tau}_2 + u\tilde{\tau}_3 = \tilde{\tau}^n, \tag{40b}$$

and from (26) and (27) we infer that

$$\begin{aligned} S_s &= c_v(\gamma - 1)(\tilde{\tau}_5/p - \tau_1/\rho), \\ H_s &= (\tau_5 - H\tau_1)/\rho. \end{aligned} \tag{41}$$

For incompressible flow, take $M = 0$ and $\rho = 1$ in (40) and replace (41) by

$$P_s = u\tilde{\tau}_2 + v\tilde{\tau}_3 + w\tilde{\tau}_4. \tag{42}$$

Divide both equations in (40) by ρq^2; differentiate (40a) with respect to s and (40b) with respect to n and compute the sum. With $\kappa = 1/(\rho q^2)$ the result is

$$((1 - M^2)\kappa p_s)_s + (\kappa p_n)_n = (\kappa\tilde{\tau}^s)_s + (\kappa\tilde{\tau}^n)_n. \tag{43}$$

The equation is elliptic in p if the flow is subsonic and $M < 1$. If $M > 1$ as in supersonic flow, then (43) is hyperbolic. A similar equation is satisfied by θ. The other governing equations (41) or (42) are hyperbolic. The conclusion is that part of the solution error is transported along streamlines, as in (41) or (42), and part of the error is spread in an elliptic fashion, as in (43), in subsonic flow. If the variation in p, q, and ρ is small in (43), then subtract the analytical solution satisfying (43) without the error terms on the right-hand side as in (11). For the error in the pressure δp we have approximately

$$(1 - M^2)\kappa\delta p_{ss} + \kappa\delta p_{nn} = (\kappa\tilde{\tau}^s)_s + (\kappa\tilde{\tau}^n)_n.$$

Under the assumptions in section 3.2 the coordinates s and n coincide with the x and y coordinates in (34) and the error equations are the same.

3.4. Flow in a channel. The compressible flow in a channel (see Figure 1) is perturbed by a right-hand side $\tau_2 = 1$ in a circle Ω_c with radius 0.2 at $(x_c, y_c) = (1.25, 1.5)$ simulating the effect of a discretization error in the momentum equation in the x-direction in (22b). The grid is uniform with 180×76 cells. The solution is computed at a subsonic speed, $M = 0.6$, and a supersonic speed, $M = 2$. The width of the channel is changed after $x = 1$ and a shock is generated by the lower wall at that point in the supersonic case.

The solution for $\tau_2 = 0$ is subtracted from the solution when $\tau_2 = 1$ and plotted for different variables in Figure 1. The isolines of the difference illustrate how the discretization error at $(1.25, 1.5)$ generates errors in the solution variables. The behavior is different depending on the Mach number. An explanation is offered by the error equations.

Except for the shock, the flow in Figure 1 satisfies the assumptions for the linear analysis in section 3.2. The error equations in two dimensions follow from (32), (33), and (34):

$$\hat{u}\delta\rho_x + \hat{\rho}(\delta u_x + \delta v_y) = 0, \tag{44a}$$

$$\hat{\rho}\hat{u}\delta u_x + \delta p_x = \tau_2, \tag{44b}$$

$$\hat{\rho}\hat{u}\delta v_x + \delta p_y = 0, \tag{44c}$$

$$(1 - M^2)\delta p_{xx} + \delta p_{yy} = (1 + M^2(\gamma - 1))\tau_{2x}, \tag{44d}$$

$$(1 - M^2)\delta u_x + \delta v_y = -\hat{u}\tau_2/\hat{p}. \tag{44e}$$

Let $M < 1$ and $\beta = \sqrt{1 - M^2}$ and introduce the coordinate transformation (35). Following [37], the solution of (44d) in free space is

$$\delta p(x/\beta, y) = \sigma \int \tau_{2\xi} g(x_1 - \xi, y - \eta)d\Omega = -\sigma \int \tau_2 g_\xi(x_1 - \xi, y - \eta)d\Omega, \tag{45}$$

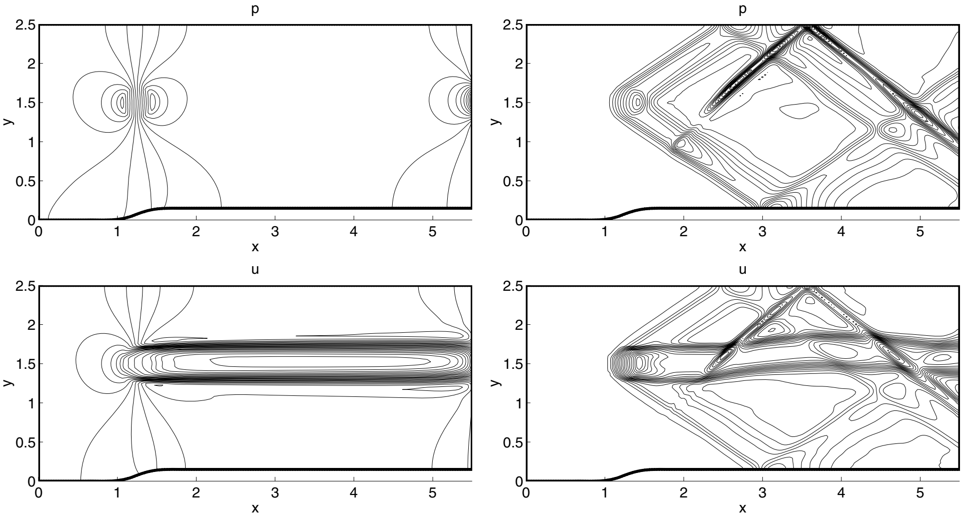


FIG. 1. *Isolines of the perturbation in the pressure p (upper) and velocity component u (lower) for subsonic (left) and supersonic (right) flow.*

where $g(x, y) = \log(x^2 + y^2)/4\pi$ is the fundamental solution in two dimensions and $\sigma = (1 + M^2(\gamma - 1))/\beta$. Since the support of τ_2 is so small we have approximately

$$\delta p \approx \frac{\sigma \tau_2 (x - x_c)}{2\pi\beta(((x - x_c)/\beta)^2 + (y - y_c)^2)}. \tag{46}$$

In subsonic flow, δp will decay at least as $1/r$ outside Ω_c.

In supersonic flow, let $\beta = \sqrt{M^2 - 1}$. The fundamental solution is

$$g(x, y) = \begin{cases} 1/2, & |y| < x, \\ 0, & |y| \geq x \end{cases}$$

(see [37]). Hence,

$$\begin{aligned} \delta p(x/\beta, y) &= -0.5\sigma \int_{-\infty}^{x_1} \int_{x_1 - \xi > |y_1 - \eta|} \tau_{2\xi} \, d\eta d\xi \\ &= -0.5\sigma \int_{-\infty}^{\infty} \int_{-\infty}^{x_1 - |y_1 - \eta|} \tau_{2\xi} \, d\xi d\eta = -0.5\sigma \int_{-\infty}^{\infty} \tau_2(x_1 - |y_1 - \eta|, \eta) d\eta. \end{aligned} \tag{47}$$

Outside the small circle Ω_c, $\tau_2 = 0$ and we have

$$\delta p(x, y) \approx -0.5\sigma\tau_2(x_c, y_c), \text{ when } x \geq x_c, \ x - x_c = \beta|y - y_c|. \tag{48}$$

Contrary to subsonic flow, δp will not decay outside Ω_c on two rays from (x_c, y_c). This behavior is also different from what we have in three dimensions in (39) reflecting the different properties of the wave equation in odd and even dimensions.

From (44b) we conclude that

$$\delta u = \left(\int_{-\infty}^{x} \tau_2 d\xi - \delta p \right) / (\hat{\rho}\hat{u}). \tag{49}$$

The error δu consists of two parts: one part is convected downstream from the source τ_2 and the other part is spread like the error in p. If we try to reduce δu where it is large by refining the grid for $x \geq 2$ in the channel to the outflow boundary (see Figure 1), the adaptation will have little effect on the solution error. It will decrease only after refinement around the source of the discretization error at (x_c, y_c).

An equation for δv similar to (44d) can be derived by combining (44c), (44b), and (44e). The error in ρ has one propagating part and one part depending on δp in the same manner as δu.

The solutions for δp and δu in (46), (48), and (49) agree with the solutions in Figure 1 locally around (x_c, y_c) when the influence of the boundaries is negligible. At the supersonic speed, the waves are reflected in the channel walls and the shock position is shifted above $y \approx y_c$ and in the reflected part by the convected error. The perturbations in δp for $M = 0.6$ at the downstream boundary is explained by the outflow boundary conditions there.

4. Numerical algorithms. In this section we describe the space discretization briefly and how the grid is refined given the computed discretization errors.

4.1. Space discretization. The flow equations are solved on a structured grid in two dimensions. The computational domain is partitioned into a number of blocks. The location of these blocks is predetermined and the edges of a block follow grid lines. The grids in each block can be refined or coarsened. At the block boundaries, jumps are allowed in the grid size. The step length along a boundary may increase by a factor of 2 when crossing the boundary. By changing all cells in a block, the administration of the adaptivity is simplified and the interpolation of data between cells of different size is concentrated at the block boundaries. There is a waste of cells since all of them are not needed for the accuracy. In [4], the refined patches can be placed without restrictions, but the bookkeeping is more complicated in such an algorithm. Every cell in [13] can be refined. This approach requires an extra data structure, and second-order accuracy is difficult to obtain.

The flux vectors in (4) are computed either with the Jameson scheme [26] or with the Osher scheme [32]. Both are second-order accurate on Cartesian grids with constant step sizes in each direction.

Boundary conditions are approximated by introducing ghost cells. In the steady state solutions, the variables in these cells are either given by the boundary data in (1) or obtained by extrapolation from the interior of the domain. The order of the extrapolation is important for the accuracy of the solution, in particular at solid walls. For the compressible equations, the extra numerical conditions are calculated according to [11].

For characteristics starting at a boundary going into the domain it is sufficient to have the boundary condition satisfied to order $\nu-1$ if ν is the order of the discretization in the interior without losing accuracy [22]. Along a solid wall we have a streamline in the Euler equations and the behavior of the error in the total pressure or the entropy is governed by (30) and (26). If the error in the boundary condition is of order $\nu - 1$, then the error in P or S will also be of that order. This is discussed in [19].

Interpolation is necessary at block faces where there is an increase in the grid size. In the coarse ghost cells, the average is computed by summing the averages in the corresponding fine cells with weights proportional to the fine cell areas. The fine ghost cells are computed by combining the neighboring coarse cell values, so that the discretization error is of second order on Cartesian grids in the fine cells adjacent to the boundary. Details of the interpolation are found in [29].

The steady state solution of incompressible flow is computed by adding $\partial p/\partial t$ to the continuity equation (28a) as in [8]. Then the time-dependent equations are integrated for both compressible and incompressible flow by a Runge–Kutta scheme (8) until the time derivatives vanish [26].

4.2. Adaptation algorithm. The solution on the original grid is computed first. Then that solution is used to compute the solution error with the estimates in Proposition 2.1 and the error equations in section 3. The choice of grid size in the different blocks of the computational domain is determined by these estimates. The error equations are solved also in [43] for estimation of the solution error but without coupling them to a fully adaptive procedure.

In the following algorithm we control the errors propagated along streamlines for three reasons:

1. The errors in S, H, and P are often used as measures of the quality of inviscid flow solutions, and they have this property; see (26), (27), and (30).
2. It is a simple way of introducing a control of the maximum solution error.
3. It is possible to take the signs of the discretization errors into account, thus avoiding unnecessarily fine grids.

It is more complicated to solve the elliptic or hyperbolic error equation based on (43) for the pressure error. An approximate solution is computed in one of the numerical examples in the next section. In the linearized equations (34) and (44), δp, caused by a source, decays rapidly as the distance from it grows when $M < 1$ (see Figure 1). In that case, it may be sufficient to control the size of the local discretization error. It is usually concentrated to the solid boundaries. It is shown in [19] using the error equation (34) in two dimensions that the effect on δp of such errors decays from straight and curved boundaries for subsonic and incompressible flow but is propagated without damping for supersonic flow (cf. (46), (48), and Figure 1).

Assume that a maximum error is required to be below a certain tolerance ϵ in a variable, whose error is propagated along streamlines as in (26), (27), (31). Let s denote the coordinate along the streamlines. They pass through a number of block interfaces at $s = s_0, s_1, s_2, \dots$. The contribution to the error in block k is $I_k(s) = \int_{s_{k-1}}^{s} \tau\, ds$ for $s_{k-1} \le s \le s_k$. The accumulated error is denoted by $E(s)$ with $E(s_0) = 0$, and in block k

$$E(s) = \sum_{j=1}^{k-1} I_j(s_j) + I_k(s) = E(s_{k-1}) + I_k(s) \ \text{ for } \ s_{k-1} \le s \le s_k. \tag{50}$$

The algorithm for choosing the grid size in the blocks along a streamline is initialized by calculating the integrals I_k and the corresponding estimate of the accumulated error $E(s)$. Then the adaptive algorithm is, with $\nu = 2$,

```
(51)
while max$|E(s)| > \epsilon$
    Locate the maximum of $|E(s)|$ : $s_{M-1} < s_{max} < s_M$
    $I_k^+(s) = I_k(s)\,\mathrm{sign}\,(E(s_{max}))$ for all $k$
    Find the block $k_m$ with the maximum of
        $I_1^+(s_1), I_2^+(s_2), \dots, I_{M-1}^+(s_{M-1}), I_M^+(s_{max})$
    Refine block $k_m$ : $I_{k_m}(s) := I_{k_m}(s)/2^\nu$
    Update $E(s)$
end while
```

The error may increase in occasional steps in the process when the maximum error changes sign. No jumps in grid size at the block boundaries are allowed to be larger than factor 2. To fulfill this condition, the grid sometimes has to be refined in some blocks to an unnecessarily fine level. If the error along a number of streamlines in a block is controlled, then the smallest grid size given by the algorithm applied to each streamline is chosen.

An alternative would be to estimate $|E(s)|$,

$$|E(s)| \leq \sum_{j=1}^{k-1} |I_j(s_j)| + |I_k(s)|,$$

from (50) and refine the grid in a block k_m where $|I_{k_m}| = \max_k |I_k|$. A problem with that approach is that sometimes the inequality is not sufficiently sharp and the grid may be too fine if the sign of the contribution from each block is ignored. Even if the bound decreases, $E(s_{max})$ may increase, e.g., if $E(s_{max}) > 0$ and $I_k > 0$, $k \neq k_m$ but $I_{k_m} < 0$.

5. Numerical results. The steady state solution of the Euler equations of inviscid flow is computed around 2 two-dimensional objects: the upper part of a cylinder and the wing profile NACA 0012. The flow around the half-cylinder is incompressible and the numerical solution is compared to the analytical solution of the linearized potential flow equation. For the second geometry, the flow is compressible in a standard transonic test case and a subsonic case. The enthalpy and the entropy in the numerical solution are compared to the exact constant value of the enthalpy and to the exact entropy, which is constant at least in parts of the domain.

5.1. Incompressible flow. The time-independent solution of the incompressible Euler equations in conservation form is computed in two dimensions around the upper part of a cylinder. The grid is cylindrical and partitioned into blocks. The system of equations is discretized by the scheme of Jameson, Schmidt, and Turkel [26]. Fourth-order extrapolation at the boundaries is used to supply missing numerical boundary conditions at the cylinder.

The solution is compared to the analytical solution of the linearized potential flow equation. Let Θ be the potential and n the normal direction on the cylinder. Then

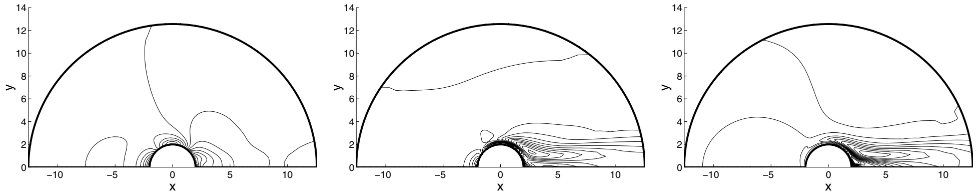
$$\begin{aligned} &\Delta\Theta = 0 \text{ in } \Omega, \quad \mathbf{U} = \nabla\Theta, \quad P = p + 0.5q^2 = \text{const}, \\ &\mathbf{U} = (1,0)^T \text{ at } \infty, \quad n \cdot \mathbf{U} = 0 \text{ on cylinder.} \end{aligned} \tag{52}$$

With the origin at the center of the cylinder and r the radius, the solution to (52) is

$$u = 1 + r^2(y^2 - x^2)/(x^2+y^2)^2, \quad v = -2r^2xy/(x^2+y^2)^2. \tag{53}$$

If there is no vorticity in the Euler solution, then (53) is the steady state solution of (28) with $\tilde{\tau} = 0$. The analytical solution (53) defines the boundary conditions at the outer boundary.

The difference between the analytical solution and the Euler solution around the cylinder on a uniform grid is plotted in Figure 2. The error propagation downstream can as expected be observed in the variables u and P. Most of the error in p is generated at the surface of the cylinder, and it decays in an elliptic way when we go away from the cylinder. The behavior of the errors in u, p, and P is what we expect from the analysis in sections 3.2 and 3.4.



p P u

FIG. 2. *Error distribution around the cylinder.*

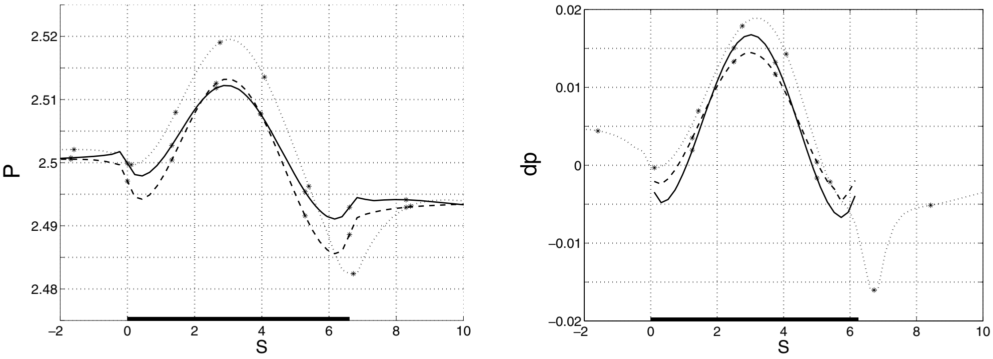


FIG. 3. *Estimated (dashed and solid) and true values (dotted) of P (left) and δp (right) along a streamline close to the surface of the cylinder.*

In Figure 3, the errors δP and δp are estimated on a streamline close to the cylinder ($s \in [0, 2\pi]$) and to $y = 0$ in front of ($s < 0$) and behind the cylinder ($s > 2\pi$). The numerical solution is computed on a coarse grid with 30×18 cells divided into 15 blocks. For the total pressure P, (31) is integrated with $P = 2.5$ as initial condition with two different right-hand sides. Firstly, the discretization error is computed by inserting the analytical solution into the discretization as in (10). Secondly, τ is estimated by Proposition 2.1. The difference in τ_1 between the two alternatives is shown in Figure 4. The first (dashed) and the second (solid) estimates are close. The difference is comparatively small in Figure 3 between the first (dashed) and the second (solid) right-hand sides and the true value of P (dotted). The error in P is somewhat underestimated. This is partly explained by the fact that the estimates are obtained via integrations along the line of symmetry $y = 0$ and the surface of the cylinder. As we can see in Figure 4, this line does not follow the streamline at the left stagnation point, and an error there cannot be corrected downstream. The error in p in Figure 3 is calculated by integrating (40b) from the outer boundary with initial condition $p = 0$ assuming $\rho q^2 \theta_s$ to be negligible. It is possible to subtract the estimated errors from the solution and obtain "superconvergence" as in [41] and a more accurate solution, but without an error estimate in the final solution.

We let the error tolerance be $\epsilon = 4.3 \cdot 10^{-3}$ and apply the adaptive algorithm in section 4.2 once to our example. The grid when P is controlled so that $|\delta P| \leq \epsilon$ is plotted in Figure 5, along with obtained errors on that grid. The abscissa follows the streamline along the cylinder as in Figures 3 and 4. The cylinder surface is marked by a thick black line. The error is somewhat underestimated in P.

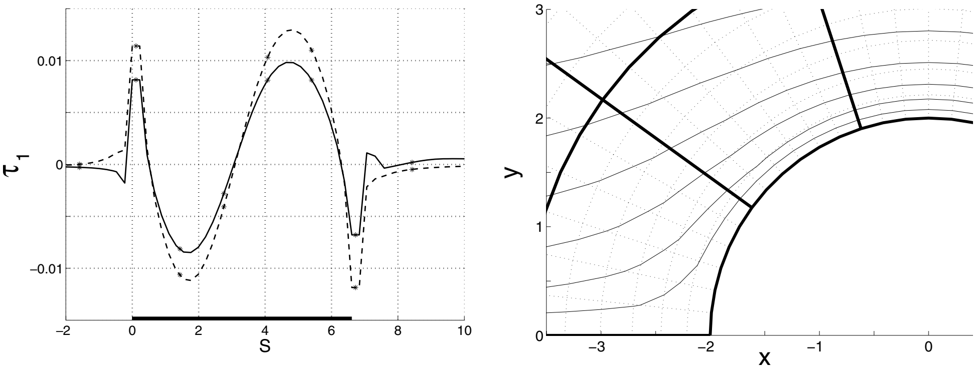


FIG. 4. *The discretization error τ_1 used for the error estimation of P (left) and streamlines close to the cylinder (right).*

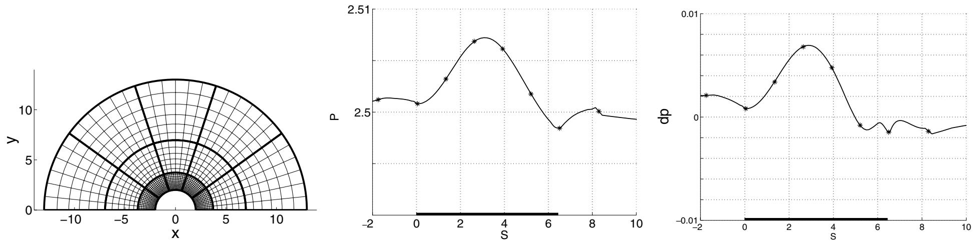


FIG. 5. *Grid for control of P and the measured values of P and δp.*

The grid for control of the error of p is generated by (51) and integration of (40b) and is plotted in Figure 6. The coarse block in the left part of the cylinder gives rise to a large error in P, which is convected downstream resulting in a peak in δp at the right stagnation point. A reliable estimate of δp requires control of P (see Figure 7) where the goal is $\max(\delta P, \delta p) \le \epsilon$. To reduce δP, it does not help to refine the grid in the top block on the cylinder, where δP has its maximum. Instead, the grid should be finer, where δP grows most rapidly (cf. section 3.4 and Figure 1).

The solution error was computed in the L_2 norm $\|\cdot\|$ along the cylinder. A small error on solid surfaces is often desirable in applications and that is the error we control here. The errors in P and p were $\|\delta P\| = 1.18 \cdot 10^{-2}$ and $\|\delta p\| = 1.86 \cdot 10^{-3}$ in the final adapted grid in Figure 7 which has 1836 cells. For comparison, a uniform grid was created with 2160 cells. There, the solution errors were $\|\delta P\| = 0.986 \cdot 10^{-2}$ and $\|\delta p\| = 3.28 \cdot 10^{-3}$. With nearly equal number of cells, the error in p is reduced by almost a factor of 2 and the error in P is about the same in the adapted grid compared to the uniform grid. The gain in adapting the grid is not very dramatic in this example because the flow is rather smooth. The error reduction is more substantial in the following numerical experiments.

5.2. Compressible flow. The steady state solution of the compressible Euler equations ((22) with $\tau = 0$) is calculated around the NACA 0012 airfoil. An original C-grid is partitioned into blocks and refined with the adaptive algorithm in section 4.2. The equations are discretized with the Osher scheme [32] with reconstruction for

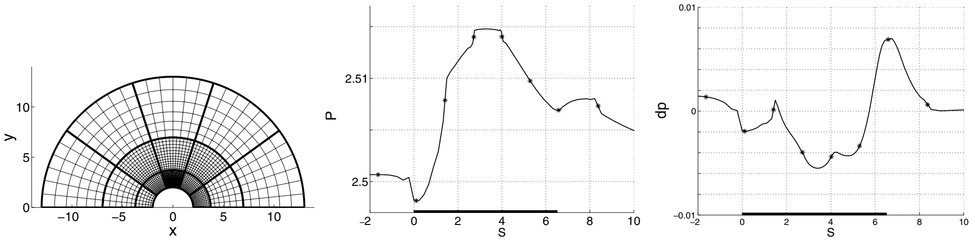


FIG. 6. *Grid for control of p and the measured values of P and δp.*

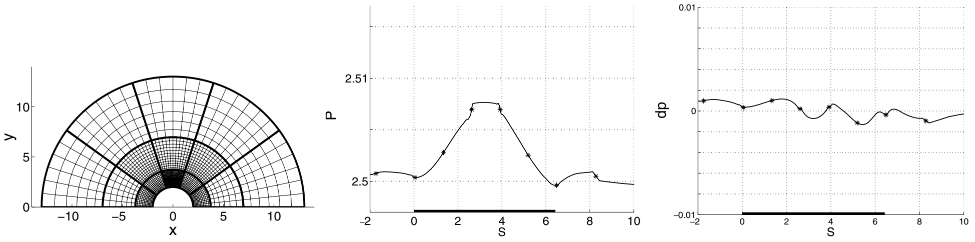


FIG. 7. *Grid for control of p and P and the measured values of P and δp.*

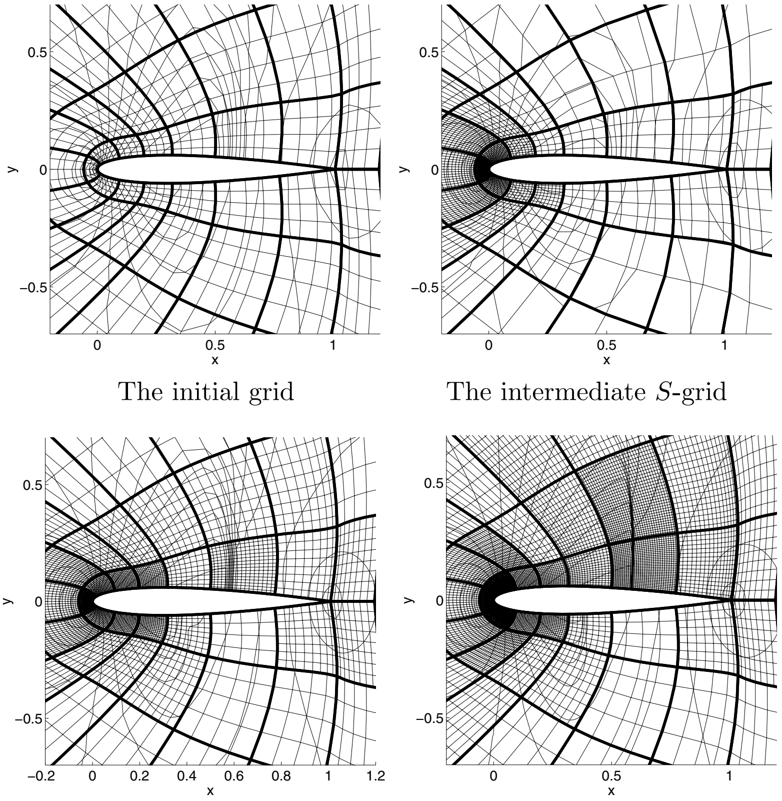
second-order accuracy. The error equations in section 3 break down at discontinuities such as shocks, but the adaptive algorithm (51) still works well.

In the first example, the flow is transonic with $M = 0.8$ and the angle of attack $\alpha = 1.25^o$ and $\epsilon = 5 \cdot 10^{-2}$ in (51). The adaptation algorithm (51) is applied twice: first on the original grid to generate an intermediate grid and then again to generate the final grid. Either the error in the enthalpy H or the error in the entropy S is monitored. The equations (26) and (27) are integrated for the errors in H and S along the streamlines on the upper and lower surfaces of the airfoil. Then the change of H or S in a block determines the grid refinement in (51). It is important that the refinement is based on the difference of H or S between the block boundaries and not the peak values of them in the blocks (cf. Figures 5, 6, and 7).

The initial grid and the intermediate grids generated from δH and δS are shown in Figure 8 together with the isobars of the solution. The estimated error in S is not sufficiently large to motivate a refinement at the shock on the upper part of the profile. On the contrary, δH captures the shock on the upper and lower side, and the algorithm (51) introduces grid refinement both at the leading edge and the shocks. The importance of a good resolution at a shock is discussed in [15] and [16]. Errors of low order are created in a shock and convected downstream.

Figures 9 and 10 display the computed δH and S on the upper and lower side of the airfoil with the initial (dotted), intermediate (dashed), and fine (solid) H-grids. The plots are extended to the outer boundary, and the wing is indicated by a thick line. The entropy has a jump at the shock on the upper side. In the wake, an average is taken between S above and below the trailing edge.

As in the previous example, the solution error in the adapted grid with 43632 cells is compared to the error in a uniform grid with 48384 cells. The block at the leading edge has 64×64 and 24×24 cells, respectively. Then along the profile $\|\delta H\|$



The initial grid

The intermediate S-grid

The intermediate H-grid

The fine H-grid

FIG. 8. *Isobars for transonic flow on the initial, intermediate, and fine grids. Every second grid line is omitted in the plot of the fine H-grid.*

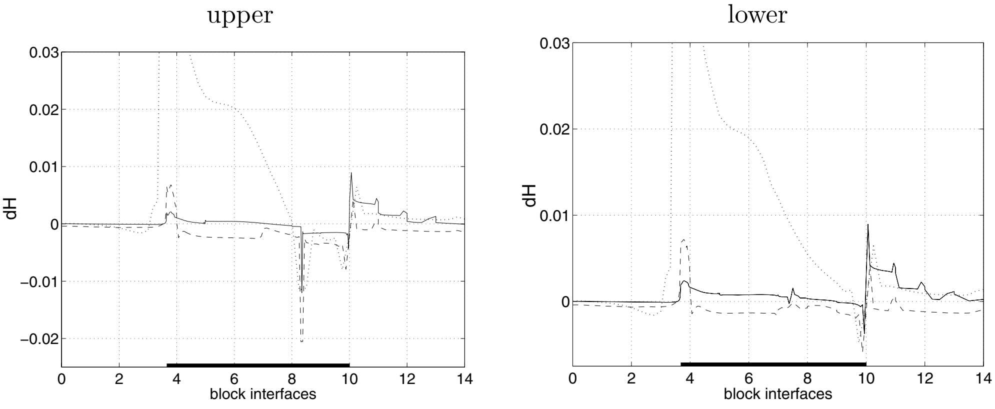


FIG. 9. *Error in the enthalpy along the streamlines at the surface of the wing profile on the initial (dotted), intermediate (dashed), and fine (solid) H-grids.*

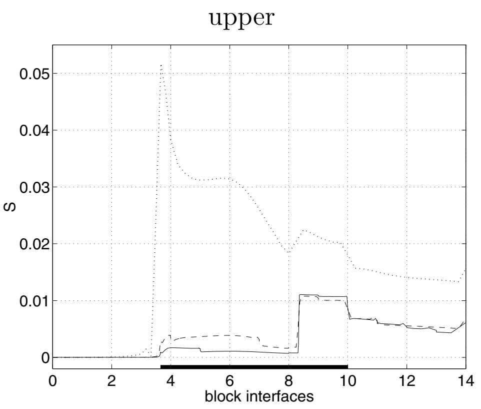
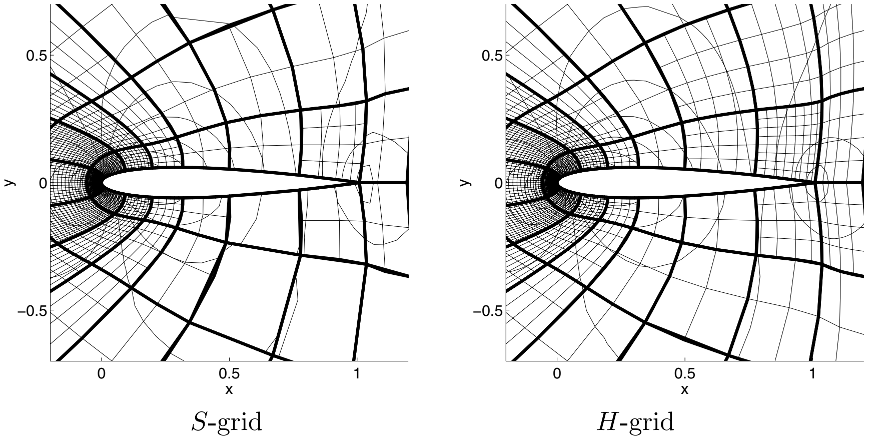


FIG. 10. *The entropy along the streamlines at the surface of the wing profile on the initial (dotted), intermediate (dashed), and fine (solid) H-grids.*



S-grid

H-grid

FIG. 11. *Isobars for subsonic flow on the final grids. Every second grid line is omitted.*

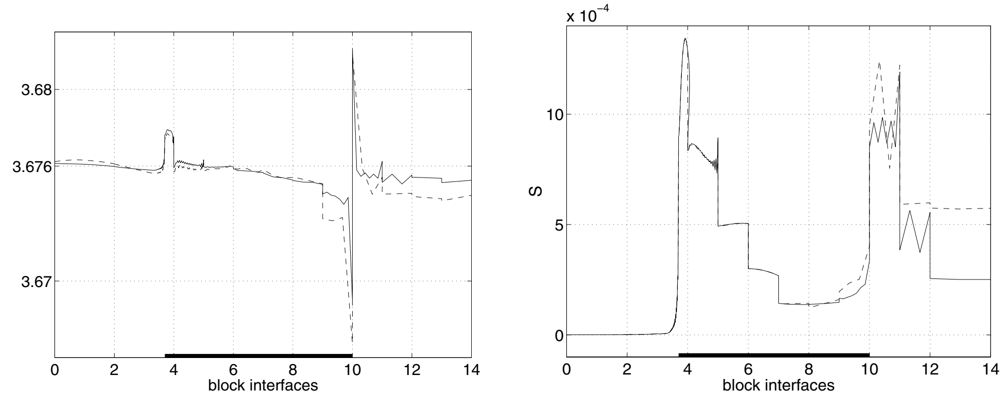


FIG. 12. *The enthalpy (left) and entropy (right) along the lower side of the wing profile computed on the S-grid (dashed) and the H-grid (solid).*

is $3.2 \cdot 10^{-3}$ in the adapted grid and $1.4 \cdot 10^{-2}$ in the uniform grid. The error is clearly reduced by redistributing the cells.

Subsonic flow is computed in the last example with $M = 0.5$, $\alpha = 1.25^o$, and $\epsilon = 2 \cdot 10^{-3}$. The initial grid is the same as in Figure 8. The results after applying the algorithm (51) twice are found in Figure 11. The grids are generated using the enthalpy or the entropy as a measure of the error. The enthalpy is more sensitive to the grid density at the trailing edge. The errors are compared on the two grids in Figure 12, where the solid line corresponds to the H-grid and the dashed line to the S-grid. In [20], the solution on the adapted S-grid is compared to the solution on a uniform grid. The adapted and uniform grids have 12816 and 50688 cells. The error $\|\delta S\|$ is reduced by 82% on the upper side of the wing profile, and the CPU time with multigrid iteration is slightly less with adaptation compared to uniform cell distribution. Better accuracy is achieved and memory is saved at about the same cost in computing time. In a three-dimensional example with transonic flow over a wing, almost the same solution is obtained with an adapted grid compared to a uniform grid with only 12% of the cells and 8% of the CPU time.

REFERENCES

[1] R. Abgrall, *On essentially non-oscillatory schemes on unstructured meshes: Analysis and implementation*, J. Comput. Phys., 114 (1994), pp. 45–58.

[2] J. D. Anderson, Jr., *Modern Compressible Flow*, 2nd ed., McGraw-Hill, New York, 1990.

[3] J. Bell, M. Berger, J. Saltzman, and M. Welcome, *Three-dimensional adaptive mesh refinement for hyperbolic conservation laws*, SIAM J. Sci. Comput., 15 (1994), pp. 127–138.

[4] M. Berger and P. Colella, *Local adaptive mesh refinement for shock hydrodynamics*, J. Comput. Phys., 82 (1989), pp. 64–84.

[5] M. Berger and A. Jameson, *Automatic adaptive grid refinement for the Euler equations*, AIAA J., 23 (1985), pp. 561–568.

[6] C. J. Budd, G. P. Koomullil, and A. M. Stuart, *On the solution of convection-diffusion boundary value problems using equidistributed grids*, SIAM J. Sci. Comput., 20 (1998), pp. 591–618.

[7] J. Casper and H. L. Atkins, *A finite-volume high-order ENO scheme for two-dimensional hyperbolic systems*, J. Comput. Phys., 106 (1993), pp. 62–76.

[8] A. J. Chorin, *A numerical method for solving incompressible viscous flow problems*, J. Comput. Phys., 2 (1967), pp. 12–36.

[9] S. D. Connell and D. G. Holmes, *Three-dimensional unstructured adaptive multigrid scheme for the Euler equations*, AIAA J., 32 (1994), pp. 1626–1632.

[10] P. Colella, M. R. Dorr, and D. D. Wake, *Numerical solution of plasma fluid equations using locally refined grids*, J. Comput. Phys., 152 (1999), pp. 550–583.

[11] A. Dadone and B. Grossman, *Surface boundary conditions for the numerical solution of the Euler equations*, AIAA J., 32 (1994), pp. 285–293.

[12] R. L. Davis and J. F. Dannenhoffer III, *3-D Adaptive Grid-Embedding Euler Technique*, AIAA paper 93-0330, AIAA, Washington, DC, 1993.

[13] D. De Zeeuw and K. G. Powell, *An adaptively refined Cartesian mesh solver for the Euler equations*, J. Comput. Phys., 104 (1993), pp. 56–68.

[14] J. De Keyser, K. Lust, and D. Roose, *Run-time balancing support for a parallel multiblock Euler/Navier-Stokes code with adaptive refinement on distributed memory computers*, Parallel Comput., 20 (1994), pp. 1069–1088.

[15] G. Efraimsson and G. Kreiss, *A remark on numerical errors downstream of slightly viscous shocks*, SIAM J. Numer. Anal., 36 (1999), pp. 853–863.

[16] B. Engquist and B. Sjögreen, *The convergence rate of finite difference schemes in the presence of shocks*, SIAM J. Numer. Anal., 35 (1998), pp. 2464–2485.

[17] K. Eriksson, D. Estep, P. Hansbo, and C. Johnson, *Introduction to adaptive methods for differential equations*, Acta Numer., (1995), pp. 121–123.

[18] L. Ferm and P. Lötstedt, *Blockwise adaptive grids with multigrid acceleration for compressible flow*, AIAA J., 37 (1999), pp. 121–123.

[19] L. Ferm and P. Lötstedt, *On numerical errors in the boundary conditions of the Euler equations*, Appl. Math. Comput., to appear.

[20] L. Ferm and P. Lötstedt, *Efficiency in the adaptive solution of inviscid compressible flow problems*, Nonlinear Anal., 47 (2001), pp. 3467–3478.

[21] M. Gerritsen and P. Olsson, *Designing an efficient solution strategy for fluid flows* II. *Stable high-order central finite difference schemes on composite adaptive grids with sharp shock resolution*, J. Comput. Phys., 147 (1998), pp. 293–317.

[22] B. Gustafsson, *The convergence rate for difference approximations to mixed initial boundary value problems*, Math. Comput., 29 (1975), pp. 396–406.

[23] E. Hairer, S. P. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations*, 2nd ed., Springer-Verlag, Berlin, 1993.

[24] R. D. Hornung and J. A. Trangenstein, *Adaptive mesh refinement and multilevel iteration for flow in porous media*, J. Comput. Phys., 136 (1997), pp. 522–545.

[25] C. Ilinca, X. D. Zhang, J.-Y. Trépanier, and R. Camarero, *A comparison of three estimation techniques for finite-volume solutions of compressible flow*, Comput. Methods Appl. Mech. Engrg., 189 (2000), pp. 1277–1294.

[26] A. Jameson, W. Schmidt, and E. Turkel, *Numerical Solutions of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time-Stepping Schemes*, AIAA-paper 81-1259, AIAA, Reston, WA, 1981.

[27] S. Li, L. Petzold, and Y. Ren, *Stability of moving mesh systems of partial differential equations*, SIAM J. Sci. Comput., 20 (1998), pp. 719–738.

[28] F. Liu, S. Ji, and G. Liao, *An adaptive grid method and its application to steady Euler flow calculations*, SIAM J. Sci. Comput., 20 (1998), pp. 811–825.

[29] P. Lötstedt, S. Söderberg, A. Ramage, and L. Hemmingsson-Frändén, *Implicit solution of hyperbolic equations with space-time adaptivity*, BIT, 42 (2002), pp. 134–158.

[30] S. Muzaferija and D. Gosman, *Finite-volume CFD procedure and adaptive error control strategy for grids of arbitrary topology*, J. Comput. Phys., 138 (1997), pp. 776–787.

[31] W. L. Oberkampf and F. G. Blottner, *Issues in computational fluid dynamics code verification and validation*, AIAA J., 36 (1998), pp. 687–695.

[32] S. Osher and F. Solomon, *Upwind schemes for hyperbolic systems of conservation laws*, Math. Comp., 38 (1982), pp. 339–377.

[33] N. G. Pantelelis and A. E. Kanarachos, *The parallel block adaptive multigrid method for the implicit solution of the Euler equations*, Internat. J. Numer. Methods Fluids, 22 (1996), pp. 411–428.

[34] A. Papastavrou and R. Verfürth, *A posteriori error estimators for stationary convection-diffusion problems: A computational comparison*, Comput. Methods Appl. Mech. Engrg., 189 (2000), pp. 449–462.

[35] K. G. Powell, *A tree-based adaptive scheme for solution of the equations of gas dynamics and magnetohydrodynamics*, Appl. Numer. Math., 14 (1994), pp. 327–352.

[36] K. G. Powell, P. L. Roe, T. J. Linde, T. I. Gombosi, and D. L. De Zeeuw, *A solution-adaptive upwind scheme for ideal magnetohydrodynamics*, J. Comput. Phys., 154 (1999), pp. 284–309.

[37] M. Renardy and R. C. Rogers, *An Introduction to Partial Differential Equations*, Springer-Verlag, New York, 1993.

[38] P. Roache, *Quantification of uncertainty in computational fluid dynamics*, Annu. Rev. Fluid Mech., 29 (1997), pp. 123–160.

[39] G. Söderlind, *The automatic control of numerical integration*, CWI Quarterly, 11 (1998), pp. 55–74.

[40] K. Srinivasan and S. G. Rubin, *Solution-based grid adaptation through segmented multigrid domain decomposition*, J. Comput. Phys., 136 (1997), pp. 467–493.

[41] D. A. Venditti and D. L. Darmofal, *Adjoint error estimation and grid adaptation for functional outputs: Application to quasi-one-dimensional flow*, J. Comput. Phys., 164 (2000), pp. 204–227.

[42] J. Wu, H. Ritzdorf, K. Oosterlee, B. Steckel, and A. Schüller, *Adaptive parallel multigrid solution of 2D incompressible Navier-Stokes equations*, Internat. J. Numer. Methods Fluids, 24 (1997), pp. 875–892.

[43] X. D. Zhang, J.-Y. Trépanier, and R. Camarero, *A posteriori error estimation for finite-volume solutions of hyperbolic conservation laws*, Comput. Methods Appl. Mech. Engrg., 185 (2000), pp. 1–19.

SIAM J. SCI. COMPUT.
Vol. 23, No. 5, pp. 1799–1816

MINIMIZING THE PROFILE OF A SYMMETRIC MATRIX*

WILLIAM W. HAGER†

Abstract. New approaches are developed for minimizing the profile of a sparse, symmetric matrix. The heuristic approaches seek to minimize the profile growth, either absolutely or in a weighted sense. The exchange methods make a series of permutations in an initial ordering to strictly improve the profile. Comparisons with the spectral algorithm, a level structure method, and a wave front method are presented.

Key words. matrix envelope, matrix profile, matrix ordering, sparse matrices, Cuthill–McKee algorithm, spectral ordering, wave front ordering, exchange method, weighted greed

AMS subject classifications. 65F05, 65F50

PII. S1064827500379215

1. Introduction.

The envelope of an n by n symmetric matrix $\mathbf{A}$ is the set of index pairs that lie between the first nonzero element in each row and the diagonal:

$$\mathrm{Env}(\mathbf{A}) = \{(i,j) : 1 \le i \le n,\ f_i(\mathbf{A}) \le j < i\},$$

where

$$f_i(\mathbf{A}) = \min\{j : 1 \le j \le i, \quad \text{with } a_{ij} \ne 0\}. \tag{1.1}$$

We assume for convenience that the diagonal elements of $\mathbf{A}$ do not vanish. The profile of a matrix is the number of elements in the envelope plus the number of elements on the diagonal.[1] The **LU** factorization of a symmetric, positive definite matrix can be performed within the space associated with the profile.

When solving sparse, symmetric, positive definite linear systems, we often store only the nonzeros. One of the standard storage structures consists of a pointer from the column to arrays holding the row indices and numerical values of the nonzero entries of that column. In contrast, if the matrix is mostly nonzero, the entire n by n matrix is often stored, both zero and nonzero entries. This storage structure eliminates the integer row indices and pointer arrays, but it requires the storage of zero entries as well as nonzeros. Profile storage is useful when the matrix is moderately sparse, or when the nonzero entries are near the main diagonal. In this approach, we only store the location of the first nonzero in each row, along with the numerical values of entries in the row between the first nonzero and the main diagonal. Profile reducing algorithms, like those presented in this paper, try to minimize the profile by making a symmetric permutation of rows and columns.

The graph G associated with $\mathbf{A}$ has vertices

$$\mathcal{V} = \{1, 2, \ldots, n\}$$

and edges

$$\mathcal{E} = \{(i,j) : i \ne j,\ a_{ij} \ne 0\}.$$

*Received by the editors October 9, 2000; accepted for publication (in revised form) May 28, 2001; published electronically February 6, 2002. This work was supported by the National Science Foundation under grant 9704912.

http://www.siam.org/journals/sisc/23-5/37921.html

†Department of Mathematics, University of Florida, Gainesville, FL 32611 (hager@math.ufl.edu, http://www.math.ufl.edu/∼hager).

[1] Some authors do not include the diagonal elements in the profile.

Here the edges are unordered pairs. The first strategies for reducing envelope size were based on graph level structures. A level structure is a partition of the vertices into disjoint sets L_1, L_2, ..., L_k with the property that all vertices adjacent to vertices in level L_i are in either L_{i-1} or L_i or L_{i+1}. The first scheme exploiting a level structure to reorder the rows and columns of a matrix and to minimize envelope size was the algorithm of Cuthill and McKee [3]. Later Alan George observed that if the Cuthill–McKee ordering was reversed, then the size of the envelope was often reduced. Liu and Sherman [24] show that this reversal can never increase the size of the envelope. Related level structure methods are developed by Gibbs, Poole, and Stockmeyer [9] and Gibbs and King [23] for which the envelope size is often comparable, but the computation time is significantly less.

Another approach for minimizing the profile is Sloan's method [27], in which the rows and columns of a matrix are ordered in accordance with a priority function; this function assigns to each of the presently unordered nodes a value that is a linear combination of the distance to an "end node" and the associated change in the wave front. In each step of the algorithm, the node with the highest priority is added to the list of labeled nodes. Further improvements to this scheme are developed by Duff, Reid, and Scott [5].

The spectral approach of Barnard, Pothen, and Simon [1] uses an eigenvector (Fiedler vector) corresponding to the second smallest eigenvalue of the graph's Laplacian to minimize the profile of a matrix. The matrix ordering corresponds to the permutation of the components of the Fiedler vector which gives either increasing or decreasing order. George and Pothen [8] provide analysis justifying this strategy. More recently Kumfert and Pothen [22] have developed a hybrid scheme that uses the spectral algorithm to obtain a preordering that is further refined using a modification of Sloan's algorithm. In their numerical experiments, this hybrid algorithm generated profiles 2 to 5 times smaller than those of the reverse Cuthill–McKee algorithm. This was developed further by Reid and Scott [26] and implemented in the HSL [16] package MC60 (HSL was formerly known as the Harwell Subroutine Library).

Recently, Boman and Hendrickson [2] and Hu and Scott [17] developed multilevel algorithms for profile optimization, analogous to the multilevel algorithms that have proved so successful in graph partitioning [13, 14, 18, 19, 20]. Solving the coarse grid problem using an adjacent exchange approach, Boman and Hendrickson achieved performance comparable to that of the original Sloan algorithm. More recently, Hu and Scott achieved performance comparable to that of the hybrid Sloan algorithm by using Sloan's algorithm itself to solve the coarse grid problem. These multilevel methods are faster than the hybrid method since they do not involve computing the Fiedler vector.

In this paper, we develop two classes of methods for optimizing the profile of a matrix. The first class of methods are heuristics based on the following observation: The envelope of a matrix contains all the (off-diagonal) nonzeros of a matrix, and if $\mathbf{P}$ is a permutation matrix, then the number of nonzeros in $\mathbf{A}$ and in $\mathbf{PAP}^{\mathsf{T}}$ are identical. Hence, finding $\mathbf{P}$ to minimize the profile is equivalent to finding $\mathbf{P}$ to minimize the number of zeros in the envelope. Our algorithm starts in the lower right corner of the matrix and builds up $\mathbf{P}$, minimizing the growth of zeros in the envelope in each step. We consider two variations of this approach: "pure greed," in which the growth of zeros is minimized in each step, and "weighted greed," in which a weighted measure of the growth in zeros is minimized.

Heuristics like these, as well as those mentioned earlier, usually improve the profile; however, improvement is not guaranteed. The other class of methods that we develop involve exchanges of adjacent rows and adjacent columns to strictly improve the profile. Boman and Hendrickson also use adjacent exchanges, along with the techniques of Kernighan and Lin [21] and Fiduccia and Mattheyses [6], to solve their coarse grid problem. Here we consider a series of adjacent exchanges—even though individual exchanges do not improve the profile, a series of exchanges may lead to an improvement. Formulas are derived for the profile change associated with the entire series of exchanges. From the structure of these formulas, we obtain a subset of the possible exchanges that could yield an improvement. These exchange methods can be applied directly to the starting matrix to improve its profile, or they can be applied to the ordering generated by any of the heuristics to further improve the profile. In fact, they could be combined with either of the hybrid methods [22, 26] to obtain a 3-method hybrid, whose profile is at least as good as that of the 2-method hybrid. Numerical comparisons between the ordering strategies are given in section 4 using matrices that arise in linear programming (netlib/LP) and matrices from the Harwell-Boeing and NASA directories of Timothy Davis's University of Florida sparse matrix collection at www.cise.ufl.edu/research/sparse/matrices.

2. The new heuristics.

Let $\mathbf{P}$ be a permutation matrix and let $\mathbf{B} = \mathbf{PAP}^{\mathsf{T}}$ be the permutation of $\mathbf{A}$. This permutation $\mathbf{A}$ can be described by a vector $\mathbf{p} \in \mathbf{R}^n$ where p_i is the row of $\mathbf{A}$ corresponding to row i in the permuted matrix $\mathbf{B}$. In our profile heuristic, we assign values to p_i, starting with $i = n$, and working toward $i = 1$. At step k the associated set of "labeled vertices" is defined by

$$\mathcal{L}_k = \{p_i : k < i \le n\},$$

where the initial set $\mathcal{L}_n$ is the empty set. The complement of $\mathcal{L}_k$, denoted $\mathcal{U}_k$, is the set of "unlabeled vertices." The "wave front" $\mathcal{W}_k$ relative to $\mathcal{L}_k$ and $\mathcal{U}_k$ is defined by

$$\mathcal{W}_k = \{i \in \mathcal{L}_k : a_{ij} \neq 0 \text{ for some } j \in \mathcal{U}_k\}.$$

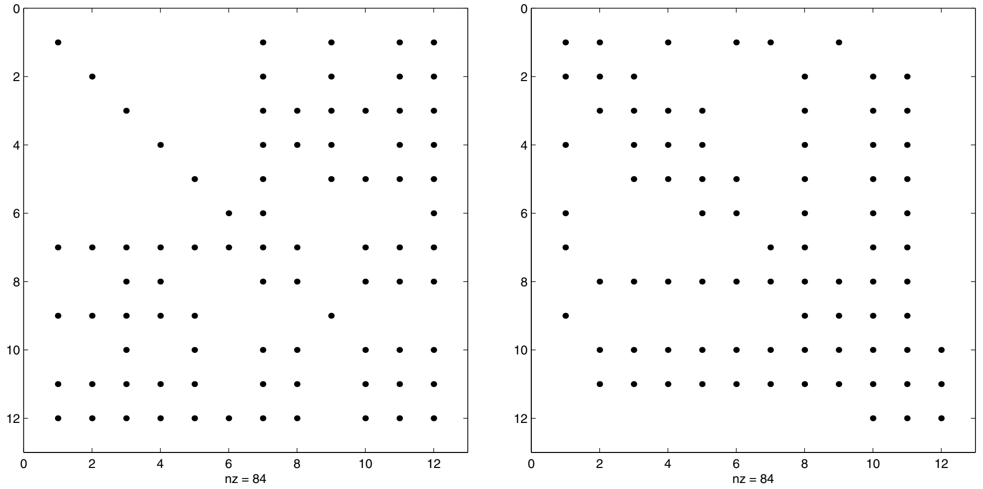
If $\mathbf{A}$ is permuted symmetrically so that the leading columns correspond to $\mathcal{U}_k$ and the trailing columns correspond to $\mathcal{L}_k$, then the wave front is the set of trailing rows where there will be nonzero multipliers in the leading columns when the lower left corner of the permuted matrix is eliminated.

Now let us consider the effect of making the assignment $p_k = j$ at step k for some $j \in \mathcal{U}_k$. The row indices of those elements a_{ij} that both vanish and lie in the envelope of the permuted matrix are given by

$$\mathcal{Z}_k(j) = \{i \in \mathcal{W}_k : a_{ij} = 0\}.$$

As noted earlier, the size of the profile is minimized when the number of zeros in the profile is minimized. A purely greedy labeling approach would simply choose j to minimize the size of $\mathcal{Z}_k(j)$, the number of new zeros that enter the envelope at step k. In other words, if $|\mathcal{Z}_k(j)|$ denotes the number of elements in $\mathcal{Z}_k(j)$, the purely greedy approach chooses $p_k = j \in \mathcal{U}_k$ where

$$|\mathcal{Z}_k(j)| = \min_{l \in \mathcal{U}_k} |\mathcal{Z}_k(l)|. \tag{2.1}$$



A. Original matrix

B. Matrix permuted by Algorithm 1

FIG. 1. *Illustration of Algorithm* 1.

A possible choice for the starting node p_n could be a node of minimum degree in the graph of the matrix. Letting d_j denote the degree of node j, this greedy labeling strategy is the following.

ALGORITHM 1 (greedy minimum profile growth).

```
$J = arg \min_{1 \le j \le n} d_j$
$p_n = j$ for any $j \in J$
for $k = n-1, n-2, \ldots, 1$
    $J = arg \min_{j \in \mathcal{U}_k} |\mathcal{Z}_k(j)|$
    $p_k = j$ for any $j \in J$
end
```
end Algorithm 1

Here the notation "arg min" denotes the set of indices which attain the minimum, while $\mathcal{U}_k$ is the set of unlabeled vertices at step k.

Figure 1 illustrates one of the deficiences with this purely greedy strategy. The last column of Figure 1B corresponds to column 6 in the original matrix and the node of minimum degree. The next four columns (numbers 11 down to 8) in Figure 1B correspond to columns 7, 12, 1, and 11 in Figure 1A. These columns are attractive to the greedy algorithm since they create no zeros in the profile in the lower right 5×5 corner of the matrix. Now, skipping to the end, the first column in Figure 1B corresponds to column 9 in Figure 1A. Notice that when putting column 9 of Figure 1A in column 1 of Figure 1B, we end up with many zeros within the profile (in rows 4, 6, 7, and 9). In this greedy approach, column 9 of Figure 1A was left to the end for the following reason: If we were to assign column 9 earlier in Figure 1B, zeros would be created in the profile in rows 11, 10, and 8. Thus Algorithm 1 does not take into account the fact that these columns are already mostly nonzero, and there is only room for 3 more zeros. As we delay the assignment of column 9 in Figure 1A, we create, in each step, more zeros in the profile in Figure 1B.

To rectify this problem, we can weight our score for each row to take account of the number of nonzeros that remain in a column (or row). The weight function should be chosen so that a zero placed in a column with a lot of nonzeros is less significant than a zero placed in a column with a small number of nonzeros. Let χ be the indicator function defined by

$$\chi(a) = 1 \text{ if } a \neq 0, \quad \chi(a) = 0 \text{ if } a = 0,$$

and for $i \in \mathcal{L}_k$, let $r_k(i)$ be the number of nonzeros associated with unlabeled nodes in row i:

$$r_k(i) = \sum_{j \in \mathcal{U}_k} \chi(a_{ij}). \tag{2.2}$$

In the purely greedy labeling strategy, we assign to each $j \in \mathcal{U}_k$ a "score" $s_k(j)$ that is simply the number of elements in the set $\mathcal{Z}_k(j)$. On the other hand, if W is the weight function, then the weighted score, based on the row sums $r_k(i)$, is

$$s_k(j) = \sum_{i \in \mathcal{Z}_k(j)} W(r_k(i)). \tag{2.3}$$

If W is identically one, then $s_k(j) = |\mathcal{Z}_k(j)|$ and Algorithm 1 corresponds to minimizing $s_k(j)$ in each step. But as illustrated in Figure 1, it is better to choose W so that $W(t)$ is small when t is large. For example, we could take $W(t) = 1/t$. In this way, zeros created in rows with lots of nonzeros receive a small weight in the score for node j. The best node to label at step k is the node $j \in \mathcal{U}_k$ with the lowest score $s_k(j)$.

The formula (2.3) involves a sum over indices associated with zero entries in $\mathbf{A}$. Since sparse matrices have many zeros, it would be better to work with an analogous score expressed in terms of nonzeros. Let $\mathcal{N}_k(j)$ be the nonzeros associated with node j and the wave front at step k:

$$\mathcal{N}_k(j) = \{i \in \mathcal{W}_k : a_{ij} \neq 0\}.$$

The score of node j based on nonzeros is

$$S_k(j) = \sum_{i \in \mathcal{N}_k(j)} W(r_k(i)) = \left(\sum_{i \in \mathcal{W}_k} W(r_k(i)) \right) - s_k(j). \tag{2.4}$$

Again, the weight function W should be monotone decreasing since rows with a lot of nonzeros are less significant than rows with a few nonzeros. Since the scores S_k and s_k are complementary, as indicated in (2.4), minimizing s_k is equivalent to maximizing S_k. Hence, when working with nonzeros, the biggest score is best, and we should choose j to maximize $S_k(j)$ in each step. Again, if W is identically one, then maximizing $S_k(j)$ over $j \in \mathcal{U}_k$ is equivalent to minimizing $s_k(j)$ over $j \in \mathcal{U}_k$, which is the greedy heuristic given in Algorithm 1.

The score $S_k(j)$ takes into account the effect of edges between an unlabeled node j and the labeled nodes $\mathcal{L}_k$. It does not take into account the effect of edges between j and the other unlabeled nodes. If an unlabeled node j is connected by a small number of edges to unlabeled nodes, it should be scored favorably since it is relatively easy to prevent the growth of zeros in the part of the profile connected with this row.

Likewise, as the number of edges between j and the other nodes in $\mathcal{U}_k$ approaches k, this node should be scored favorably since there is little room left for zeros in its row. Our final weighted score $\bar{S}_k(j)$, taking into account both edges between j and $\mathcal{L}_k$ as well as edges between j and $\mathcal{U}_k$, is the following:

$$\bar{S}_k(j) = \max\{W(d_k(j)+1), W(k - d_k(j))\} + \sum_{i \in \mathcal{N}_k(j)} W(r_k(i)), \tag{2.5}$$

where $d_k(j)$ is the degree of j in the subgraph associated with $\mathcal{U}_k$. The first term in (2.5) has the following interpretation: It is the score associated with row k if j is labeled at step k.

The first term in (2.5) leads to a natural starting procedure. For $k = n$, the summation in (2.5) disappears and $\bar{S}_n(j)$ is maximized by either a node of lowest degree or a node of highest degree. Since the best place for completely dense (nonzero) rows and columns is the trailing border of the matrix, this starting procedure treats dense rows and columns correctly. For completeness, we state the weighted greed scheme.

Algorithm 2 (weighted greed).

```
    for $k = n, n-1, \ldots, 1$
        $J = arg \max_{j \in \mathcal{U}_k} \bar{S}_k(j)$
        $p_k = j$ for any $j \in J$
    end
end Algorithm 2
```

If the weight function is strictly monotone, then evaluation of scores could be computationally expensive, with a work estimate comparable to that of an **LU** factorization itself. By choosing W to be piecewise constant, the scoring process is sped up since the score for any node will change only when one of its associated $r_k(i)$, $i \in \mathcal{N}(j)$, moves across a step in W. In the numerical experiments reported later, we used a weight function of the following form:

$$W(i) = 1/i \text{ for } 1 \le i \le \psi, \quad W(i) = 1/\psi \text{ for } i > \psi, \tag{2.6}$$

where ψ is a small positive integer which we call the cutoff. In the experiments, $\psi = 8$. For the matrix of Figure 1A, the profile generated by Algorithm 2 is 54, while the profile of Figure 1B and Algorithm 1 is 63. The notation nnz, used below, stands for "number of nonzeros."

Lemma 2.1. *For the weight function* (2.6)*, Algorithm* 2 *can be implemented in time bounded by* $O(\text{nnz}(\mathbf{A}) \log n)$*.*

Proof. In each step of Algorithm 2, $\mathbf{r}_k$, the vector whose ith component is $r_k(i)$, is adjusted according to the location of nonzeros in column p_k of $\mathbf{A}$. Hence, the number of components of $\mathbf{r}_k$ and $\mathbf{r}_{k-1}$ that are different is bounded by the number of nonzeros in column p_k of $\mathbf{A}$. It follows that the total number of changes in components of $\mathbf{r}_k$ as k ranges between n and 1 is bounded by $\text{nnz}(\mathbf{A})$. If $W(r_k(i)) \ne W(r_{k-1}(i))$ for some i, then $r_k(i) \le \psi$ according to (2.6). For each $i \in \mathcal{L}_k$, $r_k(i)$ can only decrease as k decreases since the set $\mathcal{U}_k$ in (2.2) decreases in size as k decreases from n down to 1. Hence, for any given i and for the weight function (2.6), there are at most ψ values of k for which $W(r_k(i))$ changes. Now consider the last term in (2.5), which we need to evaluate for each $j \in \mathcal{U}_k$. We can think of this sum in the following way: For each $i \in \mathcal{L}_k$ associated with a nonzero a_{ij}, $j \in \mathcal{U}_k$, let us replace a_{ij} by $W(r_k(i))$, and let $\mathbf{A}_k$ be the resulting matrix whose remaining elements are all zero. Forming the sum in (2.5) for each $j \in \mathcal{U}_k$ is equivalent to computing the column sums of $\mathbf{A}_k$; the only nonzero entries are in a submatrix in the lower left corner of $\mathbf{A}_k$. As k decreases by one, we delete a column and add a row to the submatrix. If the numbers inside the submatrix were fixed, independent of k, then the work associated with adding a row, deleting a column, and updating the column sums is bounded by the number of nonzeros in $\mathbf{A}$. Note though that not only is the submatrix changing (add a row, delete a column) in each step, but the elements inside the submatrix change too. However, for each choice of i, we saw earlier that there are at most ψ values of k for which $W(r_k(i))$ changes its value. Hence, the total number of changes in value of the elements in the submatrix is bounded by $\psi\text{nnz}(\mathbf{A})$, and the total work in maintaining the sum in (2.5) remains $\text{O}(\text{nnz}(\mathbf{A}))$.

In each step of Algorithm 2, $\mathbf{d}_k$, the vector whose entries are the degrees of the nodes in the subgraph associated with $\mathcal{U}_k$, is adjusted according to the location of nonzeros in column p_{k+1} of $\mathbf{A}$: For each $i \in \mathcal{U}_k$, we have

$$d_k(i) = d_{k+1}(i) - 1$$

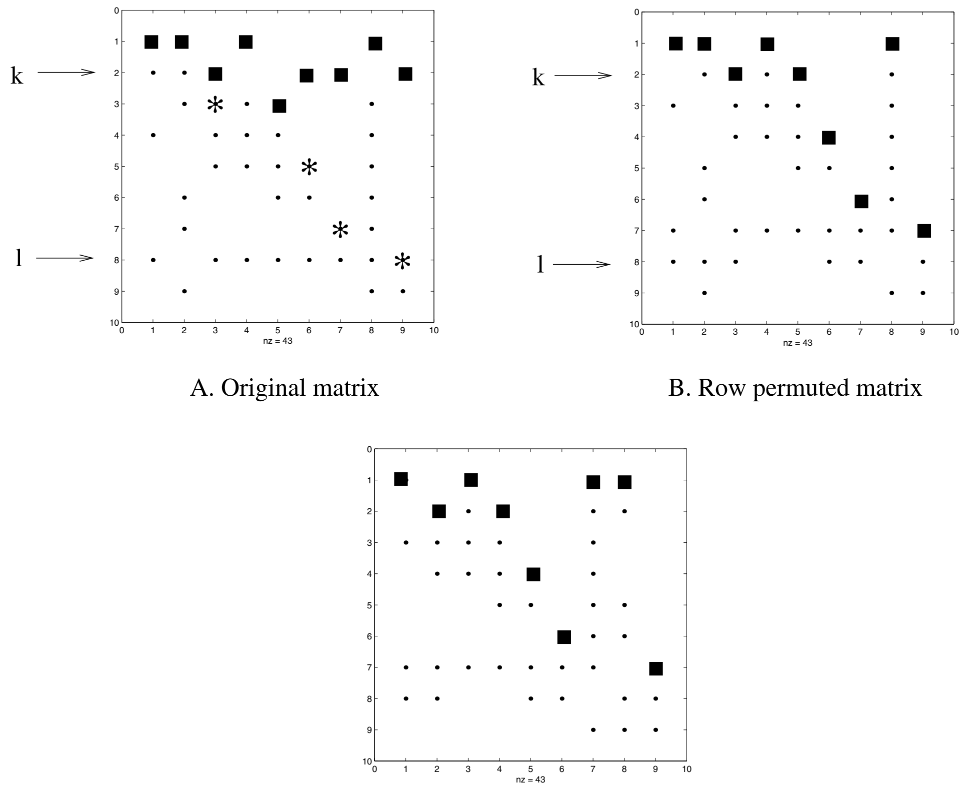
if $a_{ij} \neq 0$ for $j = p_{k+1}$. As with $\mathbf{r}_k$, the total number of changes in components of $\mathbf{d}_k$ as k ranges between n and 1 is bounded by $\text{nnz}(\mathbf{A})$. In any step k where $d_k(j) < d_{k+1}(j)$ for some $j \in \mathcal{U}_k$, we need to check whether the max in (2.5) has increased. Since the total number of changes in components of $\mathbf{d}_k$ is bounded by $\text{nnz}(\mathbf{A})$, the time needed to check whether the max in (2.5) increases when $d_k(j)$ decreases is $\text{O}(\text{nnz}(\mathbf{A}))$.

Now consider the second term in the max of (2.5). Assume that each $j \in \mathcal{U}_k$ is stored in a list, ordered by degree. This degree-ordered list can be created and maintained in time bounded by $\text{O}(\text{nnz}(\mathbf{A}))$ since the number of changes in degree is bounded by $\text{O}(\text{nnz}(\mathbf{A}))$. Since $k - d_k(j)$ decreases monotonically as k decreases, there are at most ψ values of k where $W(k - d_k(j))$ changes in value. Hence, the total number of pairs (j, k) for which $W(k - d_k(j))$ changes in value is at most ψn. Since the diagonal of $\mathbf{A}$ does not vanish, $\psi n = \text{O}(\text{nnz}(\mathbf{A}))$. Consequently, the effort involved with checking whether the max in (2.5) increases due to an increase in $W(k - d_k(j))$ is $\text{O}(\text{nnz}(\mathbf{A}))$.

In each step of Algorithm 2, we extract an unlabeled node of minimum score. If the nodes are ordered in a heap according to their score, we can update the heap when a node's score changes in time bounded by $\text{O}(\log k) \leq \text{O}(\log n)$. Since the number of terms that change in the scores (2.5) is at most $\text{O}(\text{nnz}(\mathbf{A}))$, the total work in maintaining the heap is bounded by the product $\text{O}(\text{nnz}(\mathbf{A}) \log n)$. □

3. Exchange methods. By an exchange method, we mean any strategy for improving the profile of a matrix based on a series of symmetric interchanges of rows and of columns. One strategy to improve the profile of a matrix would be to consider all possible pairs of rows, and corresponding columns, and make an exchange if the profile is reduced. In this section, however, we focus on more specialized exchanges.

For any n by n matrix $\mathbf{A}$ and $k < l \leq n$, let $D_{k:l}(\mathbf{A})$ denote the change in the profile associated with a series of adjacent exchanges: Interchange row k with $k + 1$, row $k + 1$ with $k + 2$, ..., row $l - 1$ with l, and perform the symmetric interchange of columns. Before stating a formula for $D_{k:l}(\mathbf{A})$, we give an illustration. Consider the 9 by 9 matrix depicted in Figure 2, and the values $k = 2$ and $l = 8$. Throughout this section, our figures are based on the "skyline view" of matrix profile—in other words, elements in each column between the first nonzero and the diagonal. The dots denote nonzero entries and a solid square is placed over each nonzero entry corresponding to a frontier node, a node on the top boundary of the envelope. The first nonzero beneath



A. Original matrix

B. Row permuted matrix

C. Row and column permuted matrix

FIG. 2. *Downward exchanges in a symmetric matrix.*

a frontier node is covered with an asterisk. When row k is exchanged with row $k+1$, row $k+1$ is exchanged with $k+2$, and so on, down to row l, we obtain Figure 2B. The single frontier node beneath row k is moved one level higher, increasing the profile by 1. The frontier nodes in row k, on the other hand, fall to positions right above the asterisks in Figure 2A, reducing the profile by 11. Altogether, the profile is reduced by 10. To complete the permutation, we perform the corresponding exchange of the columns. This second set of permutations restores symmetry and moves the columns around, but it does not affect the height of frontier nodes—there are still 4 frontier nodes in row 1, 2 in row 2, and so on. Since the number of zeros above the frontier nodes does not change, the profile change deduced in Figure 2B is correct.

Our formula for $D_{k:l}$ is the following.

LEMMA 3.1. *For any $k < l \leq n$ and for f defined in* (1.1), *we have*

$$D_{k:l}(\mathbf{A}) = |\{j \geq k : k < f_j(\mathbf{A}) \leq l\}| - \left(\sum_{j \in \mathcal{F}_k} (\min\{l, g_j(\mathbf{A}) - 1\} - k) \right), \tag{3.1}$$

where

$$\mathcal{F}_k = \{j : f_j(\mathbf{A}) = k\}, \tag{3.2}$$

and

$$g_i(\mathbf{A}) = \min\{j : f_i(\mathbf{A}) < j,\ a_{ij} \neq 0\} \tag{3.3}$$

if the minimum exits. Otherwise, $g_i(\mathbf{A}) = n + 1$.

The elements of the frontier set $\mathcal{F}_k$, the dark squares in Figure 2, are nodes that are adjacent to k and on the boundary of the envelope. The value of g_i is the second nonzero column index in row i, if it exists. The elements of $\mathbf{A}$ corresponding to the g_i and $i \in \mathcal{F}_k$ are the asterisks in Figure 2. Note that $g_i(\mathbf{A})$ could be larger than i, while $f_i(\mathbf{A}) \le i$.

Proof. The first set in (3.1) represents those zeros that are added to the profile as we successively exchange the rows between k and l. The summation in (3.1) corresponds to the zeros removed from the profile as the rows are exchanged and the frontier moves towards the diagonal of the matrix. Once we reach the second nonzero in a row, the frontier in that row remains stationary. To complete the permutation, we perform the corresponding exchange of the columns. This second set of permutations restores symmetry and moves the columns around, but it does not affect the number of zeros above each frontier node. Since the profile of the matrix equals the number of elements in the upper triangle minus the number of zeros above the frontier nodes, this second set of permutations does not affect the profile (since the number of zeros above the frontier nodes does not change). □

Suppose that we wish to determine whether a series of downward exchanges can be performed that would reduce the profile of a symmetric matrix. Referring to (3.1), we see that as l increases, the first term increases while the second term decreases, achieving its minimum value when l reaches the maximum of the $(g_j(\mathbf{A}) - 1)$ over $j \in \mathcal{F}_k$. When $l \ge g_j(\mathbf{A}) - 1$ for all $j \in \mathcal{F}_k$, the second term in (3.1) remains constant. Hence, when determining the best value for l, we should restrict our attention to

$$k < l < l_1 := \max_{j \in \mathcal{F}_k} g_j(\mathbf{A}).$$

Also note that for $l > k$, we have the bound

$$\sum_{j \in \mathcal{F}_k} (\min\{l, g_j(\mathbf{A}) - 1\} - k) \le \sum_{j \in \mathcal{F}_k} (g_j(\mathbf{A}) - k - 1).$$

The first term in (3.1) is a monotone increasing function of l, so when minimizing $D_{k:l}(\mathbf{A})$ over $l > k$, we should require that $l \le l_2$ where l_2 is the largest l with the property that

$$|\{j \ge k : k < f_j(\mathbf{A}) \le l\}| \le m_1 := \sum_{j \in \mathcal{F}_k} (g_j(\mathbf{A}) - k - 1).$$

Lemma 3.1 can be utilized in an exchange scheme to improve a profile in the following way: For each $k = n-1, n-2, \ldots, 1$, we could check whether $D_{k:l}(\mathbf{A}) < 0$ for some $l > k$. If L is a value l that minimizes $D_{k:l}(\mathbf{A})$ over $l > k$, and if $D_{k:L}(\mathbf{A}) < 0$, we should exchange the rows between k and L. In searching for a value of l that minimizes $D_{k:l}(\mathbf{A})$, the constraints $l \le l_1$ and $l \le l_2$ should be exploited to reduce the search space. In the following summary of the downward exchange scheme, we let $\mathbf{P}$ denote the permutation matrix associated with the permutation vector $\mathbf{p}$.

Algorithm 3 (down exchanges).

```
    for $k = n-1, n-2, \dots, 1$
        $L = arg \min_{k<l\le n} D_{k:l}(\mathbf{PAP}^{\mathsf{T}})$
        if $D_{k:l}(\mathbf{A}) < 0$ for some $l \in L$
            $q = p_k$;   $p_{k:l-1} = p_{k+1:l}$;   $p_l = q$;
        end if
    end
end Algorithm 3
```

Algorithm 3 can be performed in concert with Algorithm 2. That is, right after making the assignment $p_k = j$ in Algorithm 2, we could evaluate the L of Algorithm 2 and make a downward exchange if it is beneficial. This combined mode works since downward exchanges in the trailing part of the matrix do not change the scores of unlabeled nodes. This combined mode avoids a second pass over the matrix. For the experiments reported in section 4, we implement this combined mode; however, the improvement in the profile connected with the down exchanges is accumulated and subtracted from the total profile. This allows us to run the code in this more efficient coupled mode, while tabulating separately the profile improvement due to the exchanges.

Similar to the downward exchanges, we could perform upward exchanges to improve the profile. For any matrix $\mathbf{A}$ and $k > l$, let $D_{k:l}(\mathbf{A})$ denote the change in the profile associated with the interchange of rows k and $k-1$, rows $k-1$ and $k-2, \dots,$ rows $l-1$ and l, and the symmetric interchange of columns. The effect of upward exchanges is quite different from that of downward exchanges. As an illustration, consider the 7 by 7 symmetric matrix in Figure 3. Again, a solid black square is placed over each nonzero corresponding to a frontier node. When row 7 is exchanged successively with the rows above it, the frontier node in column 7 is moved up to row 1. This increases the profile by 3. At the same time, the frontier nodes in columns 1, 2, 3, and 6, which lie above zeros in row 7, are lowered by one row. The net decrease in the profile is 1. When the corresponding column interchanges are applied to the matrix of Figure 3B, symmetry is restored. Since these column exchanges do not affect the number of frontier nodes in each row, the change in profile deduced in Figure 3B is correct.

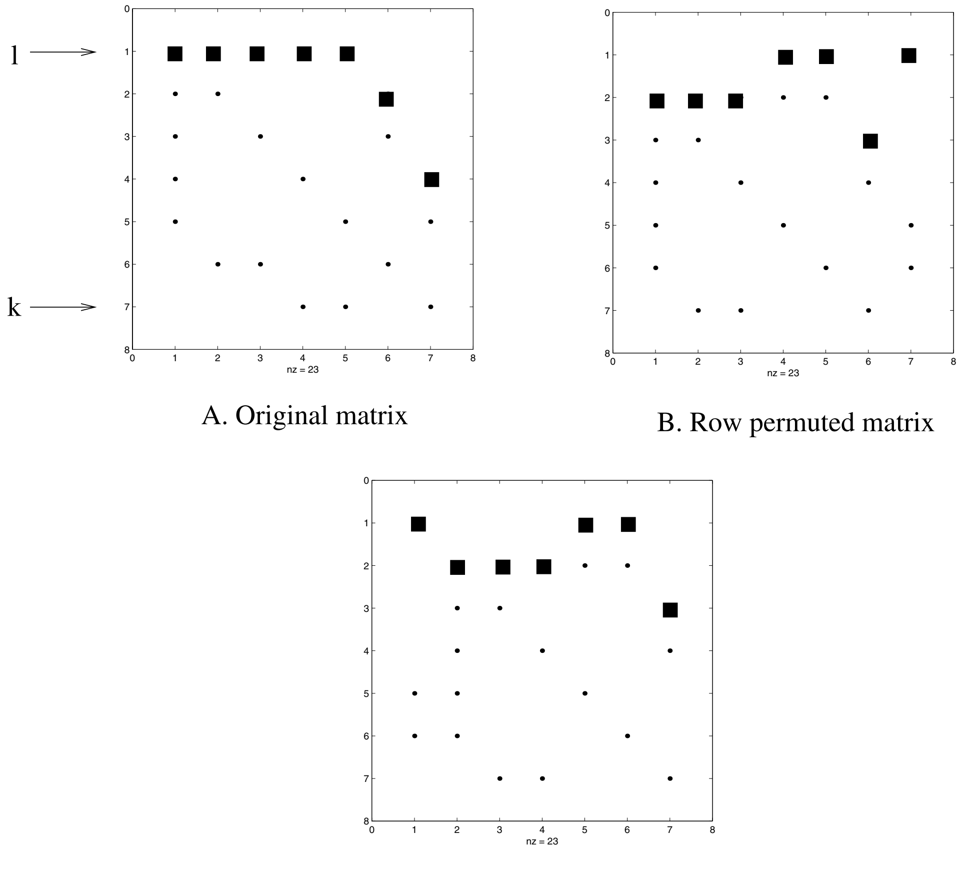
Lemma 3.2. *For any $l < k \le n$ and for f defined in* (1.1)*, we have*

$$D_{k:l}(\mathbf{A}) = \left(\sum_{\{j : a_{kj} \ne 0\}} \max\{f_j(\mathbf{A}) - l, 0\} \right) - |\{j : a_{kj} = 0,\ k > f_j(\mathbf{A}) \ge l\}|. \tag{3.4}$$

Proof. As row k is exchanged successively with rows above, the zeros in row k cause the frontier nodes above them to drop one row lower. The second term in (3.4) reflects this effect. On the other hand, as row k is exchanged upward, nonzeros in row k eventually reach the frontier, and each subsequent exchange increases the number of zeros in the profile by one. The first term in (3.4) reflects the effect of these nonzeros in row k. To complete the permutation, we perform the corresponding exchange of the columns. These column exchanges restore symmetry but do not affect the number of zeros above each frontier node. Hence, this second set of permutations does not affect the profile. □

Let l_3 be the largest value of $l < k$ for which

$$\sum_{a_{kj} \ne 0} \max\{f_j(\mathbf{A}) - l, 0\} \ge m_2 := |\{j : k > f_j(\mathbf{A}) \ge 1\}|.$$



A. Original matrix

B. Row permuted matrix

C. Row and column permuted matrix

FIG. 3. *Upward exchanges in a symmetric matrix.*

Since the first term in (3.4) increases as l decreases, and since the second term in (3.4) is no bigger than m_2, we conclude that $D_{k:l}(\mathbf{A}) \geq 0$ for all $l \leq l_3$. Hence, when searching for the minimum of $D_{k:l}(\mathbf{A})$ over $l < k$, we should restrict our attention to $l > l_3$.

By (3.4) $D_{k:l}(\mathbf{A})$ is a piecewise linear function of l whose minimum is attained either at $l = k$ or $l = f_j(\mathbf{A})$ for some j. Consequently, to achieve rapid evaluation of the minimum of $D_{k:l}(\mathbf{A})$, we should record for each value of m, the sizes of the level sets

$$\mathcal{F}_m = \{j : f_j(\mathbf{A}) = m\},$$

the same set appearing in Lemma 3.1, and those m for which $\mathcal{F}_m$ are nonempty should be ordered in a linked list. The minimum of $D_{k:l}(\mathbf{A})$ is achieved, either at $l = k$ or at one of the m in this list.

Lemma 3.2 can be utilized in an exchange scheme in much the same way that we utilize Lemma 3.1 in Algorithm 3. However, instead of starting at the bottom of the matrix and exchanging with rows below, we start at the top of the matrix and exchange with rows above. In the following summary of the upward exchange scheme, we again let $\mathbf{P}$ denote the permutation matrix associated with the permutation vector $\mathbf{p}$.

```
ALGORITHM 4 (up exchanges).
    for $k = 2, 3, \ldots, n$
        $L = arg\ \min_{1 \le l < k}\ D_{k:l}(\mathbf{PAP}^{\mathsf{T}})$
        if $D_{k:l}(\mathbf{A}) < 0$ for some $l \in L$
            $q = p_l; \quad p_{l:k-1} = p_{l+1:k}; \quad p_k = q;$
        end if
    end
end Algorithm 4
```

The down exchange and up exchange schemes are utilized in a iterative fashion; we apply Algorithm 3, followed by Algorithm 4, followed by Algorithm 3, and so on. In practice, there was no significant improvement after a relatively small number of sweeps, where a sweep denotes a pass through Algorithm 3 followed by a pass through Algorithm 4. In many cases, there was no improvement at all after 3 sweeps, while in some cases, there was a small improvement even after 20 sweeps. In the numerical experiments, we limited the number of sweeps to 5. The execution time of exchange methods can be reduced, while limiting the possible profile improvement, by restricting the extent of the adjacent rows that are exchanged. In our experiments, we did not allow exchanges that went beyond 1000 adjacent row swaps in each exchange step. Another approach to speed up the exchange process, presented in [25], is to use a bisection step to determine the rows to exchange.

4. Numerical experiments.

In this section, we report on a series of numerical experiments using three test sets, all found in Timothy Davis's University of Florida sparse matrix collection, currently located at

www.cise.ufl.edu/research/sparse/matrices

In particular, we use all 109 matrices in the linear programming test set (most of David Gay's linear programming set found originally in Netlib), the 153 symmetric, nondiagonal matrices in the Harwell-Boeing collection [4], and the 8 symmetric matrices in the NASA test set (matrices which arise in structural engineering problems).

The linear programming test set includes m by n matrices $\mathbf{A}$ where m is between 24 and 105,127, with an average value of 3236. The first series of experiments were based on a set of 14,842 matrices, which we call Test Set 1, constructed in the following way: Using the Chaco partitioning code of Hendrickson and Rothberg [15], we generated a permutation matrix $\mathbf{P}$ designed so that $\mathbf{PA}(\mathbf{PA})^{\mathsf{T}}$ has as many nonzeros as possible in diagonal blocks of size roughly k by k where $k = 2\mathrm{nnz}(\mathbf{A})/m$. This construction arises when one uses block iterative techniques to solve linear programming problems (see [10, 11, 12]). The diagonal blocks generated in this way were used for the test matrices. These matrices have an average density (fraction of nonzeros) .45, while their average dimension is 13.1. If the matrices are simply stored as dense symmetric matrices (only store the upper triangle), the storage requirement would be 3,400,729. Using suitable permutations and profile storage, the storage is reduced by a factor of more than 2.

Test Set 2 is obtained in the following way: For each of the 109 rectangular matrices, we form $\mathbf{AA}^{\mathsf{T}}$. A matrix with this nonzero pattern arises in interior point methods as well as in the LP dual active set algorithm. The average density of these matrices is .11, four times smaller than the density of the matrices in Test Set 1. The matrices in Test Set 3, the Harwell-Boeing symmetric matrices, have an average dimension of 2043 and an average density of .03, about four times smaller than the average density of Test Set 2. The matrices in Test Set 4, the NASA symmetric matrices, have an average dimension of 15,750, and an average density of .007, about

Table 1
Profiles, units of 10^6*, for Test Set* 1.

w	wx	m	mx	r	rx	s	sx	x
1.299	1.278	1.303	1.288	1.583	1.313	1.983	1.292	1.272

four times smaller than the average density of Test Set 3. The names of the matrices in Test Sets 2, 3, and 4 and the nonzero patterns for the matrices in Test Set 1 can be found at

www.math.ufl.edu/~hager/papers/profile_testsets

We consider the following five methods:

1. Algorithm 2 (weighted greed).
2. Sloan's method as implemented in Algorithm MC60 from HSL [16] (provided by John Reid).
3. Reverse Cuthill–McKee algorithm (routine symrcm of Matlab).
4. The spectral method (using Matlab's eig or eigs routines).
5. Algorithms 3 and 4 (exchange methods).

MC60 allows for several algorithmic modes; supernodes can be exploited, reverse Cuthill–McKee ordering can be done besides Sloan's method, a spectral method can be combined with Sloan's method by specifying a priority function, and an interface to a frontal solver is provided. In our experiments, we use simply Sloan's method without supernodes (none of the codes in our experiments exploit supernodes). In the fifth method above, we iterate Algorithms 3 and 4 until either there is no further improvement in the profile or until we have done 5 iterations. We initially apply Algorithms 3 and 4 to the given matrix; then we generate a random permutation of the matrix and repeat Algorithms 3 and 4. We report the best profile gotten after 10 random permutations of the given matrix. Since Algorithms 3 and 4 can be used to improve the profile of any given matrix, we also applied them to the permuted matrix generated by each of the first four algorithms. In our experiments, Algorithm 2 nearly always provided a better profile than Algorithm 1; consequently, results for Algorithm 1 are not given.

In Table 1, we give the total profile for the 14,842 matrices in the first test set. The column headings are the following:

w – weighted greed, Algorithm 2,
wx – weighted greed followed by Algorithms 3 and 4, the exchange methods,
m – MC60,
mx – MC60 followed by the exchange methods,
r – reverse Cuthill–McKee,
rx – reverse Cuthill–McKee followed by the exchange method,
s – spectral method,
sx – spectral method followed by the exchange methods,
x – apply the exchange methods to the original matrix and 10 permutations of it.

Observe that the smallest profile was gotten by method x, the exchange methods combined with 10 random permutations. The second best profile was gotten by method wx, the weighted greed/exchange method combination. All the methods received significant benefit from postprocessing using the exchange methods. The biggest beneficiary was the spectral method, for which the profile improved about 35%.

Table 2
Unique superiority for Test Set 1.

wx	mx	rx	sx	x
45	27	19	80	227

Table 3
Profiles for Test Set 2.

Dim	Scale	#	w	wx	m	mx	r	rx	s	sx	x
100	10^3	11	7.1	7.0	7.4	7.2	8.6	7.6	8.8	7.6	6.9
400	10^5	29	2.2	2.1	2.3	2.1	3.4	2.3	2.9	2.1	2.1
1600	10^6	40	2.7	2.5	3.0	2.7	4.7	3.1	2.7	2.5	2.7
6400	10^7	19	2.9	2.8	3.2	3.1	4.4	3.3	4.2	3.0	3.2
25,600	10^7	7	3.0	2.7	4.6	4.2	14.1	7.7	10.0	5.5	5.9
>25,600	10^8	3	.6	.6	9.2	8.7	15.8	9.9	11.2	8.8	.9

Although all methods generated the same profile for many of the 14,842 matrices in the first test set, there were some cases where one method was uniquely superior. As expected, we see in Table 2 that the exchange method x was more often uniquely superior than the other methods. Surprisingly, the spectral/exchange combination sx, ranking fourth in Table 1, was uniquely superior in 80 cases. Although this is a small number of cases relative to the 14,842 problems, it is significantly larger than the number of cases for any of the other methods in Table 2.

In order to estimate the deviation between the profiles of Table 1 and the smallest possible profile, we considered 10,000 random permutations of each matrix. In each case, we applied Algorithm 3 and 4 and recorded the smallest profile. For these relatively small matrices, the smallest profile generated from these 10,000 starting guesses should be very close to the smallest possible profile. In fact, over the entire set of 14,842 matrices, there were only 14 cases where any of the heuristics generated a profile strictly better than the best obtained from these 10,000 random starting guesses, and in these 14 cases, the heuristics yielded a total improvement of just 137. Hence, we expect that the smallest profile for these 14,842 matrices is very close to the profile 1,263,882 achieved using the 10,000 random starting guesses. Referring to Table 1, method x differs from the estimated optimal profile by less than a percent.

In Test Set 2, we apply each of the profile schemes to the matrix $\mathbf{AA}^\mathsf{T}$ associated with each linear program. In Table 3, the first column indicates the dimension range, the row scale factor, and the number of matrices. For example, the first row of Table 3 corresponds to a set of 13 matrices whose dimensions lie between 1 and 100, and the profiles in that row should be multiplied by 10^3 to obtain the total profile of the 13 matrices in that class. The second row corresponds to a set of 29 matrices of dimension between 101 and 400, and the profiles in that row should be multiplied by 10^5. For the matrices of dimension up to 400, the exchange method x gave the best results. For larger matrices, weighted greed combined with the exchange methods gave the best profiles. As the dimension of a matrix increases, the 10 random permutations used in method x become an increasingly small subset of the total set of permutations.

Running times on a Sun Ultra 10 computer are given in Table 4 for weighted greed, MC60, and the exchange methods. The time for the exchange method is subdivided into x3, corresponding to Algorithm 3, and x4, corresponding to Algorithm 4. The

Table 4
Total running times for Test Set 2.

Dim	w	wx	m	mx	x3	x4
100	.060	.120	.003	.011	.006	.009
400	.728	1.527	.083	.241	.090	.271
1600	3.222	7.748	.569	2.297	.750	3.603
6400	6.183	23.600	5.530	16.680	4.328	21.264
25,600	9.064	63.460	1.392	48.560	19.136	66.509
>25,600	18.430	157.800	2.684	186.900	648.636	258.818

times for x3 and x4 were the total times of Algorithms 3 and 4 divided by 11 (the number of permutations including the starting matrix). MC60 is coded in Fortran while the other algorithms are coded in C. Note that MC60 runs several times faster than weighted greed. The complexity estimate $\mathrm{O}(\mathrm{nnz}(\mathbf{A})+n \log n)$ for Sloan's method, given in [22], when compared to the complexity estimate $\mathrm{O}(\mathrm{nnz}(\mathbf{A}) \log n)$ for weighted greed in Lemma 2.1, also seems to suggest that Sloan's method should be faster than our current implementation, which is patterned after the proof of Lemma 2.1. Observe that when the exchange methods are applied after either weighted greed or Sloan's method, overall run time can increase by several factors. Nonetheless, the time to factor a matrix is often much less than the time used by the exchange method. For example, Matlab's built-in Cholesky factorization routine (chol) applied to a matrix with the same nonzero pattern as that of the eighth NASA problem, skirt, takes 33 secs, compared to .1 secs by MC60 to generate a low profile permutation, and 3.7 secs for both MC60 and the exchange methods. The combination mx reduces the profile from 3,766,168 down to 687,730.

Focusing on the times for the exchange methods in Table 4, we note that Algorithm 3 is generally much faster than Algorithm 4. The one exception is the last line of Table 4. This line corresponds to 3 matrices, and one of these, ken18, dominates the total time. This matrix, with 105,127 rows, comes from the manufacturer with a relatively nice row ordering, and a random permutation generates a horrendous ordering. Algorithm 3, which is the first one to process the permuted matrix, makes many interchanges while improving the profile. Even though Algorithm 3 is faster than Algorithm 4, the number of interchanges is so large that the Algorithm 3 time dominates.

We emphasize that the x3 and x4 times in Table 4 correspond to the exchange methods applied to a single permuted version of the original matrix (the time was evaluated by averaging over a set of permutations). It is clear from the times given in Table 4 that trying to find an optimal profile by randomly permuting the matrix and improving the profile by exchanges is only appropriate when the matrix is relatively small.

In Table 5 we give the profiles associated with the Harwell-Boeing test set. These matrices are generally more structured than the linear programming matrices, and more sparse. Again, the exchange methods were competitive for dimensions up to 400. Thereafter, the MC60/exchange combination mx or the spectral/exchange combination sx gave the best profiles.

Running times, shown in Table 6, again parallel those of Table 4, with MC60 being the fastest code.

The profiles shown in Table 7 for Test Set 4, the largest and sparsest matrices, were similar to those of Table 5, with either mx or sx giving the best profiles in each case.

Table 5
Profiles for Test Set 3.

Dim	Scale	#	w	wx	m	mx	r	rx	s	sx	x
100	10^4	19	1.43	1.42	1.32	1.30	1.49	1.32	1.40	1.29	1.34
400	10^5	31	1.14	1.12	1.05	1.02	1.23	1.05	1.20	1.02	1.08
1600	10^6	71	2.07	2.00	1.82	1.73	2.36	1.86	2.21	1.88	2.56
6400	10^6	24	8.28	8.08	5.66	5.55	7.04	6.26	6.16	5.53	9.45
25,600	10^7	5	4.23	3.98	1.62	1.57	2.36	2.16	2.59	2.00	1.89
>25,600	10^7	3	8.58	8.52	7.28	7.27	9.91	9.84	5.12	4.54	13.60

Table 6
Total running times for Test Set 3.

Dim	w	wx	m	mx	x3	x4
100	.121	.247	.006	.021	.012	.026
400	.645	1.328	.051	.161	.053	.245
1600	6.375	15.330	.412	3.808	.927	9.236
6400	6.906	23.260	.717	10.360	2.696	22.191
25,600	7.040	25.300	1.139	15.280	4.222	35.100
>25,600	13.420	41.390	3.324	17.400	6.366	78.282

Table 7
Profiles for Test Set 4.

Dim	Scale	#	w	wx	m	mx	r	rx	s	sx	x
6400	10^6	4	2.28	2.26	1.64	1.64	1.89	1.86	1.73	1.48	1.79
25,600	10^6	2	2.84	2.82	1.34	1.29	1.81	1.67	1.79	1.61	11.23
>25,600	10^7	2	3.15	3.14	2.41	2.38	2.63	2.55	2.52	2.35	8.18

Table 8
Total running times for Test Set 4.

Dim	w	wx	m	mx	x3	x4
6400	1.249	2.545	.207	.451	.191	2.143
25,600	2.414	10.950	.199	8.347	2.216	17.627
>25,600	10.700	51.580	1.887	42.670	18.964	81.400

Again, in Table 8, we see that MC60 is significantly faster than the other codes.

These experiments seem to indicate that for matrices in the test suite with dimensions up to 400, the best or close to the best profiles are obtained by the exchange methods applied to a small number of random permutations of the starting matrix. If the exchange methods are applied after MC60, the profile improves as it must, but the computing time increases by several factors (at least in our implementation). By restricting the distance between the rows and columns that are exchanged, the execution time of the exchange methods is O(nnz($\mathbf{A}$)). For the matrices $\mathbf{AA}^{\mathsf{T}}$ associated with the linear programming test programs, weighted greed combined with the exchange methods provided the best profiles on average for matrices of dimension greater than 400. For the Harwell-Boeing and NASA test sets, which are generally more structured than the linear programming matrices, Sloan's method was the best single method, obtaining the best profile for each class of matrices except for the 3 matrices of dimension greater than 25,600 in Table 5. When the profiles for the single methods were further improved using the exchange methods, the combination sx achieved a better profile than mx in 5 of the 9 matrix classes associated with Tables 5 and 7.

Acknowledgment. The author gratefully acknowledges the thoughtful and detailed comments of the referees, which led to an improved presentation.

REFERENCES

[1] S. T. Barnard, A. Pothen, and H. Simon, *A spectral algorithm for envelope reduction of sparse matrices*, Numer. Linear Algebra Appl., 2 (1995), pp. 317–334.

[2] E. G. Boman and B. Hendrickson, *A Multilevel Algorithm for Reducing the Envelope of Sparse Matrices*, Tech. Report SCCM-96-14, Stanford University, Stanford, CA, 1996.

[3] E. Cuthill and J. McKee, *Reducing the bandwidth of sparse symmetric matrices*, in Proceedings of the ACM National Conference, ACM, New York, 1969, pp. 157–172.

[4] I. S. Duff, R. G. Grimes, and J. G. Lewis, *Sparse matrix test problems*, ACM Trans. Math. Software, 15 (1989), pp. 1–14.

[5] I. S. Duff, J. K. Reid, and J. A. Scott, *The use of profile reduction algorithms with a frontal code*, Internat. J. Numer. Methods Engrg., 28 (1989), pp. 2555–2568.

[6] C. M. Fiduccia and R. M. Mattheyses, *A linear time heuristic for improving network partitions*, in Proceedings of the 19th IEEE Design Automation Conference, Las Vegas, NV, 1982, IEEE, Piscataway, NJ, 1982, pp. 175–181.

[7] A. George and J. W. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1981.

[8] A. George and A. Pothen, *An analysis of spectral envelope reduction via quadratic assignment problems*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 706–732.

[9] N. E. Gibbs, W. G. Poole, and P. K. Stockmeyer, *An algorithm for reducing the bandwidth and profile of a sparse matrix*, SIAM J. Numer. Anal., 13 (1976), pp. 236–250.

[10] W. W. Hager, *Iterative methods for nearly singular linear systems*, SIAM J. Sci. Comput., 22 (2000), pp. 747–766.

[11] W. W. Hager, *The LP dual active algorithm*, in High Performance Algorithms and Software in Nonlinear Optimization, R. De Leone, A. Murli, P. M. Pardalos, and G. Toraldo, eds., Kluwer, Dordrecht, 1998, pp. 243–254.

[12] W. W. Hager, C.-L. Shih, and E. O. Lundin, *Active Set Strategies and the LP Dual Active Set Algorithm*, Department of Mathematics, University of Florida, Gainesville, FL, 1996; available from www.math.ufl.edu/∼hager.

[13] B. Hendrickson and R. Leland, *A Multilevel Algorithm for Partitioning Graphs*, Tech. Report SAND93-1301, Sandia National Laboratories, Albuquerque, NM, 1993.

[14] B. Hendrickson and R. Leland, *An improved spectral graph partitioning algorithm for mapping parallel computations*, SIAM J. Sci. Comput., 16 (1995), pp. 452–469.

[15] B. Hendrickson and E. Rothberg, *Improving the run time and quality of nested dissection ordering*, SIAM J. Sci. Comput., 20 (1998), pp. 468–489.

[16] HSL, *A Collection of Fortran Codes for Large Scale Scientific Computation*, 2000; available online from http://www.cse.clrc.ac.uk/Activity/HSL.

[17] Y. F. Hu and J. A. Scott, *Multilevel Algorithms for Wavefront Reduction*, RAL-TR-2000-031, Rutherford Appleton Laboratory, Chilton, England, 2000; available online from www.numerical.rl.ac.uk/reports/reports.html.

[18] G. Karypis and V. Kumar, *A fast and high quality multilevel scheme for partitioning irregular graphs*, SIAM J. Sci. Comput., 20 (1998), pp. 359–392.

[19] G. Karypis and V. Kumar, *Multilevel k-way partitioning scheme for irregular graphs*, J. Parallel Distrib. Comput., 48 (1998), pp. 96–129.

[20] G. Karypis and V. Kumar, *Parallel multilevel k-way partitioning scheme for irregular graphs*, SIAM Rev., 41 (1999), pp. 278–300.

[21] B. Kernighan and S. Lin, *An efficient heuristic procedure for partitioning graphs*, Bell System Tech. J., 29 (1970), pp. 291–307.

[22] G. Kumfert and A. Pothen, *Two improved algorithms for reducing the envelope and wavefront*, BIT, 37 (1997), pp. 1–32.

[23] J. G. Lewis, *Implementations of the Gibbs–Poole–Stockmeyer and Gibbs–King algorithms*,

ACM Trans. Math. Software, 8 (1982), pp. 180–189.

[24] W.-H. Liu and A. H. Sherman, *Comparative analysis of the Cuthill–McKee and the reverse Cuthill–McKee ordering algorithms for sparse matrices*, SIAM J. Numer. Anal., 13 (1976), pp. 198–213.

[25] J. K. Reid and J. A. Scott, *Implementing Hager's exchange methods for matrix profile reduction*, RAL-TR-2001-039, Rutherford Appleton Laboratory, Chilton, England, 2001.

[26] J. K. Reid and J. A. Scott, *Ordering symmetric sparse matrices for small profile and wavefront*, Internat. J. Numer. Methods Engrg., 45 (1999), pp. 1737–1755.

[27] S. W. Sloan, *An algorithm for profile and wavefront reduction of sparse matrices*, Internat. J. Numer. Methods Engrg., 23 (1986), pp. 239–251.

SIAM J. SCI. COMPUT.
Vol. 23, No. 6, pp. 1817–1841

A WEAKLY OVERLAPPING DOMAIN DECOMPOSITION PRECONDITIONER FOR THE FINITE ELEMENT SOLUTION OF ELLIPTIC PARTIAL DIFFERENTIAL EQUATIONS[*]

RANDOLPH E. BANK[†], PETER K. JIMACK[‡], SARFRAZ A. NADEEM[‡], AND SERGEI V. NEPOMNYASCHIKH[§]

Abstract. We present a new two-level additive Schwarz domain decomposition preconditioner which is appropriate for use in the parallel finite element solution of elliptic partial differential equations (PDEs). As with most parallel domain decomposition methods each processor may be assigned one or more subdomains, and the preconditioner is such that the processors are able to solve their own subproblem(s) concurrently. The novel feature of the technique proposed here is that it requires just a single layer of overlap in the elements which make up each subdomain at each level of refinement, and it is shown that this amount of overlap is sufficient to yield an optimal preconditioner. Some numerical experiments—posed in both two and three space dimensions—are included to confirm that the condition number when using the new preconditioner is indeed independent of the level of mesh refinement on the test problems considered.

Key words. domain decomposition, Schwarz methods, sparse linear systems, finite element discretization

AMS subject classifications. 65F10, 65N22, 65N55

PII. S1064827501361425

1. Introduction.

In this paper we introduce a two-level overlapping additive Schwarz (AS) algorithm which may be applied as an optimal domain decomposition (DD) preconditioner for the adaptive finite element solution of a variety of second-order self-adjoint elliptic problems defined on a bounded Lipschitz domain $\Omega \subset \mathbb{R}^n$ ($n = 2, 3$). In recent years there has been a great amount of research into DD and related methods, and we refer to some of the recent survey and review articles, such as [11, 24, 27, 30, 36, 37], for further details. In particular we note that a number of the algorithms proposed have been successfully implemented as software (see, for example, [12, 16, 21]) and many take the form of multiplicative or multilevel methods (e.g., [3, 8, 10, 15, 38]). Furthermore, by viewing these iterative techniques in terms of subspace corrections it is possible to develop a unified theory for a variety of algorithms and so, although this paper mainly discusses a two-level AS approach, it is certainly possible to generalize this to a multiplicative or a multilevel variant. This is briefly considered in section 6.

Although the work in this paper applies in both two and three dimensions, the

[*] Received by the editors January 9, 2001; accepted for publication (in revised form) September 7, 2001; published electronically February 20, 2002.
http://www.siam.org/journals/sisc/23-6/36142.html

[†] Department of Mathematics, University of California at San Diego, La Jolla, CA 92093 (rbank@ucsd.edu). The work of this author was supported by the National Science Foundation under contract DMS-9706090.

[‡] School of Computing, University of Leeds, Leeds LS2 9JT, UK (pkj@comp.leeds.ac.uk, sarfraz@comp.leeds.ac.uk). The work of the second author was supported by a research grant from the Leverhulme Trust (while visiting UCSD) and by a British Council ARC grant (while visiting TU Chemnitz). The work of the third author was supported by the Pakistan Government through a Quaid-e-Azam scholarship.

[§] Computing Centre, Siberian Branch of the Russian Academy of Sciences, Novosibirsk 630090, Russia (svnep@mail.nsk.ru). The work of this author was supported by the Deutsche Forschungsgemeinschaft, Sonderforschungsbereich 393.

first four sections consider the following model problem in just two dimensions so as to simplify the explanations and illustrations provided. In section 5 the ideas presented are then generalized to three dimensions. It should also be noted that the ideas presented may also be applied to a wider variety of boundary conditions than the zero Dirichlet conditions imposed here for simplicity.

PROBLEM 1.1. *Find $u \in \mathcal{H}_0^1(\Omega)$ such that*

$$\mathcal{A}(u,v) = \mathcal{F}(v) \quad \forall v \in \mathcal{H}_0^1(\Omega), \tag{1.1}$$

where $\Omega \subset \mathbb{R}^2$ is the problem domain and

$$\mathcal{H}_0^1(\Omega) = \{u \in \mathcal{H}^1(\Omega) : u|_{\partial\Omega_E} = 0\}. \tag{1.2}$$

Here $\partial\Omega_E$ is a closed, nonempty subset of the boundary, $\partial\Omega$, upon which zero Dirichlet boundary conditions are imposed and $\mathcal{A}(\cdot,\cdot)$ and $\mathcal{F}(\cdot)$ are the bilinear and linear forms

$$\mathcal{A}(u,v) = \int_\Omega (P(\underline{x})\underline{\nabla} u) \cdot \underline{\nabla} v \, d\underline{x} \quad \text{and} \quad \mathcal{F}(v) = \int_\Omega fv \, d\underline{x} + \int_{\partial\Omega_N} gv \, ds, \tag{1.3}$$

where $P(\underline{x})$ is bounded, symmetric, and strictly positive-definite, and $\partial\Omega_N = \partial\Omega - \partial\Omega_E$ is the part of the boundary subject to Neumann boundary conditions: $\underline{n} \cdot (P(\underline{x})\underline{\nabla} u) = g(\underline{x})$.

The Galerkin finite element method for the solution of (1.1) requires a triangulation, $\mathcal{T}^h$ say, of Ω to be produced so that one may define a piecewise polynomial space of trial functions, $\mathcal{V}^h$ say, on $\mathcal{T}^h$. Further details of the construction of this triangulation are given in the following sections and, for the sake of clarity, we consider only continuous piecewise linear finite element spaces on $\mathcal{T}^h$ throughout the rest of this paper. Section 2 also provides background on the relevant theoretical and practical details of AS preconditioning that are required for section 3. This section introduces details of the DD preconditioner that we propose and presents a detailed analysis of its convergence properties. In the analysis it is demonstrated that it is possible to obtain an optimal preconditioner (i.e., with condition number independent of the mesh size and the number and size of the subdomains) with an overlap of just one element at each level of a mesh hierarchy. This is the main result of the paper. Finally, sections 4 and 5 present a small number of numerical examples using the proposed DD preconditioner for two- and three-dimensional problems, respectively. The paper concludes with a brief discussion of possible extensions and applications of this work.

2. Background.

In order to approximate the solution of (1.1) from the finite-dimensional space $\mathcal{V}^h$ (of continuous piecewise linears on $\mathcal{T}^h$ (where h is the diameter of the largest triangle)), it is necessary to solve the following discrete problem.

PROBLEM 2.1. *Find $u^h \in \mathcal{V}^h \cap \mathcal{H}_0^1(\Omega)$ such that*

$$\mathcal{A}(u^h, v^h) = \mathcal{F}(v^h) \quad \forall v^h \in \mathcal{V}^h \cap \mathcal{H}_0^1(\Omega). \tag{2.1}$$

This is achieved by choosing a basis for $\mathcal{V}^h$ and expressing the problem as a matrix equation:

$$A\underline{u} = \underline{b}. \tag{2.2}$$

For the usual, local, choice of basis the stiffness matrix A is sparse, symmetric, and strictly positive-definite, and so an iterative solution method, such as the conjugate gradient (CG) algorithm (e.g., [2, 17]), is most appropriate. However, it is well known that when the triangulation $\mathcal{T}^h$ is uniformly refined, the condition number of A grows like $O(h^{-2})$ as $h \to 0$ (see [23], for example); hence it is necessary to apply a preconditioned version of the CG algorithm for realistic mesh sizes h.

In this work we consider the use of AS preconditioning (e.g., [14, 13, 18, 25, 28, 33]), which is suitable for use with both nonuniformly refined meshes and parallel computer architectures. Let us define $\mathcal{V} = \mathcal{V}^h \cap \mathcal{H}_0^1(\Omega)$ to be the trial and test space in (2.1) and assume that the triangulation $\mathcal{T}^h$ may be obtained by the uniform refinement of some coarser triangulation, $\mathcal{T}^H$ say, of the domain Ω, where $\mathcal{V}_0$ is the corresponding piecewise linear finite element space, $\mathcal{V}^H \cap \mathcal{H}_0^1(\Omega)$, defined on $\mathcal{T}^H$.

Having introduced a coarse mesh $\mathcal{T}^H$ it is now possible to decompose Ω into (possibly overlapping) subdomains, $\Omega_1, \ldots, \Omega_p$ say, which are each the union of triangles in $\mathcal{T}^H$. We now define the spaces $\mathcal{H}_0^1(\Omega_i) \subset \mathcal{L}^2(\Omega)$ for $i = 1, \ldots, p$ to be the extensions of $\mathcal{H}^1(\Omega_i)$ for which

$$u(\underline{x}) = 0 \ \ \forall \underline{x} \in (\Omega - \overline{\Omega_i}) \cup \partial\Omega_E, \tag{2.3}$$

and the corresponding finite-dimensional spaces $\mathcal{V}_i = \mathcal{V}^h \cap \mathcal{H}_0^1(\Omega_i)$. Note that these local spaces, $\mathcal{V}_i$, form a decomposition of the finite element space $\mathcal{V}$:

$$\mathcal{V} = \sum_{i=1}^{p} \mathcal{V}_i. \tag{2.4}$$

Thus, for each $v \in \mathcal{V}$, there exists a (not necessarily unique) combination of $v_i \in \mathcal{V}_i$ $(i = 1, \ldots, p)$ such that $v = \sum_{i=1}^{p} v_i$.

Given any space decomposition of the form (2.4), the AS algorithm defines a preconditioner, B, for A in (2.2) in the following manner. Let Q_i be the projection from $\mathcal{V}$ to $\mathcal{V}_i$ (for $i = 1, \ldots, p$) given by

$$\int_\Omega (Q_i u) v_i \, d\underline{x} \ = \ \int_\Omega u v_i \, d\underline{x} \quad \forall u \in \mathcal{V}, v_i \in \mathcal{V}_i, \tag{2.5}$$

and define $\mathcal{A}_i$ to be the restriction of $\mathcal{A}$ to $\mathcal{V}_i \times \mathcal{V}_i$ given by

$$\mathcal{A}_i(u_i, v_i) = \mathcal{A}(u_i, v_i) \quad \forall u_i, v_i \in \mathcal{V}_i. \tag{2.6}$$

Note that (given the usual finite element bases for $\mathcal{V}$ and $\mathcal{V}_i$) Q_i may be expressed as a rectangular matrix, $\overline{Q}_i$ say, and a local stiffness matrix, A_i say, may be derived from $\mathcal{A}_i$ (in the same way that the global stiffness matrix A is derived from $\mathcal{A}$ above). The AS (parallel subspace correction) preconditioner for (2.2) is then given by

$$B = \sum_{i=1}^{p} \overline{Q}_i^T A_i^{-1} \overline{Q}_i. \tag{2.7}$$

Note that each of the subdomain solves $(A_i^{-1}\underline{r}_i)$, required when solving the system $B^{-1}\underline{s} = \underline{r}$ at each preconditioned CG iteration, may be performed concurrently and, for simplicity, we will assume for now that all such subdomain solves are exact.

The following theorem, which is proved in [36], for example (or see [30] for a slightly more general form), provides the main theoretical justification for considering preconditioners of the form (2.7).

THEOREM 2.1. *The matrix B defined by* (2.7) *is symmetric and positive-definite. Furthermore, if we assume that there is some constant $C > 0$ such that for all $v \in \mathcal{V}$ there are $v_i \in \mathcal{V}_i$ such that $v = \sum_{i=1}^{p} v_i$ and*

$$\sum_{i=1}^{p} \mathcal{A}_i(v_i, v_i) \leq C\mathcal{A}(v, v), \tag{2.8}$$

then the spectral condition number of BA is given by

$$\kappa(BA) \leq \nu_c C, \tag{2.9}$$

where ν_c is the minimum number of colors required to color the subdomains Ω_i in such a way that no neighbors are the same color.

This result demonstrates that the quality of any AS preconditioner depends only upon the stability of the splitting of $\mathcal{V}$ into subspaces $\mathcal{V}_i$. In particular, if the splitting is such that (2.8) holds with C independent of h, H, or p, then the preconditioner is said to be optimal.

Unfortunately, the decomposition described in (2.3) to (2.4) does not permit such a choice of C since it is entirely local in nature, and so significant reductions in the low-frequency error components can require many preconditioned CG iterations. This is easily rectified, however, by the introduction of an extra, coarse grid, term in the preconditioner (2.7):

$$B = \sum_{i=0}^{p} \overline{Q}_i^T A_i^{-1} \overline{Q}_i. \tag{2.10}$$

Here $\overline{Q}_0$ is another rectangular matrix corresponding to the $\mathcal{L}^2$ projection, Q_0 say, from $\mathcal{V}$ to the coarse grid space $\mathcal{V}_0$, given by

$$\int_\Omega (Q_0 u) v_0 \, d\underline{x} \;=\; \int_\Omega u v_0 \, d\underline{x} \quad \forall u \in \mathcal{V}, v_0 \in \mathcal{V}_0, \tag{2.11}$$

and A_0 is the stiffness matrix derived from $\mathcal{A}_0$, the restriction of $\mathcal{A}$ to $\mathcal{V}_0 \times \mathcal{V}_0$ given by

$$\mathcal{A}_0(u_0, v_0) = \mathcal{A}(u_0, v_0) \quad \forall u_0, v_0 \in \mathcal{V}_0. \tag{2.12}$$

The following result is also proved in [30] and [36] and applies to the new two-level preconditioner defined in (2.10).

THEOREM 2.2. *Provided the overlap between the subdomains Ω_i is of size $O(H)$, where H represents the mesh size of $\mathcal{T}^H$, then there exists $C > 0$, which is independent of h, H, and p, such that for any $v \in \mathcal{V}$ there are $v_i \in \mathcal{V}_i$ such that $v = \sum_{i=0}^{p} v_i$ and*

$$\sum_{i=0}^{p} \mathcal{A}_i(v_i, v_i) \leq C\mathcal{A}(v, v). \tag{2.13}$$

The above result demonstrates that, provided a coarse-grid solve is undertaken and there is a "generous" overlap between the subdomains, the AS technique may indeed be used to achieve optimal preconditioning. It should be noted, however, that these two provisos do raise important practical concerns over the efficiency of such a preconditioner. For example, the solution of the coarse-grid problem is hard to achieve in parallel and so care must be taken when developing parallel software to ensure that this does not become a significant bottleneck. More importantly, however, the fixed $O(H)$ overlap that is required between the subdomains means that as the mesh $\mathcal{T}^h$ is refined (assuming uniform global refinement for simplicity), the number of elements of $\mathcal{T}^h$ in the overlap regions is $O(h^{-2})$ as $h \to 0$. This represents a significant computational overhead when h becomes small.

In practice the usual way in which this second issue is addressed (see, for example, [30]) is to drop the optimality requirement and allow subdomains only to overlap by a small, fixed number of fine element layers. In the following section we address this issue in a different manner by proposing a new two-level optimal AS preconditioner of the form (2.10), which requires substantially fewer elements in the overlap region as $\mathcal{T}^h$ is refined ($O(h^{-1})$ as $h \to 0$ as opposed to $O(h^{-2})$). This is achieved by considering a hierarchy of meshes between $\mathcal{T}^H$ and $\mathcal{T}^h$, each defined by a single level of refinement of its predecessor. While the total overlap between the subdomains remains $O(H)$ in size, at each level of the mesh hierarchy the overlap is the width of just one element. This is illustrated in Figure 1, which shows an overlap of size H in two cases: the first with a uniformly refined mesh in the overlap region, and the second with a mesh which is refined into the overlap region to a width of just one element at each level of the mesh hierarchy.

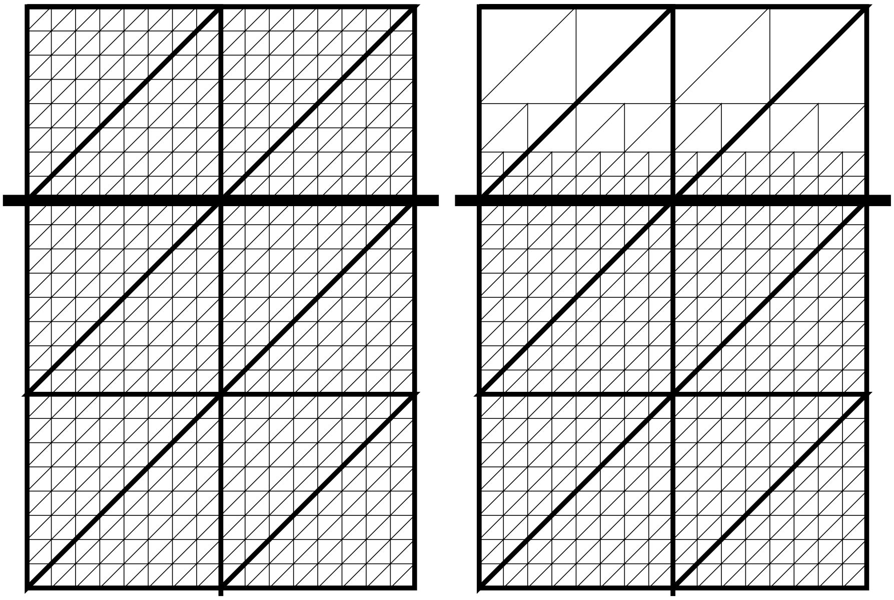


Fig. 1. *A comparison between a mesh which is uniformly refined in the overlap region (left) and one that is nonuniformly refined in the overlap region (right).*

3. A new preconditioner. In order to describe the DD preconditioner that we propose in this section it is necessary to begin by establishing some notation. We will again consider (1.1) to (1.3) as a test problem, maintaining the zero Dirichlet boundary conditions for simplicity. Many of the technical details that follow are concerned with ensuring that both the method itself and the analysis that follows are completely general with respect to the domain geometry and how it is decomposed.

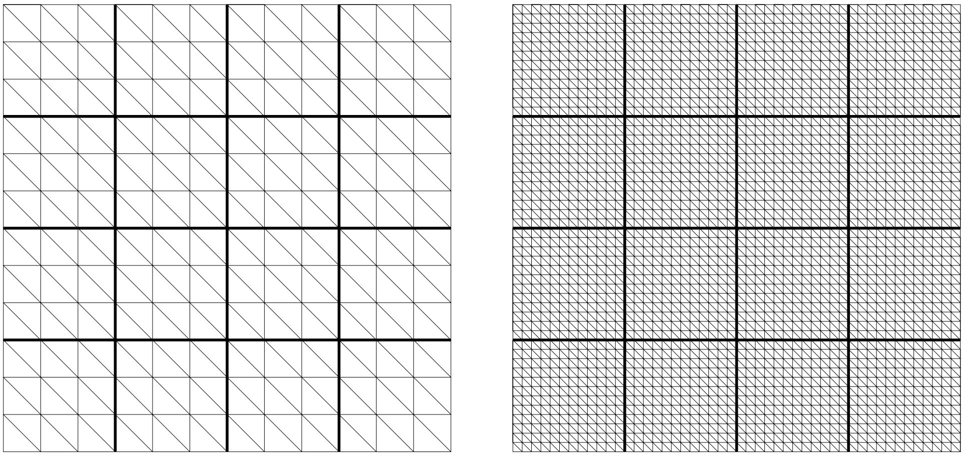


Fig. 2. *The model example, illustrating a regular Cartesian product decomposition of a simple rectangular domain: coarse grid $\mathcal{T}_0$ (left) and uniformly refined grid $\mathcal{T}_2$ (right).*

On first reading, however, it might be more straightforward to visualize the ideas presented by considering a less than general situation. For this reason we also provide a specific model example using a uniformly refined rectangular domain with a regular Cartesian product decomposition (see Figure 2).

Let $\mathcal{T}_0$ be a coarse triangulation of Ω consisting of N_0 triangular elements, $\tau_j^{(0)}$, such that $\tau_j^{(0)} = \overline{\tau}_j^{(0)}$,

$$\overline{\Omega} = \bigcup_{j=1}^{N_0} \tau_j^{(0)} \quad \text{and} \quad \mathcal{T}_0 = \{\tau_j^{(0)}\}_{j=1}^{N_0}. \tag{3.1}$$

Also let diameter$(\tau_j^{(0)}) = O(H)$ (so this triangulation could also be referred to as $\mathcal{T}^H$ in the notation of the previous section), and divide Ω into p *nonoverlapping* subdomains Ω_i. These subdomains should be such that

$$\overline{\Omega} = \bigcup_{i=1}^{p} \overline{\Omega}_i, \tag{3.2}$$

$$\Omega_i \cap \Omega_j = \phi \quad (i \neq j), \tag{3.3}$$

$$\overline{\Omega}_i = \bigcup_{j \in I_i} \tau_j^{(0)}, \quad \text{where } I_i \subset \{1, \ldots, N_0\} \quad (I_i \neq \phi). \tag{3.4}$$

We now permit $\mathcal{T}_0$ to be refined several times to produce a family of triangulations, $\mathcal{T}_0, \ldots, \mathcal{T}_J$, where each triangulation, $\mathcal{T}_k$, consists of N_k elements, $\tau_j^{(k)}$, such that

$$\overline{\Omega} = \bigcup_{j=1}^{N_k} \tau_j^{(k)} \quad \text{and} \quad \mathcal{T}_k = \{\tau_j^{(k)}\}_{j=1}^{N_k}. \tag{3.5}$$

The successive mesh refinements that define this sequence of triangulations need not be global and may be nonconforming; however, we do require that they satisfy a number of conditions, as in [9], for example:

1. $\tau \in \mathcal{T}_{k+1}$ implies that either
   (a) $\tau \in \mathcal{T}_k$, or
   (b) τ has been generated as a refinement of an element of $\mathcal{T}_k$ into four similar children;
2. the level of any triangles which share a common point can differ by at most one;
3. only triangles at level k may be refined in the transition from $\mathcal{T}_k$ to $\mathcal{T}_{k+1}$.

(Here the level of a triangle is defined to be the least value of k for which that triangle is an element of $\mathcal{T}_k$.)

In addition to the above we will also require that

4. in the final mesh, $\mathcal{T}_J$, all pairs of triangles on either side of the boundary of each subdomain Ω_i have the same level as each other.

Note that Figure 2 shows a simple example of such a nested sequence of triangulations for $J = 2$. In this case every triangle in $\mathcal{T}_2$ is a level 2 triangle and the number of subdomains, p, is 16.

Having defined a decomposition of Ω into subdomains and a nested sequence of triangulations of Ω we next define the restrictions of each of these triangulations onto each subdomain by

$$\Omega_{i,k} = \{\tau_j^{(k)} : \tau_j^{(k)} \subset \overline{\Omega}_i\}. \tag{3.6}$$

In order to introduce a certain amount of overlap between neighboring subdomains we also define

$$\tilde{\Omega}_{i,k} = \{\tau_j^{(k)} : \tau_j^{(k)} \text{ has a common point with } \overline{\Omega}_i\}. \tag{3.7}$$

Following this we introduce the finite element spaces associated with these local triangulations. Let G be some triangulation, and denote by $\mathcal{S}(G)$ the space of continuous piecewise linear functions on G. Then we can make the following definitions:

$$\mathcal{W} = \mathcal{S}(\mathcal{T}_J), \tag{3.8}$$

$$\mathcal{W}_0 = \mathcal{S}(\mathcal{T}_0), \tag{3.9}$$

$$\mathcal{W}_{i,k} = \mathcal{S}(\Omega_{i,k}), \tag{3.10}$$

$$\tilde{\mathcal{W}}_{i,k} = \mathcal{S}(\tilde{\Omega}_{i,k}), \tag{3.11}$$

$$\tilde{\mathcal{W}}_i = \tilde{\mathcal{W}}_{i,0} + \cdots + \tilde{\mathcal{W}}_{i,J}. \tag{3.12}$$

It is evident that

$$\mathcal{W} = \mathcal{W}_0 + \tilde{\mathcal{W}}_1 + \cdots + \tilde{\mathcal{W}}_p, \tag{3.13}$$

and this is the decomposition that we propose for the two-level AS preconditioner of the form (2.10). Figure 3 illustrates the meshes $\tilde{\Omega}_{i,k}$ that are the basis for the decomposition (3.13) in the case of our regularly decomposed model problem (where i is the number of the top left subdomain and $k = 0$, 1, and 2).

In order to prove that this preconditioner is optimal Theorem 2.1 demonstrates that it is sufficient, given any $u^h \in \mathcal{W}$, to provide a construction for $u_0^h \in \mathcal{W}_0$ and $u_i^h \in \tilde{\mathcal{W}}_i$ $(i = 1, \ldots, p)$ such that

$$u^h = \sum_{i=0}^{p} u_i^h \tag{3.14}$$

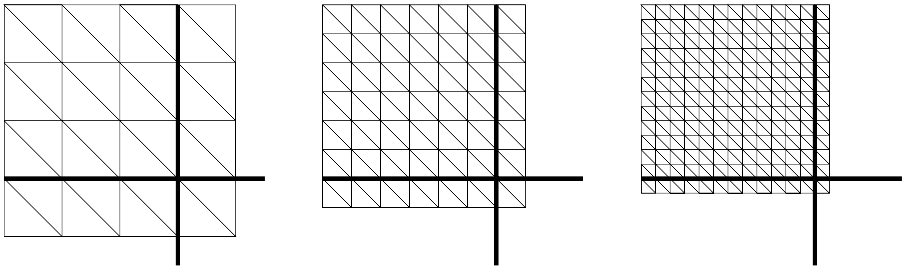


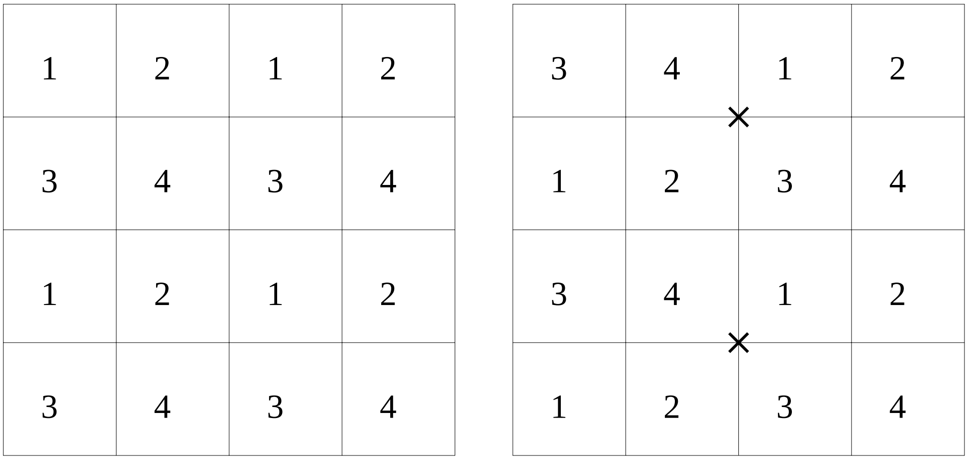
FIG. 3. *An illustration of* $\tilde{\Omega}_{i,0}$ *(left),* $\tilde{\Omega}_{i,1}$ *(center), and* $\tilde{\Omega}_{i,2}$ *(right), where subdomain i is the top left subdomain of Figure* 2.

and

$$\sum_{i=0}^{p} \mathcal{A}_i(u_i^h, u_i^h) \leq C\mathcal{A}(u_i^h, u_i^h) \tag{3.15}$$

for some $C > 0$ which is independent of h, H, and p. To allow such a construction to be produced it is necessary first to introduce some further notation and then to prove a preliminary lemma, following an approach similar to that used in [19, 20, 26].

Given the set of nonoverlapping subdomains Ω_i we define a coloring of the Ω_i such that no neighboring subdomains are the same color and that the boundary of each subdomain of color m should have no isolated points in common with the union of the boundary of all subdomains of color 1 to $m - 1$. Let the number of colors required be n_c, which we assume is independent of h, H, and p (although it may be slightly different to ν_c appearing in Theorem 2.1). Figure 4 illustrates an example of a suitable coloring (with $n_c = 4$) for the subdomains defined in Figure 2. It also illustrates a second example, which violates the restriction that each subdomain of color m should have no isolated points in common with the union of the boundary of subdomains of lower-numbered colors.



1	2	1	2
3	4	3	4
1	2	1	2
3	4	3	4

3	4	1	2
1	2	3	4
3	4	1	2
1	2	3	4

FIG. 4. *Examples of a valid (left) and an invalid (right) coloring of the subdomains used in the example of Figure* 2. *The points marked with an X are isolated points on the boundary of subdomains of color* 2 *that are also on the boundary of subdomains of color* 1.

Having introduced an appropriate coloring of the subdomains, let $c(m)$ denote the set of indices of those subdomains of color m (for $m = 1, \dots, n_c$). We may now define

$$\partial\Omega_i = \text{ the boundary of } \Omega_i, \tag{3.16}$$

$$\Omega_{c(m)} = \bigcup_{i \in c(m)} \Omega_i, \tag{3.17}$$

$$\partial\Omega_{c(m)} = \text{ the boundary of } \Omega_{c(m)} \tag{3.18}$$

and, for each $i \in c(m)$ $(m = 1, \dots, n_c)$,

$$\Gamma_i = \partial\Omega_E \bigcup \left(\partial\Omega_i \bigcap \left(\bigcup_{n=1}^{m-1} \partial\Omega_{c(n)} \right) \right). \tag{3.19}$$

Figure 5 illustrates these sets Γ_i for our regularly decomposed model problem and the valid coloring shown in Figure 4. Furthermore, with this definition of Γ_i, it is possible to introduce three more finite element spaces, $\mathcal{W}_{i,k,0}$, $\hat{\mathcal{W}}_{i,k}$, and $\hat{\mathcal{W}}_i$, which are subspaces of $\mathcal{W}_{i,k}$, $\tilde{\mathcal{W}}_{i,k}$, and $\tilde{\mathcal{W}}_i$, respectively:

$$\mathcal{W}_{i,k,0} = \{u_i^h \in \mathcal{W}_{i,k} : u_i^h(\underline{x}) = 0 \ \ \forall \underline{x} \in \Gamma_i\}, \tag{3.20}$$

$$\hat{\mathcal{W}}_{i,k} = \mathcal{S}(\hat{\Omega}_{i,k}), \tag{3.21}$$

and

$$\hat{\mathcal{W}}_i = \hat{\mathcal{W}}_{i,0} + \cdots + \hat{\mathcal{W}}_{i,J}, \tag{3.22}$$

where

$$\hat{\Omega}_{i,k} = \left\{ \bigcup_j \tau_j^{(k)} : \tau_j^{(k)} \text{ has a common point with } \overline{\Omega}_i - \Gamma_i \right\}. \tag{3.23}$$

Also, let $\hat{\Omega}_i$ be the subset of Ω which is covered by the triangulation $\hat{\Omega}_{i,0}$. Figure 6 illustrates the meshes $\hat{\Omega}_{i,k}$ for one specific choice of i using the first coloring shown in Figure 4. Note that this construction is such that overlap from subdomain i is only allowed to occur into subdomains with a larger-numbered color than the color of subdomain i; see Figure 5.

We are now ready to define a mechanism for extending an arbitrary function $w_i^h \in \mathcal{W}_{i,J,0}$ to $\hat{\mathcal{W}}_i$, as follows. In defining this mechanism, we note that the vertices of $\hat{\Omega}_{i,k} - \overline{\Omega}_{i,k}$ are easily identified in the example shown in Figure 6.

Algorithm 3.1. *Let $w_i^h \in \mathcal{W}_{i,J,0}$. Let $Q_{i,k} : \mathcal{L}^2(\Omega_i) \to \mathcal{W}_{i,k,0}$ be the usual $\mathcal{L}^2$ orthogonal projection onto $\mathcal{W}_{i,k,0}$ and define*

$$v_{i,0}^h = Q_{i,0} w_i^h \quad \textit{and} \quad v_{i,k}^h = (Q_{i,k} - Q_{i,k-1}) w_i^h \quad \textit{for } k = 1, \dots, J. \tag{3.24}$$

Now denote by $\hat{v}_{i,k}^h \in \hat{\mathcal{W}}_{i,k}$ the extension of $v_{i,k}^h$ which is zero at all vertices of $\hat{\Omega}_{i,k} - \overline{\Omega}_{i,k}$, so it easily follows that

$$\|\hat{v}_{i,k}^h\|^2_{\mathcal{L}^2(\hat{\Omega}_{i,k})} \le C_0 \|v_{i,k}^h\|^2_{\mathcal{L}^2(\Omega_i)} \tag{3.25}$$

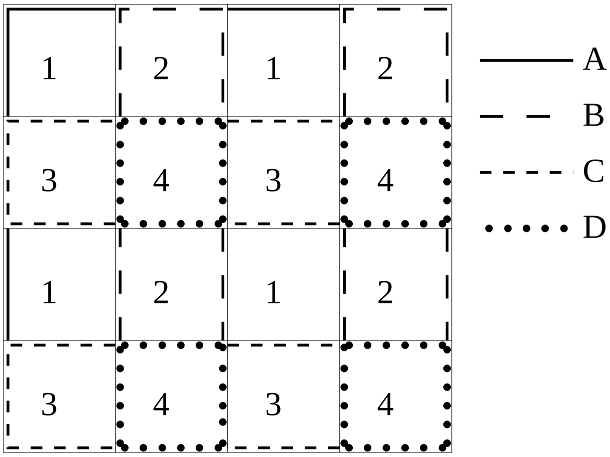


FIG. 5. *An illustration of* Γ_i *for the valid coloring shown in Figure* 4*: line A is* $\cup_{i\in c(1)}\Gamma_i \cap \overline{\Omega}_i$*; line B is* $\cup_{i\in c(2)}\Gamma_i \cap \overline{\Omega}_i$*; line C is* $\cup_{i\in c(3)}\Gamma_i \cap \overline{\Omega}_i$*; line D is* $\cup_{i\in c(4)}\Gamma_i \cap \overline{\Omega}_i$*.*

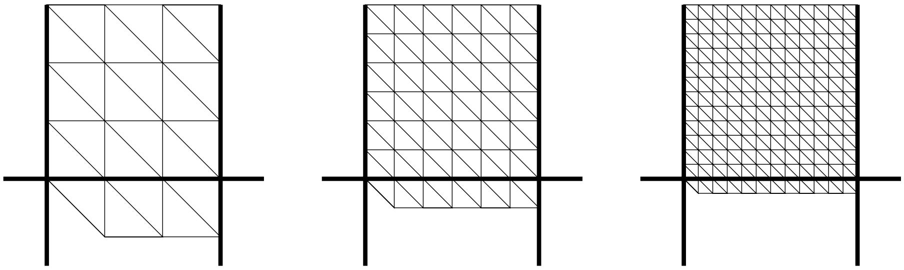


FIG. 6. *An illustration of* $\hat{\Omega}_{i,0}$ *(left),* $\hat{\Omega}_{i,1}$ *(center), and* $\hat{\Omega}_{i,2}$ *(right), where subdomain i is the subdomain in the top row and second column of Figures* 2 *and* 5 *(i.e., of color number* 2*).*

for some $C_0 > 0$*, which is independent of* h*,* H*, and* p*. We are now in a position to define*

$$\hat{v}_i^h = \hat{v}_{i,0}^h + \cdots + \hat{v}_{i,J}^h, \tag{3.26}$$

which is the required local extension of $w_i^h \in \mathcal{W}_{i,J,0}$ *to* $\hat{\mathcal{W}}_i$*.*

LEMMA 3.1. *Given* $w_i^h \in \mathcal{W}_{i,J,0}$ *let* $\hat{v}_i^h \in \hat{\mathcal{W}}_i$ *be the extension of* w_i^h *defined by Algorithm* 3.1 *above. Then there exists* $C_1 > 0$*, which is independent of* h*,* H*, and* p*, such that*

$$\|\hat{v}_i^h\|^2_{\mathcal{H}^1(\hat{\Omega}_i)} \le C_1 \left\{ \frac{1}{H^2}\|w_i^h\|^2_{\mathcal{L}^2(\Omega_i)} + |w_i^h|^2_{\mathcal{H}^1(\Omega_i)} \right\}. \tag{3.27}$$

Proof. First we introduce the following change of variables:

$$x = H\,s, \qquad y = H\,t\,; \qquad (x,y) \in \Omega_i. \tag{3.28}$$

Under this transformation the domain Ω_i is the image of a domain Ω_i' whose geometric properties are independent of H in the (s,t) plane. Furthermore

$$\frac{1}{H^2}\|w_i^h(x,y)\|^2_{\mathcal{L}^2(\Omega_i)} + |w_i^h(x,y)|^2_{\mathcal{H}^1(\Omega_i)} = \|w_i^h(s,t)\|^2_{\mathcal{L}^2(\Omega_i')} + |w_i^h(s,t)|^2_{\mathcal{H}^1(\Omega_i')}, \tag{3.29}$$

and we may define by $Q'_{i,k}$ the projection in the (s,t) variables which corresponds to $Q_{i,k}$. From [9] it follows that there exists $C_2 > 0$, which is independent of h, H, and p, such that

$$\begin{aligned}
&\frac{1}{H^2}\sum_{k=0}^{J} 4^k \|v_{i,k}^h\|^2_{\mathcal{L}^2(\Omega_i)} \\
&\qquad = \frac{1}{H^2}\left(\|Q_{i,0} w_i^h(x,y)\|^2_{\mathcal{L}^2(\Omega_i)} + \sum_{k=1}^{J} 4^k \|(Q_{i,k} - Q_{i,k-1}) w_i^h(x,y)\|^2_{\mathcal{L}^2(\Omega_i)} \right) \\
&\qquad = \|Q'_{i,0} w_i^h(s,t)\|^2_{\mathcal{L}^2(\Omega'_i)} + \sum_{k=1}^{J} 4^k \|(Q'_{i,k} - Q'_{i,k-1}) w_i^h(s,t)\|^2_{\mathcal{L}^2(\Omega'_i)} \\
&\qquad \le C_2 \|w_i^h(s,t)\|^2_{\mathcal{H}^1(\Omega'_i)} \\
&\qquad = C_2 \left(\frac{1}{H^2} \|w_i^h(x,y)\|^2_{\mathcal{L}^2(\Omega_i)} + |w_i^h(x,y)|^2_{\mathcal{H}^1(\Omega_i)} \right). \qquad (3.30)
\end{aligned}$$

A second inequality that we require comes from [26], where it is shown that there exists $C_3 > 0$, which is independent of h, H, and p, such that

$$\begin{aligned}
\|\hat{v}_i^h(x,y)\|^2_{\mathcal{H}^1(\hat{\Omega}_i)} &= H^2 \|\hat{v}_i^h(s,t)\|^2_{\mathcal{L}^2(\hat{\Omega}'_i)} + |\hat{v}_i^h(s,t)|^2_{\mathcal{H}^1(\hat{\Omega}'_i)} \\
&\le C_3 \inf_{\substack{\hat{v}_i^h = \hat{\xi}_0 + \cdots + \hat{\xi}_J \\ (\hat{\xi}_k \in \hat{\mathcal{W}}'_{i,k})}} \sum_{k=0}^{J} 4^k \|\hat{\xi}_k\|^2_{\mathcal{L}^2(\hat{\Omega}'_i)}, \qquad (3.31)
\end{aligned}$$

where $\hat{\mathcal{W}}'_{i,k}$ is the space which corresponds to $\hat{\mathcal{W}}_{i,k}$ with the change of variables (3.28). From this, along with (3.25) and (3.30), it follows that

$$\begin{aligned}
\|\hat{v}_i^h(x,y)\|^2_{\mathcal{H}^1(\hat{\Omega}_i)} &\le C_3 \sum_{k=0}^{J} 4^k \|\hat{v}_{i,k}^h\|^2_{\mathcal{L}^2(\hat{\Omega}'_i)} \\
&= \frac{C_3}{H^2} \sum_{k=0}^{J} 4^k \|\hat{v}_{i,k}^h\|^2_{\mathcal{L}^2(\hat{\Omega}_{i,k})} \\
&\le \frac{C_0 C_3}{H^2} \sum_{k=0}^{J} 4^k \|v_{i,k}^h\|^2_{\mathcal{L}^2(\Omega_i)} \\
&\le C_0 C_2 C_3 \left(\frac{1}{H^2} \|w_i^h\|^2_{\mathcal{L}^2(\Omega_i)} + |w_i^h|^2_{\mathcal{H}^1(\Omega_i)} \right), \qquad (3.32)
\end{aligned}$$

as required. □

This lemma forms the main component of our proof that the proposed splitting is stable. The following theorem completes this proof by explicitly constructing a suitable decomposition of any $u^h \in \mathcal{W}$. It should be noted that the proof of the theorem holds for an arbitrary partition of Ω into subdomains Ω_i and could certainly be simplified if less general partitions (e.g., into strips or regular blocks) were considered. For the completely general case it is necessary to introduce a small amount of additional notation. For $m \in \{1, \ldots, n_c\}$ let

$$\mathcal{W}_0^{(m)} = \left\{ u^H \in \mathcal{W}_0 : u^H(\underline{x}) = 0 \ \ \forall \underline{x} \in \bigcup_{n=1}^{m-1} \Omega_{c(n)} \right\} \qquad (3.33)$$

and let $Q_0^{(m)} : \mathcal{L}^2(\Omega) \to \mathcal{W}_0^{(m)}$ be the $\mathcal{L}^2$ orthogonal projection onto $\mathcal{W}_0^{(m)}$ given by

$$\int_\Omega (Q_0^{(m)} u) w_0^{(m)} d\underline{x} \ = \ \int_\Omega u w_0^{(m)} d\underline{x} \quad \forall w_0^{(m)} \in \mathcal{W}_0^{(m)}. \tag{3.34}$$

Hence, by the $\mathcal{H}^1$ stability of the $\mathcal{L}^2$ projection we have

$$\|Q_0^{(m)} u^h\|_{\mathcal{H}^1(\Omega)} \le C_4 \|u^h\|_{\mathcal{H}^1(\Omega)}, \tag{3.35}$$

and using standard finite element interpolation estimates we have

$$\frac{1}{H}\|u^h - Q_0^{(m)} u^h\|_{\mathcal{L}^2(\Omega)} + |u^h - Q_0^{(m)} u^h|_{\mathcal{H}^1(\Omega)} \le C_4 \|u^h\|_{\mathcal{H}^1(\Omega)} \tag{3.36}$$

for some $C_4 > 0$, which is independent of h, H, and p. (Note that in the case $m = 1$, $\mathcal{W}_0^{(1)} = \mathcal{W}_0$ and $Q_0^{(1)} = Q_0$ as defined in section 2 above.)

THEOREM 3.2. *There exists $C > 0$, which is independent of h, H, and p, such that for any $u^h \in \mathcal{W}$ there are $u^H \in \mathcal{W}_0$ and $u_i \in \tilde{\mathcal{W}}_i$ ($i = 1, \ldots, p$) such that*

$$u^h = u^H + u_1^h + \cdots + u_p^h \tag{3.37}$$

and

$$\|u^H\|^2_{\mathcal{H}^1(\Omega)} + \|u_1^h\|^2_{\mathcal{H}^1(\Omega)} + \cdots + \|u_p^h\|^2_{\mathcal{H}^1(\Omega)} \le C \|u^h\|^2_{\mathcal{H}^1(\Omega)}. \tag{3.38}$$

Proof. Given any $u^h \in \mathcal{W}$, we now construct functions $u^H \in \mathcal{W}_0$ and $u_i^h \in \hat{\mathcal{W}}_i \subset \tilde{\mathcal{W}}_i$ (for $i = 1, \ldots, p$) such that (3.37) and (3.38) are satisfied.

```
Let $r_1^h = u^h$.
Let $u_1^H = Q_0^{(1)} r_1^h$.
Let $w_1^h = r_1^h - u_1^H$.
Let $[w_1^h]_{\Omega_i}$ be the restriction of $w_1^h$ to $\Omega_i$.
For each $i \in c(1)$:
    use Algorithm 3.1 to define $u_i^h \in \hat{\mathcal{W}}_i$ to be the extension of $[w_1^h]_{\Omega_i}$.
For $m = 2$ to $n_c$.
    Let $r_m^h = w_{m-1}^h - \sum_{j \in c(m-1)} u_j^h$ (hence $r_m^h(\underline{x}) = 0$ for all $\underline{x} \in \bigcup_{n=1}^{m-1} \Omega_{c(n)}$).
    Let $u_m^H = Q_0^{(m)} r_m^h$.
    Let $w_m^h = r_m^h - u_m^H$.
    Let $[w_m^h]_{\Omega_i}$ be the restriction of $w_m^h$ to $\Omega_i$ (hence $[w_m^h]_{\Omega_i} \in \mathcal{W}_{i,J,0}$).
    For each $i \in c(m)$:
        use Algorithm 3.1 to define $u_i^h \in \hat{\mathcal{W}}_i$ to be the extension of $[w_m^h]_{\Omega_i}$.
Let $u^H = u_1^H + \cdots + u_{n_c}^H$.
```

Note that in the above definitions, when $i \in c(n_c)$ the functions u_i are just the restrictions of $w_{n_c}^h$ to Ω_i since the extension operation is just the identity in this case (because $\Gamma_i = \partial\Omega_i$ for $i \in c(n_c)$ and so $\hat{\Omega}_{i,k} = \Omega_{i,k}$ and $\hat{\mathcal{W}}_{i,k} = \mathcal{W}_{i,k}$ for $k = 1, \ldots, J$). For these definitions of u^H and u_i^h (for $i = 1, \ldots, p$) we now prove that (3.37) and (3.38) both hold.

To prove (3.37) first let $\underline{x} \in \Omega_i$ for $i \in c(1)$. Then

$$\begin{aligned} u^H + u_1^h + \cdots + u_p^h &= \sum_{n=1}^{n_c} u_n^H(\underline{x}) + \sum_{j=1}^{p} u_j^h(\underline{x}) \\ &= u_1^H(\underline{x}) + u_i^h(\underline{x}) \\ &= u_1^H(\underline{x}) + w_1^h(\underline{x}) \\ &= r_1^h \\ &= u^h. \end{aligned}$$

Now let $\underline{x} \in \Omega_i$ for $i \in c(m)$ for any $m \in \{2, \ldots, n_c\}$. Then

$$\begin{aligned} u^H + u_1^h + \cdots + u_p^h &= \sum_{n=1}^{n_c} u_n^H(\underline{x}) + \sum_{j=1}^{p} u_j^h(\underline{x}) \\ &= \sum_{n=1}^{m} \left(u_n^H(\underline{x}) + \sum_{j \in c(n)} u_j^h(\underline{x}) \right) \\ &= \sum_{n=1}^{m-1} \left(u_n^H(\underline{x}) + \sum_{j \in c(n)} u_j^h(\underline{x}) \right) + u_m^H(\underline{x}) + u_i^h(\underline{x}) \\ &= \sum_{n=1}^{m-1} \left(u_n^H(\underline{x}) + \sum_{j \in c(n)} u_j^h(\underline{x}) \right) + u_m^H(\underline{x}) + w_m^h(\underline{x}) \\ &= \sum_{n=1}^{m-1} \left(u_n^H(\underline{x}) + \sum_{j \in c(n)} u_j^h(\underline{x}) \right) + r_m^h(\underline{x}) \\ &= \sum_{n=1}^{m-1} \left(u_n^H(\underline{x}) + \sum_{j \in c(n)} u_j^h(\underline{x}) \right) + r_{m-1}^h(\underline{x}) - u_{m-1}^H(\underline{x}) - \sum_{j \in c(m-1)} u_j^h \\ &= \sum_{n=1}^{m-2} \left(u_n^H(\underline{x}) + \sum_{j \in c(n)} u_j^h(\underline{x}) \right) + r_{m-1}^h(\underline{x}) \\ &= \sum_{n=1}^{m-3} \left(u_n^H(\underline{x}) + \sum_{j \in c(n)} u_j^h(\underline{x}) \right) + r_{m-2}^h(\underline{x}) \\ &= : \\ &= \left(u_1^H(\underline{x}) + \sum_{j \in c(1)} u_j^h(\underline{x}) \right) + r_2^h(\underline{x}) \\ &= r_1^h \\ &= u^h. \end{aligned}$$

Finally, we observe that if $\underline{x}$ is on the boundary between two or more subdomains, then the above argument may be applied to the subdomain of the lowest color to show that $u^H + u_1^h + \cdots + u_p^h = u^h$ at this point too. (This argument uses the continuity of each w_m^h and the fact that $w_m^h(\underline{x}) = 0$ for each subdomain m whose boundary contains $\underline{x}$, except the one with the lowest color.)

To conclude the proof we now demonstrate that (3.38) also holds. In order to do this, first note that since $\cup_{m=1}^{n_c} c(m) = \{1, \ldots, p\}$ and $u^H = \sum_{m=1}^{n_c} u_m^H$,

$$\|u^H\|^2_{\mathcal{H}^1(\Omega)} + \|u_1^h\|^2_{\mathcal{H}^1(\Omega)} + \cdots + \|u_p^h\|^2_{\mathcal{H}^1(\Omega)} \le \sum_{m=1}^{n_c} \left(\|u_m^H\|^2_{\mathcal{H}^1(\Omega)} + \sum_{i \in c(m)} \|u_i^h\|^2_{\mathcal{H}^1(\Omega)} \right). \tag{3.39}$$

Hence, since n_c is assumed to be independent of h, H, and p, it is sufficient to show that

$$\|u_m^H\|^2_{\mathcal{H}^1(\Omega)} + \sum_{i \in c(m)} \|u_i^h\|^2_{\mathcal{H}^1(\Omega)} \le C\|u^h\|^2_{\mathcal{H}^1(\Omega)} \tag{3.40}$$

for some C which is independent of h, H, and p, and any $m \in \{1, \ldots, n_c\}$. Let the quantity on the left-hand side of (3.40) be S_m. Then

$$\begin{aligned}
S_m &= \|u_m^H\|^2_{\mathcal{H}^1(\Omega)} + \sum_{i \in c(m)} \|u_i^h\|^2_{\mathcal{H}^1(\hat{\Omega}_i)} \\
&\quad \text{(since } u_i^h \in \hat{\mathcal{W}}_i\text{)} \\
&\le C_4\|r_m^h\|^2_{\mathcal{H}^1(\Omega)} + \sum_{i \in c(m)} \|u_i^h\|^2_{\mathcal{H}^1(\hat{\Omega}_i)} \\
&\quad \text{(using (3.35))} \\
&\le C_4\|r_m^h\|^2_{\mathcal{H}^1(\Omega)} + C_1 \sum_{i \in c(m)} \left\{ \frac{1}{H^2}\|w_m^h\|^2_{\mathcal{L}^2(\Omega_i)} + |w_m^h|^2_{\mathcal{H}^1(\Omega_i)} \right\} \\
&\quad \text{(using Lemma 3.1)} \\
&\le C_4\|r_m^h\|^2_{\mathcal{H}^1(\Omega)} + C_1C_4^2 \sum_{i \in c(m)} \|r_m^h\|^2_{\mathcal{H}^1(\Omega_i)} \\
&\quad \text{(using (3.36))} \\
&= C_4\|r_m^h\|^2_{\mathcal{H}^1(\Omega)} + C_1C_4^2\|r_m^h\|^2_{\mathcal{H}^1(\Omega)} \\
&= C_5\|r_m^h\|^2_{\mathcal{H}^1(\Omega)}. \qquad (3.41)
\end{aligned}$$

Clearly if $m = 1$, then $r_m^h = u^h$ and we are done. Otherwise note that

$$\begin{aligned}
S_m &\le C_5\|r_m^h\|^2_{\mathcal{H}^1(\Omega)} \\
&\le C_5 \left(\|r_{m-1}^h\|^2_{\mathcal{H}^1(\Omega)} + S_{m-1} \right) \\
&\le C_5 \left(\|r_{m-1}^h\|^2_{\mathcal{H}^1(\Omega)} + C_5\|r_{m-1}^h\|^2_{\mathcal{H}^1(\Omega)} \right) \\
&\quad \text{(using the same argument as in (3.41) above)} \\
&= C_6\|r_{m-1}^h\|^2_{\mathcal{H}^1(\Omega)} \\
&\le C_7\|r_1^h\|^2_{\mathcal{H}^1(\Omega)} \\
&\quad \text{(repeating this argument } m-2 \text{ further times)} \\
&= C_7\|u^h\|^2_{\mathcal{H}^1(\Omega)},
\end{aligned}$$

as required. □

Having demonstrated that the splitting given by (3.13) is stable it is now a simple matter to invoke Theorem 2.1 (with ν_c replaced by $\nu_c + 1$ to take into account the coarse grid space $\mathcal{W}_0$) and the equivalence of the norm $\mathcal{A}(\cdot,\cdot)^{1/2}$ with the $\mathcal{H}^1$ norm to deduce that the corresponding AS preconditioner is optimal.

The results of this section show that, in two dimensions, it is possible to obtain an optimal two-level AS preconditioner with an overlap which contains $O(h^{-1})$ elements only, provided appropriate use is made of the mesh hierarchy. While the proofs developed here are fully general in terms of subdomain shapes and connectivity a simple, more regular example has also been included for illustrative purposes. In contrasting these results with more standard theoretical results (as in the following section), which require $O(h^{-2})$ elements in the overlap regions for optimality, a number of practical points should be noted. First, as described in [30], for example, the standard two-level AS method does not use a generous overlap in practice. Typically, two to four fine mesh layers are found to be most economical. It follows, therefore, that the weakly overlapping approach that we have analyzed will not generally be any less (or more) computationally expensive per iteration than standard two-level AS solvers. What our approach does offer, however, is the guarantee of optimality, which does not hold when only a fixed number of fine grid layers of overlap are used. Furthermore, the communication cost at each iteration is $O(h^{-1})$ in both cases, and the weakly overlapping approach has the added simplicity of not requiring any trade-off between cost per iteration (i.e., overlap size) and the total iteration count to be considered. Second, when the weakly overlapping approach is applied to an arbitrary decomposition of Ω, the fact that only a single layer of overlap is required at each mesh level makes its implementation extremely straightforward. This is arguably less complex than the implementation of a more standard approach where four (say) layers of elements are required in the overlap, which can be quite cumbersome to calculate on a geometrically complex decomposition.

4. Numerical examples.

In this section we present a small number of two-dimensional numerical examples which demonstrate the efficiency of the preconditioner introduced above. For these examples we make a slight modification to the preconditioner so as to allow the practical parallel generation of the partitioned hierarchical meshes that are required.

In this modified algorithm, once the coarse mesh $\mathcal{T}_0$ has been partitioned into the p nonoverlapping subdomains, Ω_i, p copies of it are made. Copy i is then refined only in $\tilde{\Omega}_{i,k}$ at level k of the refinement process. The continuous piecewise linear finite element spaces on the resulting meshes are then

$$\mathcal{U}_i = \mathcal{W}_0 \cup \tilde{\mathcal{W}}_i \tag{4.1}$$

for $i = 1, \ldots, p$. The following corollary follows immediately from Theorem 3.2.

Corollary 4.1. *Let the spaces $\mathcal{U}_i$ be given by* (4.1) *for $i = 1, \ldots, p$. Then*

$$\mathcal{W} = \mathcal{U}_1 + \cdots + \mathcal{U}_p \tag{4.2}$$

is also a stable decomposition.

The advantages of this approach are outlined in some detail in [4], where it is shown that parallel adaptive mesh generation may be achieved in a well load-balanced manner. The main disadvantage is that one is effectively completing a coarse-mesh solve as part of *each* subspace correction (i.e., p times per iteration) rather than once per iteration. However, in many practical parallel codes (e.g., [22]) the coarse-grid solve is completed sequentially on a single processor anyway, so the overhead of repeating it on all p processors simultaneously is not necessarily that great.

The other minor practical modification that we have made to the preconditioner outlined in the previous section comes from the use of transition (sometimes known as "green" [29]) elements in our meshes in order to keep them conforming. In Figure 1 it may be observed that there are a number of "slave" nodes in the nonuniformly refined mesh which cause the mesh shown to be nonconforming. The solution values at these nodes are not free: they are determined by the nodal values at the ends of the edges on which the slave nodes lie. For a practical implementation it turns out to be much simpler to allow the solution values at these nodes to be free by bisecting the element on the unrefined side of the edge that has the "hanging" node on it. This is the approach that is used in the examples below.

For the first two test problems considered, sequences of uniformly refined meshes, $\mathcal{T}_k$, have been used.

Problem 4.1.

$$\begin{aligned} -\underline{\nabla} \cdot (\underline{\nabla} u) &= f \quad \forall \underline{x} \in \Omega \equiv (0,1) \times (0,1), \\ u &= g \quad \forall \underline{x} \in \partial\Omega. \end{aligned}$$

Problem 4.2.

$$\begin{aligned} -\underline{\nabla} \cdot \left(\begin{pmatrix} 10^2 & 0 \\ 0 & 1 \end{pmatrix} \underline{\nabla} u \right) &= f \quad \forall \underline{x} \in \Omega \equiv (0,1) \times (0,1), \\ u &= g \quad \forall \underline{x} \in \partial\Omega. \end{aligned}$$

In each case f has been chosen to permit the exact solution $u = g$ for a quadratic choice of g.

Tables 1 and 2 show the number of iterations required by both the conventional optimal (as defined by (2.10) and Theorem 2.2) and the new two-level AS preconditioners in order to reduce the 2-norm of the initial residual by a factor of 10^6 for these two problems, respectively. In each case a 256-element coarse mesh has been used and the final fine grids have between 4096 and 1,048,576 elements. The number of subdomains goes from 2 to 16, and for the purposes of these comparisons the local solves on each subproblem are exact (see below for a discussion of this).

Inspection of these tables shows that there is very little to choose between the number of iterations required by the conventional optimal AS preconditioner, with $O(h^{-2})$ elements in the overlap regions, and the new preconditioner, with just $O(h^{-1})$ elements in the overlap regions. This is an important observation since it indicates that the splitting constant, C, is of approximately the same size for the new splitting as for the conventional two-level splitting with a generous overlap. Indeed, it is the significance of this observation that motivates our comparison between the two approaches. Full details of the practical parallel implementation of the preconditioners is beyond the scope of this paper (see [5] for a complete description of our parallel implementation). Nevertheless, the relative timings when using one subdomain per processor are also included in Tables 1 and 2 in order to illustrate the advantages of the weakly overlapping approach. It is noticeable that the significant reduction in the size of the subproblems that must be solved at each iteration of the new algorithm, due to the smaller number of elements in the overlap region, clearly leads to a reduced parallel solution time. Not surprisingly, this reduction becomes more significant the more the mesh is refined.

Table 1

The number of iterations required to reduce the 2-norm of the residual by a factor of 10^6 *using the two different AS preconditioners when solving Problem* 4.1 *using piecewise linear finite elements. Column* 4 *shows the relative speed of the new versus the conventional optimal two-level preconditioner.*

Fine mesh	Conventional AS	New AS	CPU ratio	p
4096	6	6	1.0	2
16384	6	6	0.85	
65536	6	6	0.57	
262144	6	6	0.47	
1048576	6	6	0.35	
4096	8	9	1.0	4
16384	8	8	0.65	
65536	8	8	0.56	
262144	8	7	0.42	
1048576	8	7	0.27	
4096	13	12	0.88	8
16384	13	12	0.68	
65536	12	12	0.52	
262144	11	11	0.36	
1048576	11	11	0.29	
4096	14	14	0.73	16
16384	14	13	0.58	
65536	14	13	0.44	
262144	14	12	0.30	
1048576	13	12	0.25	

Table 2

The number of iterations required to reduce the 2-norm of the residual by a factor of 10^6 *using the two different AS preconditioners when solving Problem* 4.2 *using piecewise linear finite elements. Column* 4 *shows the relative speed of the new versus the conventional optimal two-level preconditioner.*

Fine mesh	Conventional AS	New AS	CPU ratio	p
4096	7	8	1.0	2
16384	7	8	0.82	
65536	7	8	0.69	
262144	7	8	0.49	
1048576	7	8	0.46	
4096	11	12	1.0	4
16384	12	13	0.74	
65536	12	13	0.66	
262144	11	12	0.50	
1048576	10	12	0.30	
4096	14	15	0.90	8
16384	14	16	0.67	
65536	13	16	0.65	
262144	13	17	0.53	
1048576	13	17	0.39	
4096	18	19	0.71	16
16384	18	19	0.60	
65536	19	20	0.58	
262144	18	21	0.41	
1048576	18	21	0.31	

It should also be noted at this point that the results of Tables 1 and 2 do show an increase in the number of iterations required as p, the number of subdomains, is increased (for both versions of the algorithm). This may be accounted for by the fact that n_c increases as p increases from 2 to 8, and that for the relatively small values of p considered here (up to $p = 16$) the asymptotic limit has not yet been reached, even though n_c is at most 5 in all cases. It may be observed, however, that the growth in the number of iterations as p increases is already beginning to slow down as p goes from 8 to 16. Also, the iteration counts for Problem 4.2 are greater than those for corresponding solutions of Problem 4.1. This may be accounted for in the theory of the previous section by noting that the norm equivalence $\mathcal{A}(\cdot,\cdot)^{1/2} \sim \mathcal{H}^1$ will involve a larger ratio of constants in the upper and lower bounds for Problem 4.2 than Problem 4.1.

In practice it is not necessary to solve the subproblems involving the matrices A_i in (2.10) exactly in order to maintain the optimality of the preconditioner. This is discussed in some detail elsewhere (see, for example, [11, 36]) and so we merely comment that our numerical experiments suggest that reducing the residual by a factor of between 10^1 and 10^2 when "solving" these subproblems appears to provide a good balance between allowing a small increase in the total number of iterations and keeping the cost of each iteration as low as possible. For the examples in this section we have applied a preconditioned CG solver with an algebraic preconditioner based upon an incomplete factorization of A_i (see [6, 7], for example).

We conclude this section by demonstrating the effectiveness of the preconditioner for a locally, rather than uniformly, refined mesh.

Problem 4.3.

$$\begin{aligned} -\underline{\nabla}\cdot(\underline{\nabla} u) &= f \quad \forall \underline{x} \in \Omega \equiv (0,1)\times(0,1), \\ u &= g \quad \forall \underline{x} \in \partial\Omega. \end{aligned}$$

In this case f and g are now chosen so that the analytic solution is given by

$$u = (1-(2x_1-1)^{100})(1-(2x_2-1)^{100}) \quad \forall \underline{x} \in \Omega. \tag{4.3}$$

Note that this solution is unity in the interior of Ω but tends to zero very rapidly in a thin layer (of width ≈ 0.02) near to the boundary, allowing the Dirichlet condition $u = 0$ to be satisfied throughout $\partial\Omega$.

When solving this problem we again use a coarse triangulation which contains just 256 elements divided into 2, 4, 8, and 16 subdomains. Note that for this example it is only appropriate to refine the mesh in the boundary layer, where the solution changes from one to zero, and four such sets of results are presented in Table 3 corresponding to 5, 6, 7, and 8 levels of refinement in the layer. (For the purposes of these tests the local refinement of an element is triggered when the exact interpolation error on that element is greater than 10^{-10} (2-norm) unless the element is already at the maximum refinement level.) In each case the coarse mesh $\mathcal{T}_0$ has been partitioned in such a way that the number of elements in the final mesh, $\mathcal{T}_J$, in each subregion is approximately equal. In the cases where $p = 8$ and $p = 16$ this means that there are differing numbers of coarse elements in each subregion (between 8 and 40 when $p = 16$, for example); the partitions into 2 or 4 may be computed more simply, however, using the symmetry of the problem and the coarse mesh (see Figure 7, for example). As with Tables 1 and 2, the results of Table 3 clearly show independence from h for a sufficient level of refinement; however, the independence from p is not yet evident for

TABLE 3
The number of iterations required to reduce the 2*-norm of the residual by a factor of* 10^6 *using the new AS preconditioner when solving Problem* 4.3 *using piecewise linear finite elements.*

Refinements (elements/vertices)	$p = 2$	$p = 4$	$p = 8$	$p = 16$
5 (48680/24853)	7	9	13	18
6 (145088/73569)	7	10	15	18
7 (362272/184685)	8	10	16	18
8 (543632/275913)	8	10	16	19

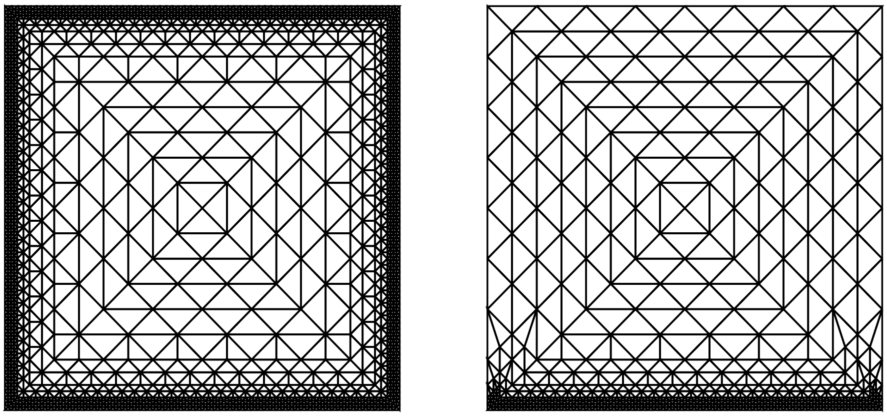


FIG. 7. *The overall refined mesh (left) and a typical mesh on one processor when* $p = 4$ *(right) for Problem* 4.3*. Here the coarse mesh contains* 256 *elements and, for this illustration, at most three levels of refinement are permitted.*

the small values considered (up to $p = 16$), although we again see a reduction in the growth of the number of iterations as p increases.

As well as verifying our theoretical results in the case of a locally refined mesh, the last example also illustrates the potential of the technique for use within truly adaptive algorithms based upon a posteriori error estimates and local refinement (see [1, 35], for example). Such algorithms are not without their difficulties, however, since undertaking adaptivity in parallel presents challenging dynamic load-balancing problems (e.g., [32]) which clearly cannot be dealt with using the simple a priori partitioning approach taken above. For one possible means of overcoming these problems we refer the reader to [4], where a paradigm is presented in which the a posteriori error estimates are themselves used to assist with load-balancing. We also note the additional theoretical issue of selecting an appropriate definition for h, the mesh size parameter, for locally refined meshes. While this does not affect the theoretical results presented in this paper, it does have an important bearing on quantitative overhead estimates in terms of the problem size.

5. Extension to three dimensions. So far, for reasons of clarity and simplicity, our discussion has been restricted to problems of the form (1.1)–(1.3) in two dimensions. In this section we demonstrate that both the theoretical results and the practical realization may be successfully generalized to three dimensions. Again we consider problems of the form (1.1)–(1.3), but now we allow $\Omega \subset \mathbb{R}^3$. We will assume that Ω may be covered by a set of tetrahedra, $\mathcal{T}_0$, consisting of N_0 tetrahedral elements $\tau_j^{(0)}$ $(j = 1, \ldots, N_0)$. In order to refine a tetrahedron we use the standard

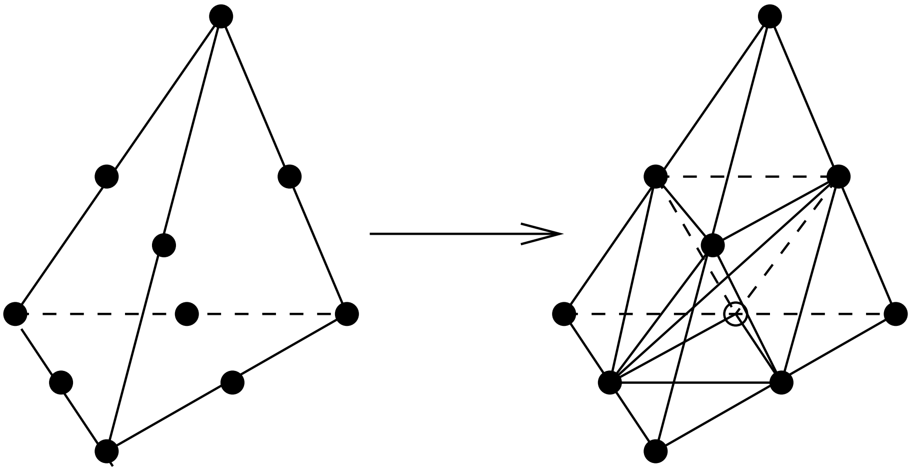


FIG. 8. *Refinement of a tetrahedron into eight children by bisecting each edge.*

approach of bisecting each edge to produce eight children, as shown in Figure 8. Note that although the children are not generally geometrically similar, their aspect ratios are always bounded independently of h. Full details of the refinement algorithm may be found in [31].

Using the above refinement strategy it is possible to generate a family of three-dimensional triangulations, $\mathcal{T}_0, \dots, \mathcal{T}_J$, satisfying the same restrictions as the two-dimensional family introduced in section 3 (with the obvious modification to condition 1(b) for eight children). From this, the spaces $\mathcal{W}$, $\mathcal{W}_{i,k,0}$ $(k = 1, \dots, J)$, and $\hat{\mathcal{W}}_i$, may be defined in the corresponding manner. Algorithm 3.1 therefore also generalizes directly to three dimensions. In fact, it is straightforward to see that, with the exception of the proof of Lemma 3.1, all of the remaining theory in section 3 now generalizes immediately from two to three dimensions. Hence, in order to prove that the corresponding weakly overlapping preconditioner is optimal for tetrahedral meshes in three dimensions, it is sufficient to prove Lemma 3.1 for this case. Such a proof is provided in the appendix.

It should be noted that in three dimensions the number of elements in the overlap regions when using the weakly overlapping algorithm is $O(h^{-2})$ as opposed to $O(h^{-3})$ when a standard generous overlap is used. Similarly, for practical (nonoptimal) two-level AS algorithms with a fixed number of fine mesh layers in the overlap region, the number of elements in the overlap is also $O(h^{-2})$ in three dimensions. We again see therefore that the weakly overlapping approach has approximately the same cost (and communication overhead) per iteration as the practical small overlap two-level algorithms but with the advantage of guaranteed optimality.

Having discussed the generalization of the main theoretical result of this paper to three dimensions we now illustrate this with some further numerical examples. These examples have been chosen to correspond to the two-dimensional examples already considered in the previous section.

PROBLEM 5.1.

$$\begin{aligned} -\underline{\nabla} \cdot (\underline{\nabla} u) &= f \quad \forall \underline{x} \in \Omega \equiv (0,1) \times (0,1) \times (0,1), \\ u &= g \quad \forall \underline{x} \in \partial\Omega. \end{aligned}$$

PROBLEM 5.2.

$$\begin{aligned} -\underline{\nabla} \cdot \left(\begin{pmatrix} 10^2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \underline{\nabla} u \right) &= f \quad \forall \underline{x} \in \Omega \equiv (0,1) \times (0,1) \times (0,1), \\ u &= g \quad \forall \underline{x} \in \partial\Omega. \end{aligned}$$

Here f has again been chosen in each case so as to permit the exact solution $u = g$ throughout Ω.

PROBLEM 5.3.

$$\begin{aligned} -\underline{\nabla} \cdot (\underline{\nabla} u) &= f \quad \forall \underline{x} \in \Omega \equiv (0,1) \times (0,1) \times (0,1), \\ u &= g \quad \forall \underline{x} \in \partial\Omega. \end{aligned}$$

In Problem 5.3, f and g are now chosen so that the analytic solution is given by

$$u = (1 - (2x_1 - 1)^{70})(1 - (2x_2 - 1)^{70})(1 - (2x_3 - 1)^{70}) \quad \forall \underline{x} \in \Omega. \tag{5.1}$$

Table 4 presents iteration counts for Problems 5.1 and 5.2 using the weakly overlapping preconditioner on a coarse grid of 384 elements using between 1 and 4 levels of uniform mesh refinement for a typical set of partitions into 2, 4, 8, and 16 subdomains. It may be observed that the number of iterations is indeed independent of h. Furthermore, as p increases the number of iterations appears to have almost stopped growing by the time $p = 16$.

TABLE 4

The number of iterations required to reduce the 2*-norm of the residual by a factor of* 10^6 *using the weakly overlapping AS preconditioner when solving Problems* 5.1 *and* 5.2 *using piecewise linear finite elements.*

Fine mesh size	Problem 5.1	Problem 5.2	Procs.
3072	7	8	2
24576	8	9	
196608	10	10	
1572864	10	11	
3072	13	13	4
24576	14	15	
196608	15	16	
1572864	16	17	
3072	13	13	8
24576	17	18	
196608	17	18	
1572864	16	17	
3072	12	13	16
24576	20	21	
196608	18	20	
1572864	17	17	

To solve Problem 5.3 local h-refinement has been used with a slightly larger coarse grid, containing 3072 tetrahedral elements. As in two dimensions the precise choice of partition of this grid is important in terms of both the final load balance and the total numbers of iterations required. Table 5 presents iteration counts for one such set of partitions into 2, 4, 8, and 16 subdomains. Again we see that the number of iterations is almost independent of h already (after just 4 levels of refinement). The partition into 16 for this problem is probably far from optimal (since simple recursive coordinate bisection [34] was used), which may well account for the noticeable jump in iterations between $p = 8$ and $p = 16$. Nevertheless, these simple numerical experiments do show that the weakly overlapping technique can be effectively implemented in three dimensions as well as two.

TABLE 5

The number of iterations required to reduce the 2*-norm of the residual by a factor of* 10^6 *using the weakly overlapping AS preconditioner when solving Problem* 5.3 *using piecewise linear finite elements.*

Refinements (elements/vertices)	$p = 2$	$p = 4$	$p = 8$	$p = 16$
1 (20832/4403)	11	15	19	25
2 (198816/22237)	14	17	20	27
3 (499123/100708)	15	18	21	28
4 (2139159/429435)	17	21	24	30

6. Discussion.

In this paper we have presented a two-level AS preconditioner based upon a weakly overlapping domain decomposition of a nested sequence of meshes, $\mathcal{T}_0, \ldots, \mathcal{T}_J$. Such a decomposition involves just a single element of overlap at each level of the mesh hierarchy and so requires only $O(h^{-1})$ overlapping elements in total in two dimensions (and $O(h^{-2})$ in three dimensions). It has been demonstrated that, when combined with the full coarse mesh space $\mathcal{W}_0$, the splitting that is induced by this decomposition is stable and therefore leads to an optimal preconditioner. Although the work presented here is for an AS algorithm there is no reason why the same splitting should not also be used in a multiplicative manner. The resulting preconditioner will again be optimal. The extension of this approach to a multilevel algorithm is less interesting, however, since this leads to a standard multilevel Schwarz scheme (see [30] for example), with subdomains chosen to be the same at each level and with the overlap chosen to be minimal at each level. Such schemes are already known to be optimal.

Although the paper concentrates mainly on test problems of the form (1.1)–(1.3) in two dimensions, all aspects of the theory in sections 2 and 3 are extended to three-dimensional problems using nested tetrahedral meshes in section 5. Supporting numerical experiments in both two and three dimensions are also provided. Furthermore, the simplifying assumption of zero Dirichlet boundary conditions throughout $\partial\Omega_E$ may also be dropped, so as to include other Dirichlet conditions or even mixed conditions, without significant modification. Further extensions may also be made within the theoretical framework described by permitting the initial mesh, $\mathcal{T}_0$, to have unequal element sizes. This would therefore allow the use of a different value for H, the size of the elements of the coarse mesh, on each subdomain.

Appendix. Proof of Lemma 3.1 in the three-dimensional case.

Proof. First we introduce the following change of variables:

$$x = H\,r, \qquad y = H\,s, \qquad z = H\,t\,; \qquad (x, y, z) \in \Omega_i. \tag{A.1}$$

Under this transformation the domain Ω_i is the image of a domain Ω_i' whose geometric properties are independent of H in the (r, s, t) plane. Furthermore

$$\begin{aligned}\frac{1}{H^2}\|w_i^h(x,y,z)\|^2_{\mathcal{L}^2(\Omega_i)} &+ |w_i^h(x,y,z)|^2_{\mathcal{H}^1(\Omega_i)}\\ &= H\left(\|w_i^h(r,s,t)\|^2_{\mathcal{L}^2(\Omega_i')} + |w_i^h(r,s,t)|^2_{\mathcal{H}^1(\Omega_i')}\right),\end{aligned} \tag{A.2}$$

and we may define by $Q'_{i,k}$ the projection in the (r,s,t) variables which corresponds to $Q_{i,k}$. From [9] and [26] it follows that there exists $C_2>0$, which is independent of h, H, and p, such that

$$\begin{aligned}
&\frac{1}{H^2}\sum_{k=0}^{J}4^k\|v_{i,k}^h\|^2_{\mathcal{L}^2(\Omega_i)}\\
&\quad=\frac{1}{H^2}\left(\|Q_{i,0}w_i^h(x,y,z)\|^2_{\mathcal{L}^2(\Omega_i)}+\sum_{k=1}^{J}4^k\|(Q_{i,k}-Q_{i,k-1})w_i^h(x,y,z)\|^2_{\mathcal{L}^2(\Omega_i)}\right)\\
&\quad=H\left(\|Q'_{i,0}w_i^h(r,s,t)\|^2_{\mathcal{L}^2(\Omega'_i)}+\sum_{k=1}^{J}4^k\|(Q'_{i,k}-Q'_{i,k-1})w_i^h(r,s,t)\|^2_{\mathcal{L}^2(\Omega'_i)}\right)\\
&\quad\le HC_2\|w_i^h(r,s,t)\|^2_{\mathcal{H}^1(\Omega'_i)}\\
&\quad=C_2\left(\frac{1}{H^2}\|w_i^h(x,y,z)\|^2_{\mathcal{L}^2(\Omega_i)}+|w_i^h(x,y,z)|^2_{\mathcal{H}^1(\Omega_i)}\right).
\end{aligned}\tag{A.3}$$

A second inequality that we require comes from [26], where it is shown that there exists $C_3>0$, which is independent of h, H, and p, such that

$$\begin{aligned}
\|\hat v_i^h(x,y,z)\|^2_{\mathcal{H}^1(\hat\Omega_i)}&=H^3\|\hat v_i^h(r,s,t)\|^2_{\mathcal{L}^2(\hat\Omega'_i)}+H|\hat v_i^h(r,s,t)|^2_{\mathcal{H}^1(\hat\Omega'_i)}\\
&\le HC_3\inf_{\substack{\hat v_i^h=\hat\xi_0+\cdots+\hat\xi_J\\(\hat\xi_k\in\hat{\mathcal{W}}'_{i,k})}}\sum_{k=0}^{J}4^k\|\hat\xi_k\|^2_{\mathcal{L}^2(\hat\Omega'_i)},
\end{aligned}\tag{A.4}$$

where $\hat{\mathcal{W}}'_{i,k}$ is the space which corresponds to $\hat{\mathcal{W}}_{i,k}$ with the change of variables (A.1). From this, along with (3.25) and (A.3), it follows that

$$\begin{aligned}
\|\hat v_i^h(x,y,z)\|^2_{\mathcal{H}^1(\hat\Omega_i)}&\le HC_3\sum_{k=0}^{J}4^k\|\hat v_{i,k}^h\|^2_{\mathcal{L}^2(\hat\Omega'_i)}\\
&=\frac{C_3}{H^2}\sum_{k=0}^{J}4^k\|\hat v_{i,k}^h\|^2_{\mathcal{L}^2(\hat\Omega_{i,k})}\\
&\le\frac{C_0C_3}{H^2}\sum_{k=0}^{J}4^k\|v_{i,k}^h\|^2_{\mathcal{L}^2(\Omega_i)}\\
&\le C_0C_2C_3\left(\frac{1}{H^2}\|w_i^h\|^2_{\mathcal{L}^2(\Omega_i)}+|w_i^h|^2_{\mathcal{H}^1(\Omega_i)}\right),
\end{aligned}\tag{A.5}$$

as required. □

REFERENCES

[1] M. Ainsworth and J. T. Oden, *A Posteriori Error Estimation in Finite Element Analysis*, Wiley, Chichester, 2000.

[2] S. F. Ashby, T. A. Manteuffel, and P. E. Saylor, *A taxonomy for conjugate gradient methods*, SIAM J. Numer. Anal, 27 (1990), pp. 1542–1568.

[3] R. E. Bank, T. F. Dupont, and H. Yserentant, *The hierarchical basis multigrid method*, Numer. Math., 52 (1988), pp. 427–458.

[4] R. E. Bank and M. Holst, *A new paradigm for parallel adaptive meshing algorithms*, SIAM J. Sci. Comput., 22 (2000), pp. 1411–1443.

[5] R. E. Bank and P. K. Jimack, *A new parallel domain decomposition method for the adaptive finite element solution of elliptic partial differential equations*, Concurrency and Computation: Practice and Experience, 13 (2001), pp. 327–350.

[6] R. E. Bank and R. K. Smith, *The incomplete factorization multigraph algorithm*, SIAM J. Sci. Comput., 20 (1999), pp. 1349–1364.

[7] R. E. Bank and C. Wagner, *Multilevel ILU decomposition*, Numer. Math., 82 (1999), pp. 543–576.

[8] R. E. Bank and J. Xu, *A hierarchical basis multigrid method for unstructured grids*, in Fast Solvers for Flow Problems, Notes Numer. Fluid Mech. 49, W. Hackbusch and G. Wittum, eds., Vieweg, Braunschweig, 1995, pp. 1–13.

[9] V. Bornemann and H. Yserentant, *A basic norm equivalence for the theory of multilevel methods*, Numer. Math., 64 (1993), pp. 455–476.

[10] J. Bramble, J. Pasciak, and J. Xu, *Parallel multilevel preconditioners*, Math. Comp., 55 (1990), pp. 1–21.

[11] T. Chan and T. Matthew, *Domain decomposition algorithms*, in Acta Numerica, Cambridge University Press, Cambridge, UK, 1994, pp. 61–143.

[12] T. Chan, B. Smith, and J. Zou, *Overlapping Schwarz methods on unstructured meshes using non-matching coarse grids*, Numer. Math., 73 (1995), pp. 149–167.

[13] M. Dryja and O. B. Widlund, *Some domain decomposition algorithms for elliptic problems*, in Iterative Methods for Large Linear Systems, T. F. Chan, R. Glowinski, J. Périaux, and O. B. Widlund, eds., Academic Press, Boston, 1990, pp. 273–291.

[14] M. Dryja and O. B. Widlund, *Towards a unified theory of domain decomposition algorithms for elliptic problems*, in Proceedings of the Third International Symposium on Domain Decomposition Methods for Partial Differential Equations, SIAM, Philadelphia, 1990, pp. 3–21.

[15] M. Dryja and O. B. Widlund, *Multilevel additive methods for elliptic finite element problems*, in Proceedings of the Sixth GAMM-Seminar Kiel, W. Hackbusch and G. Wittum, eds., Vieweg, Braunschweig, 1991, pp. 58–69.

[16] C. Farhat, J. Mandel, and F.-X. Roux, *Optimal convergence properties of the FETI domain decomposition method*, Comput. Methods Appl. Mech. Engrg., 115 (1994), pp. 365–385.

[17] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed., Johns Hopkins University Press, Baltimore, MD, 1996.

[18] M. Griebel and P. Oswald, *On additive Schwarz preconditioners for sparse grid discretizations*, Numer. Math., 66 (1994), pp. 449–463.

[19] G. Haase, U. Langer, A. Meyer, and S. V. Nepomnyaschikh, *Hierarchical extension operators and local multigrid methods in domain decomposition preconditioners*, East-West J. Numer. Math., 2 (1994), pp. 172–193.

[20] G. Haase and S. V. Nepomnyaschikh, *Explicit extension operators on hierarchical grids*, East-West J. Numer. Math., 5 (1997), pp. 231–248.

[21] M. E. Hayder, D. E. Keyes, and P. Mehrotra, *A comparison of PETSc library and HPF implementations of an archetypal PDE computation*, Adv. Engrg. Software, 29 (1998), pp. 415–424.

[22] D. C. Hodgson and P. K. Jimack, *A domain decomposition preconditioner for a parallel finite element solver on distributed unstructured grids*, Parallel Comput., 23 (1997), pp. 1157–1181.

[23] C. Johnson, *Numerical Solution of Partial Differential Equations by the Finite Element Method*, Cambridge University Press, Cambridge, UK, 1987.

[24] P. LeTallec, *Domain decomposition methods in computational mechanics*, Comput. Mech. Adv., 2 (1994), pp. 121–220.

[25] A. M. Matsokin and S. V. Nepomnyaschikh, *Schwarz alternating method in subspaces*, Soviet Math., 29 (1985), pp. 78–84.

[26] P. Oswald, *Multilevel Finite Element Approximation: Theory and Applications*, B. G. Teubner, Stuttgart, 1994.

[27] A. Quarteroni and A. Valli, *Domain Decomposition Methods for Partial Differential Equations*, Clarendon Press, Oxford, New York, 1999.

[28] W. Rachowicz, *An overlapping domain decomposition preconditioner for an anisotropic h-adaptive finite element method*, Comput. Methods Appl. Mech. Engrg., 127 (1995), pp. 269–292.

[29] M.-C. Rivara, *A grid generator based on 4-triangles conforming mesh refinement algorithm*, Internat. J. Numer. Methods Engrg., 24 (1987), pp. 1343–1354.

[30] B. Smith, P. Bjorstad, and W. Gropp, *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge University Press, Cambridge, UK, 1996.

[31] W. Speares and M. Berzins, *A 3-d unstructured mesh adaptation algorithm for time-dependent shock dominated problems*, Internat. J. Numer. Methods Fluids, 25 (1997), pp. 81–104.

[32] N. Touheed, P. Selwood, P. K. Jimack, and M. Berzins, *A comparison of some dynamic load-balancing algorithms for a parallel adaptive flow solver*, Parallel Comput., 26 (2000), pp. 1535–1554.

[33] O. B. Widlund, *Some Schwarz methods for symmetric and nonsymmetric elliptic problems*, in Proceedings of the Fifth International Symposium on Domain Decomposition Methods for Partial Differential Equations, SIAM, Philadelphia, 1992, pp. 19–36.

[34] R. D. Williams, *Performance of dynamic load balancing for unstructured mesh calculations*, Concurrency: Practice and Experience, 3 (1991), pp. 457–481.

[35] R. Verfürth, *A Review of A Posteriori Error Estimation and Adaptive Mesh Refinement Techniques*, Wiley-Teubner, Chichester, Stuttgart, 1996.

[36] J. Xu, *Iterative methods by space decomposition and subspace correction*, SIAM Rev., 34 (1992), pp. 581–613.

[37] J. Xu and J. Zou, *Some nonoverlapping domain decomposition methods*, SIAM Rev., 40 (1998), pp. 857–914.

[38] X. Zhang, *Multilevel Schwarz methods*, Numer. Math., 63 (1992), pp. 521–539.

SIAM J. SCI. COMPUT.
Vol. 23, No. 6, pp. 1842–1860

A TRUST-REGION APPROACH TO THE REGULARIZATION OF LARGE-SCALE DISCRETE FORMS OF ILL-POSED PROBLEMS[*]

MARIELBA ROJAS[†] AND DANNY C. SORENSEN[‡]

Abstract. We consider large-scale least squares problems where the coefficient matrix comes from the discretization of an operator in an ill-posed problem, and the right-hand side contains noise. Special techniques known as regularization methods are needed to treat these problems in order to control the effect of the noise on the solution. We pose the regularization problem as a quadratically constrained least squares problem. This formulation is equivalent to Tikhonov regularization, and we note that it is also a special case of the trust-region subproblem from optimization. We analyze the trust-region subproblem in the regularization case and we consider the nontrivial extensions of a recently developed method for general large-scale subproblems that will allow us to handle this case. The method relies on matrix-vector products only, has low and fixed storage requirements, and can handle the singularities arising in ill-posed problems. We present numerical results on test problems, on an inverse interpolation problem with field data, and on a model seismic inversion problem with field data.

Key words. regularization, constrained quadratic optimization, trust region, Lanczos method, ill-posed problems, inverse problems, seismic inversion

AMS subject classifications. 86A22, 65K10, 90C06

PII. S1064827500378167

1. Introduction.

Discrete forms of ill-posed problems arise when we discretize the continuous operator in an ill-posed problem and introduce experimental data contaminated by noise. One of the main sources of ill-posed problems are inverse problems, where we want to determine the internal structure of a system from the observed behavior of the system. Inverse problems arise in many important applications such as image processing [2], seismic inversion [39], and medical and seismic tomography [30], [32]. Discrete forms of ill-posed problems are usually formulated as linear systems or least squares problems. The focus of this paper is the numerical treatment of large-scale discrete forms of ill-posed least squares problems.

We are interested in recovering x_{LS}, the minimum norm solution of

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|, \tag{1}$$

where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, and $m \geq n$. Throughout the paper we assume that A comes from the discretization of a continuous operator in an ill-posed problem, and instead of the *exact* data vector b, only a *perturbed* data vector $\bar{b}$ is available. Specifically, we regard $\bar{b}$ as $\bar{b} = b + s$, where s is a random vector of uncorrelated

[*] Received by the editors September 15, 2000; accepted for publication (in revised form) September 29, 2001; published electronically February 20, 2002.

http://www.siam.org/journals/sisc/23-6/37816.html

[†] CERFACS, 42 avenue Gaspard Coriolis, 31057 Toulouse CEDEX 1, France. Current address: Department of Mathematics, Wake Forest University, 302 Calloway Hall, Winston-Salem, NC 27109 (mrojas@math.wfu.edu). This author was supported in part by NSF cooperative agreement CCR-9120008.

[‡] Department of Computational and Applied Mathematics, Rice University, 6100 Main St., Houston, TX 77005-1892 (sorensen@caam.rice.edu). This author was supported in part by NSF grant CCR-9988393 and in part by the Los Alamos National Laboratory Computer Science Institute (LACSI) through LANL contract 03891-99-23 as part of the prime contract (W-7405-ENG-36) between the Department of Energy and the Regents of the University of California.

noise. The norm is the Euclidean norm throughout the paper, and it will be denoted by $\|\cdot\|$.

We will assume that the matrix A is large and might not be available explicitly but that we can compute the action of A and A^T on vectors of the appropriate dimensions. We will also assume that errors in A, due to discretization or finite-precision representation, are small in comparison to the noise in $\bar{b}$. Finally, we will not assume any particular structure for A.

Given the fact that only $\bar{b}$ is available, we could formulate the problem

$$\min_{x \in \mathbb{R}^n} \|Ax - \bar{b}\| \tag{2}$$

and use its minimum norm solution, denoted by $\bar{x}_{LS}$, to approximate x_{LS}. Unfortunately, as we shall see, the two solutions might differ considerably.

If we use a reasonably accurate discretization to obtain A, this matrix will be highly ill-conditioned with a singular spectrum that decays to zero gradually, a large cluster of small singular values, and high-frequency components of the singular vectors associated with small singular values. If, in addition, the discrete Picard condition [16] holds, we will have that the expansion coefficients of the exact data vector b in the left singular vectors basis decay to zero faster than the singular values of A, while the expansion coefficients of the noise vector s remain constant. Therefore, those components of $\bar{x}_{LS}$ corresponding to small singular values are magnified by the noise.

As a consequence of the ill conditioning of the matrix A and the presence of noise in the right-hand side, standard numerical algebra methods such as the ones discussed in [3], [13, Chap. 5], and [26] applied to problem (2) produce meaningless solutions with very large norm. Therefore, to solve these problems, we need special techniques known as *regularization* or *smoothing methods*. These methods aim to recover information about the desired solution of the unknown problem with exact data from the solution of a better conditioned problem that is related to the problem with noisy data but incorporates additional information about the desired solution. The formulation of the new problem involves a special parameter known as the *regularization parameter*, used to control the effect of the noise on the solution. The conditioning of the new problem depends on the choice of the regularization parameter. Excellent surveys on regularization methods can be found, for example, in [15], [20] and more recently in [31].

While there are many alternatives for solving small- to medium-scale problems, this is not the case in the large-scale setting. However, in recent years interesting methods for large-scale ill-posed problems have been proposed. Among these are Golub and von Matt [14], Björck, Grimme, and Van Dooren [4], Calvetti, Reichel, and Zhang [8], Rojas, Santos, and Sorensen [35], as well as several variants of the conjugate gradient method on the normal equations (CGLS), including the use of preconditioners chosen according to the structure of the problem [22], [24], [29]. In spite of these developments, the efficient solution of large-scale discrete forms of ill-posed problems remains a challenge.

In practice, the most common approach is to apply the conjugate gradient method to the normal equations associated with problem (2), taking advantage of what seems to be an intrinsic regularization property of this method. It has been observed that, at early stages, CGLS generates iterates with components in the direction of right singular vectors associated with *large* singular values, while components associated with small singular values come into play at later stages. This observation leads to the heuristic that the number of iterations acts as a regularization parameter.

Thus, we could compute a regularized solution by stopping the iteration before the unwanted components contaminate the current approximation. The success of this approach depends, in the first place, on the reliability of the heuristic and, second, on accurately determining when to stop the iteration, which is a difficult problem in itself, and most practical termination strategies rely on visual inspection. We are not aware of any systematic termination criterion for this approach. Alternatively, we could use CGLS on the Tikhonov regularization problem, discussed in the next section, which can be formulated as a damped least squares problem. The success of this approach depends on a good choice of the damping parameter, and also on the availability of a preconditioner, and good general preconditioners have not emerged yet. The approach we propose here does not depend on either a heuristic or a preconditioner.

In this paper, we formulate the regularization problem as a quadratically constrained least squares problem. It is well known that this approach is equivalent to Tikhonov regularization (cf. [10] and the references therein), and we also observe that the problem is a special case of the problem of minimizing a quadratic subject to a quadratic constraint, which is known in optimization as the trust-region subproblem arising in trust-region methods (see also [3, sections 5.3 and 9.2.3]). The connection between the trust-region subproblem and the regularization problem is well known, but the specific nature of the numerical difficulties for solving the regularization problem as a trust-region subproblem was first studied extensively in [34]. We discuss the properties of the trust-region subproblem in the regularization case and apply the recently developed method LSTRS [35] for the large-scale trust-region subproblem to the regularization of discrete forms of ill-posed problems from a variety of applications. The method relies on matrix-vector products only, has low and fixed storage requirements and robust stopping criteria, and computes both a solution and the corresponding Tikhonov regularization parameter. Moreover, LSTRS can efficiently handle the high-degree singularities associated with ill-posed problems. Most of the results presented here are based on [34].

The organization of the paper is as follows. In section 2 we describe our regularization approach and show its connection with Tikhonov regularization and with the trust-region subproblem. In section 3 we describe the trust-region subproblem and show its special properties in the discrete ill-posed case. In section 4 we describe the method LSTRS from [35] and discuss the issues related to ill-posed problems. In section 5 we present numerical results of LSTRS on regularization problems, including test problems from the Regularization Tools package [19], an inverse interpolation problem with field data, and a model seismic inversion problem with field data. We present some conclusions in section 6.

2. Regularization through trust regions.

As we mentioned before, regularization involves the formulation of a problem related to both the original problem with exact data and the problem with noisy data, where we incorporate a priori information such as the size or smoothness of the desired solution, the noise level in the data, or the statistical properties of the noise process.

One of the most popular regularization approaches is the classical Tikhonov regularization approach [40]

$$\min_{x\in\mathbb{R}^n} \; \|Ax-\bar{b}\|^2+\varepsilon^2\|x\|^2, \tag{3}$$

where $\varepsilon^2 > 0$ is the regularization parameter, and where the term $\|x\|^2$ could also be of the form $\|Lx\|^2$ for a general square or rectangular matrix L. This matrix could be, for example, the identity matrix as in (3), or a discrete form of first derivative. In the first case, the regularization parameter ε acts as a penalty parameter on the size of the solution, while in the latter case ε acts as a penalty parameter on the smoothness of the solution. Notice that if L is any square and nonsingular matrix, then a change of variable will reduce the problem to the form in (3). We can also accomplish such a transformation when L is a full-rank rectangular matrix by means of the methods in [10], [15]. Throughout the paper we assume that this transformation is possible, and therefore we consider only the case in which L is the identity matrix.

Observe that given ε, problem (3) becomes a damped least squares problem that we can solve with standard numerical linear algebra techniques for medium- and large-scale problems (cf. [3], [13]). However, determining an optimal value for the Tikhonov regularization parameter ε^2 can be as difficult as the original problem, and most of the methods currently available require the solution of several problems of type (3) for different values of ε. This approach might be very expensive in the large-scale setting. Recent and promising methods for computing the Tikhonov regularization parameter for large-scale problems have been proposed in [7], [8], and [25]. Calvetti, Reichel, and Zhang [8] propose a very elegant way of computing the parameter from the noise level in the data; Calvetti, Golub, and Reichel [7] propose a strategy based on the L-curve (see [17], [18]); while Kilmer and O'Leary [25] propose several strategies for computing the parameter from a problem in an appropriate subspace of smaller dimension.

In this work we will not assume a priori knowledge of the noise level or noise properties. Instead, we will assume that some information about the size or smoothness of the desired solution is available, and we formulate the regularization problem as

$$\min_{s.t. \|x\| \leq \Delta} \|Ax - \bar{b}\| \tag{4}$$

with $\Delta > 0$. As we show next, this formulation is equivalent to Tikhonov regularization.

Observe that if $\bar{b} \notin \mathcal{R}(A)$, where $\mathcal{R}(A)$ is the range of A, any solution of problem (4) is a regular point, and therefore the Karush–Kuhn–Tucker conditions for a feasible point x_* to be a solution of problem (4) with corresponding Lagrange multiplier λ_* are $(A^TA - \lambda_* I)x_* = -A^T\bar{b}$, $\lambda_* \leq 0$, and $\lambda_*(\|x_*\| - \Delta) = 0$. Further, since (4) is a convex quadratic problem, these conditions are both necessary and sufficient. Equivalence with problem (3) follows directly, since a solution x_* to problem (4) is also a solution to problem (3) corresponding to $\varepsilon^2 = -\lambda_*$. Conversely, if x_ε is a solution of (3) for a given ε, then x_ε solves problem (4) for $\Delta = \|x_\varepsilon\|$.

While Tikhonov regularization involves the computation of a parameter that does not necessarily have a physical meaning in most problems, the quadratically constrained least squares formulation has the advantage that, in some applications, the physical properties of the problem either determine or make it easy to estimate an optimal value for the norm constraint Δ. This is the case, for example, in image restoration where Δ represents the energy of the target image (cf. [2]).

Another example is the following problem closely related to (4):

$$\min_{s.t. \|Ax - \bar{b}\| \leq \rho} \|x\|,$$

where ρ is an estimate of the noise level in the data. For A nonsingular, the problem can be transformed into the form (4) by means of a change of variable. It is possible to do this in some special applications where an effective approximation to the inverse of A is available. This is the case in the example presented in section 5.3.

An additional advantage of the quadratically constrained least squares formulation is that it is a special case of a well-known problem in optimization, namely, that of minimizing a quadratic on a sphere or the trust-region subproblem

$$\min_{s.t. \|x\| \le \Delta} \frac{1}{2} x^T H x + g^T x, \tag{5}$$

where $H \in \mathbb{R}^{n \times n}$, $H = H^T$, $g \in \mathbb{R}^n$, and $\Delta > 0$. Problem (4) is a special case of (5) when $H = A^T A$ and $g = -A^T \bar{b}$.

The high degree of structure of the trust-region subproblem leads to strong theoretical properties and makes it possible to design efficient solution methods. For this reason we shall formulate the regularization problem as a trust-region subproblem.

3. The trust-region subproblem.

In this section we present the properties of the trust-region subproblem. In section 3.1, we consider the problem when H is any symmetric matrix in $\mathbb{R}^{n \times n}$, and g is any vector in $\mathbb{R}^n$. In section 3.2, we focus on the special case when $H = A^T A$, $g = -A^T \bar{b}$, and in addition A is a discretized version of a continuous operator in an ill-posed problem and $\bar{b}$ contains noise.

3.1. Structure of the problem.

A first observation about the trust-region subproblem is that it always has a solution. A not-so-obvious and quite remarkable fact about the problem is the existence of a characterization of its solutions, discovered independently by Gay [11] and Sorensen [36]. The result is contained in the following lemma.

LEMMA 3.1 (see [36]). *A feasible vector $x_* \in \mathbb{R}^n$ is a solution to* (5) *with corresponding Lagrange multiplier λ_* if and only if x_*, λ_* satisfy $(H - \lambda_* I)x_* = -g$ with $H - \lambda_* I$ positive semidefinite, $\lambda_* \le 0$, and $\lambda_*(\Delta - \|x_*\|) = 0$.*

Proof. See [36] for the proof. □

The optimality conditions imply that all the solutions of the trust-region subproblem are of the form $x = -(H - \lambda I)^\dagger g + z$ for $z \in \mathcal{N}(H - \lambda I)$, where $\mathcal{N}(\cdot)$ denotes the null space of a matrix and † denotes pseudoinverse. These solutions may lie in the interior or on the boundary of the set $\{x \in \mathbb{R}^n \mid \|x\| \le \Delta\}$. There are no solutions on the boundary if and only if H is positive definite and $\|H^{-1}g\| < \Delta$ (see [28]). In this case, the unique interior solution is $x = -H^{-1}g$ with Lagrange multiplier $\lambda = 0$. Boundary solutions satisfy $\|x\| = \Delta$ with $\lambda \le \delta_1$, where δ_1 is the smallest eigenvalue of H. The case $\lambda = \delta_1$ can only occur if $\delta_1 \le 0$, $g \perp \mathcal{S}_1$, where $\mathcal{S}_1 \equiv \mathcal{N}(H - \delta_1 I)$, and $\|(H - \delta_1 I)^\dagger g\| \le \Delta$. This corresponds to the so-called *hard case*, which poses great difficulties for the numerical solution of the trust-region subproblem since in this case it is necessary to compute an approximate eigenvector associated with the smallest eigenvalue of H. Moreover, in practice g will be nearly orthogonal to $\mathcal{S}_1$, and we can expect greater numerical problems in this case. We call this situation a *near hard case.* Note that whenever g is nearly orthogonal to $\mathcal{S}_1$ there is the possibility for the hard case or near hard case to occur. Therefore we call this a *potential hard case.* We show in section 3.2 that the potential hard case is precisely the common case for discrete ill-posed problems.

The conditions in Lemma 3.1 are computationally attractive since they provide a means for reducing the problem of computing boundary solutions for the trust-region subproblem from an n-dimensional problem to a zero-finding problem in one variable. We can accomplish this, for example, by solving the following equation in λ, known as the *secular equation*:

$$\Delta - \|x_\lambda\| = 0, \tag{6}$$

where $x_\lambda = -(H - \lambda I)^{-1}g$, and λ is monitored to ensure that $H - \lambda I$ is positive definite. Newton's method is particularly efficient for solving an equation equivalent to (6), and this approach, due to Moré and Sorensen [28], is the method of choice whenever it is affordable to compute the Cholesky factorization of matrices of the form $H - \lambda I$. However, in some applications this computation may be prohibitive either because of storage considerations or because the matrix H is not explicitly available. Therefore, we need other strategies to treat the problem in those cases. We describe one such strategy in section 4.

3.2. Discrete ill-posed trust-region subproblem.

We now study the trust-region subproblem in the special case when $H = A^TA$ and $g = -A^T\bar{b}$, where A comes from the discretization of a continuous operator in an ill-posed problem, and $\bar{b}$ contains noise. We will show that the potential hard case is the common case for these problems and also that it will occur in a *multiple* instance, where g is orthogonal to the eigenpaces associated with several of the smallest eigenvalues of H. This was first shown in [34] and is a consequence of the following result.

LEMMA 3.2. *Let $H = A^TA$ and $g = -A^T\bar{b}$, with $\bar{b} = b + s$. Suppose σ_k is the kth largest singular value of A with multiplicity m_k. Suppose u_j, v_j, $1 \le j \le m_k$, are left and right singular vectors associated with σ_k. Then*

$$g^Tv_j = -\sigma_k(u_j^Tb + u_j^Ts)\ ,\quad 1 \le j \le m_k.$$

Proof. The result follows directly assuming $A = U\Sigma V^T$ is a singular value decomposition of A, since this yields $g = -V\Sigma U^T\bar{b}$ with V orthogonal. □

Since $H = A^TA = V\Sigma^2V^T$, we see that for discrete ill-posed problems, the hard case is always present in an extreme form. Lemma 3.2 implies that whenever σ_k is very small then, for any reasonable noise level in $\bar{b}$, g will be nearly orthogonal to the subspace spanned by the right singular vectors associated with σ_k. This is precisely the case in discrete ill-posed problems, where the matrix A has a large cluster of very small singular values, and therefore we can expect g to be nearly orthogonal to the right singular vectors associated with such singular values. Since these vectors are eigenvectors corresponding to the smallest eigenvalues of A^TA, then g will be orthogonal to the eigenspaces corresponding to several of the smallest eigenvalues of A^TA and the potential hard case will occur in a *multiple* instance. Figure 1 illustrates this situation for problem **foxgood** from the Regularization Tools package by Hansen [19]. The problem is of dimension 300, and in the logarithmic plot we observe that g^Tv_k is of order 10^{-15} for approximately 292 of the right singular vectors of A.

Observe that for large noise level and σ_k not so small, g will not be nearly orthogonal to the eigenspace corresponding to the smallest eigenvalue of A^TA and the hard case will not occur. Therefore, in this case a high noise level implies a less difficult trust-region subproblem. However, we do not expect to compute a good approximation in the presence of large noise.

4. LSTRS for discrete ill-posed problems.

In this section we give a brief description of the method LSTRS from [35]. We present the method for a general symmetric matrix H and nonzero vector g and discuss the advantages of using this method for the special case of large-scale discrete ill-posed trust-region subproblems of type (4). LSTRS is based on formulating the trust-region subproblem as a parameterized eigenvalue problem. Such formulation comes from the observation that

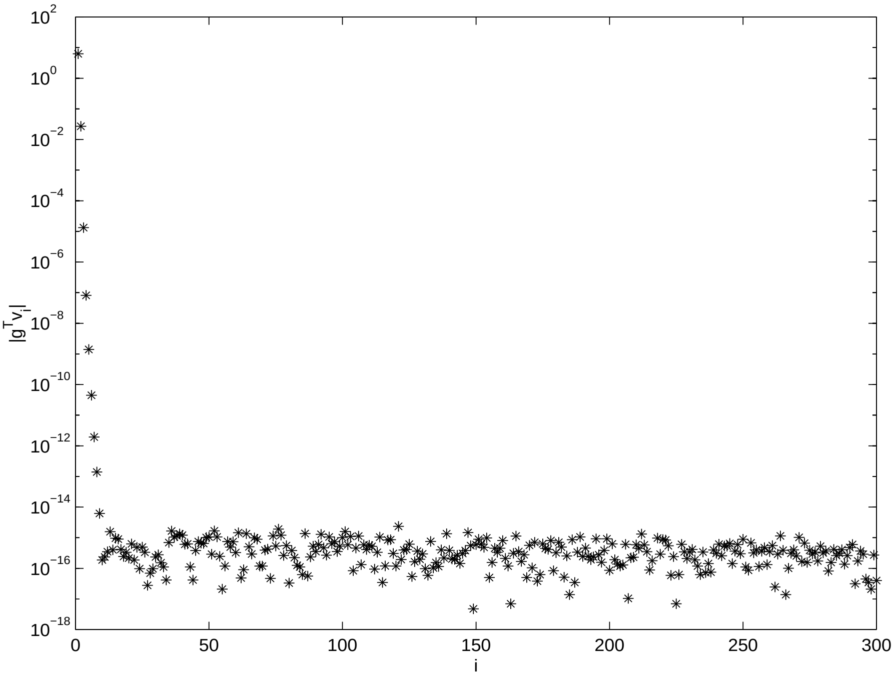


Fig. 1. *Orthogonality of g with respect to right singular vectors of a discretized operator in an ill-posed problem.*

if x_*, λ_* solve problem (5), then for $\alpha = \lambda_* - g^T x_*$, problem (5) is equivalent to

$$\begin{array}{rl} \min & \frac{1}{2} y^T B_\alpha y \\ \text{s.t.} & y^T y \le 1 + \Delta^2, \ e_1^T y = 1, \end{array} \tag{7}$$

where e_1 is the first canonical unit vector in $\mathbb{R}^{n+1}$ and $B_\alpha = \left(\begin{smallmatrix} \alpha & g^T \\ g & H \end{smallmatrix}\right)$. The solution of the trust-region subproblem consists of the last n components of the solution of problem (7).

Problem (7) suggests that if we know the optimal value for α, we can solve the trust-region subproblem by solving an eigenvalue problem for the smallest eigenvalue of B_α and an eigenvector with special structure. To see this, observe that if $\{\lambda, (1, x^T)^T\}$ is an eigenpair of B_α, then

$$\begin{pmatrix} \alpha & g^T \\ g & H \end{pmatrix} \begin{pmatrix} 1 \\ x \end{pmatrix} = \begin{pmatrix} 1 \\ x \end{pmatrix} \lambda,$$

which is equivalent to

$$\alpha - \lambda = -g^T x \quad \text{and} \quad (H - \lambda I)x = -g \ . \tag{8}$$

If λ is the smallest eigenvalue of B_α and since the eigenvalues of H interlace the eigenvalues of B_α by the Cauchy interlace theorem (cf. [33]), then $H - \lambda I$ is positive semidefinite. Therefore, two of the optimality conditions in Lemma 3.1 are automatically satisfied in this case. If, in addition, $\lambda \le 0$ and $\|x\| = \Delta$, we will have a solution for the trust-region subproblem.

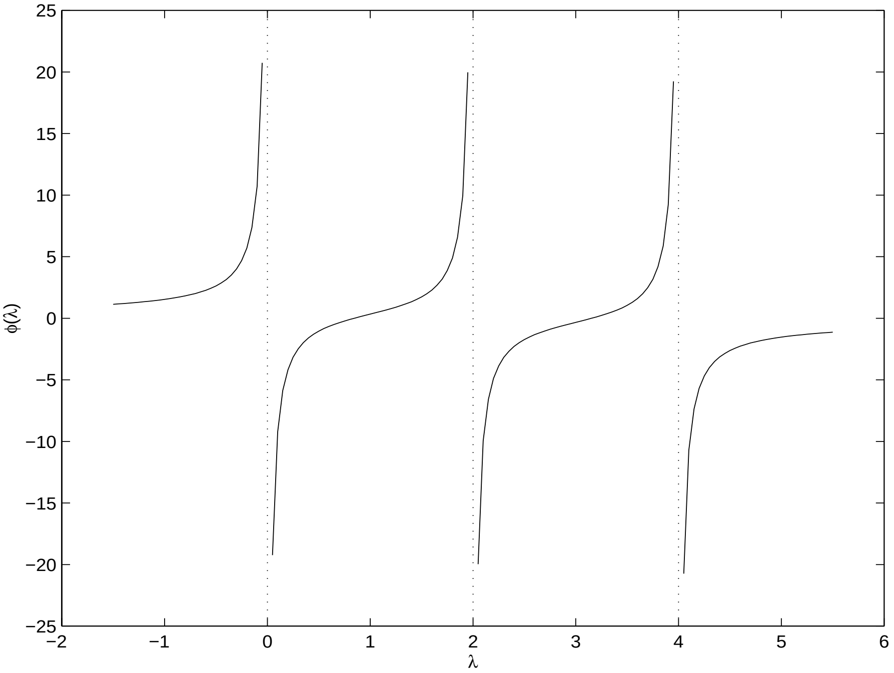


FIG. 2. *Secular function* $\phi(\lambda)$.

LSTRS consists of iteratively adjusting the parameter α to drive it towards the optimal value $\alpha_* = \lambda_* - g^T x_*$. This is accomplished in the following way. Let $\phi(\lambda) = -g^T x$ for x satisfying $(H - \lambda I)x = -g$, and note that $\phi'(\lambda) = x^T x$. Both ϕ and ϕ' are rational functions with poles at a subset of the eigenvalues of H. Figure 2 illustrates the typical behavior of $\phi(\lambda)$ when H is a 3×3 matrix with eigenvalues $0, 2, 4$. The LSTRS iteration is based on approximately solving the secular equation (6), using rational interpolation on ϕ and ϕ'. Observe that, in view of (8), we can obtain convenient interpolation points by solving eigenvalue problems for the smallest eigenvalue of B_α for different values of the parameter α. LSTRS computes the interpolation points in this way, using the implicitly restarted Lanczos method (IRLM) [37] as implemented in ARPACK [27] to solve the eigenvalue problems. The IRLM has fixed storage requirements and relies upon matrix-vector products only, features that make it suitable for large-scale problems.

The strategy described above works as long as the smallest eigenvalue of B_α has a corresponding eigenvector that can be safely normalized to have first component one. The adjustment of the parameter becomes very difficult in the hard case and near hard case since in these situations the smallest eigenvalue of B_α might not have a corresponding eigenvector with the desired structure (see [34], [35], [38]). Moreover, in the near hard case δ_1 is a weak pole of $\phi(\lambda)$ and the function becomes very steep around this value, as Figure 3 illustrates. This makes the interpolation problem very ill-conditioned. The situation is considerably more difficult for ill-posed problems where *several* of the smallest eigenvalues of H are weak poles of $\phi(\lambda)$. We illustrate this case in Figure 4 which shows $\phi(\lambda)$ for the test problem to be discussed in section 5.3. LSTRS relies on the complete characterization of the hard case given in [34] to proceed with the iteration even when the desired eigenvector cannot be normalized

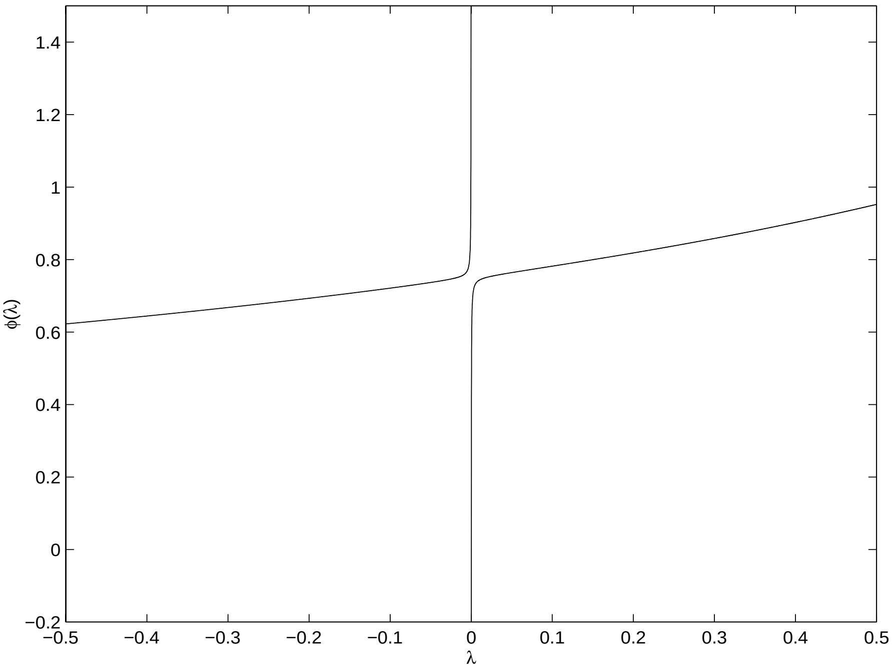


FIG. 3. $\phi(\lambda)$ *in the near hard case.*

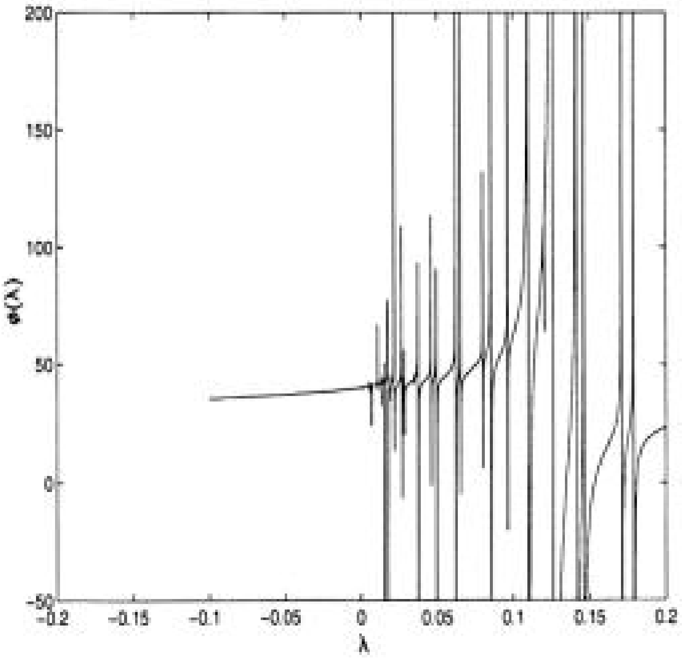


FIG. 4. $\phi(\lambda)$ *for the viscoacoustic wave equation,* $n = 500$.

to have first component one, and to compute nearly optimal solutions in any instance of the hard case including *multiple* occurrences as in ill-posed problems.

A detailed description of the results concerning the hard case and the elaborate algorithmic techniques derived from those results are beyond the scope of this paper. We refer the reader to [34] and [35] for more details.

From the above presentation we see that LSTRS has desirable features for solving large-scale trust-region subproblems in general, and for handling discrete ill-posed problems in particular. This is not surprising since the regularization of discrete forms of ill-posed problems was part of the motivation for developing the method. There are, however, some issues that must be taken into account when implementing LSTRS to treat ill-posed problems. As we saw in section 2, for these problems the smallest eigenvalues of H are clustered and close to zero, and because of the interlacing property the smallest eigenvalues of B_α will also be clustered and small for certain values of Δ. Computing a clustered set of small eigenvalues with a method that relies only on matrix-vector products with the original matrix is likely to fail since the multiplication will annihilate components precisely in the direction of the eigenvectors of interest. This difficulty may be overcome through the use of a spectral transformation. Instead of trying to find the smallest eigenvalue of B_α directly, we work with a matrix function $T(B_\alpha)$ and use the fact that $B_\alpha q = q\lambda \iff T(B_\alpha)q = qT(\lambda)$. If we are able to construct T so that $|T(\lambda_1)| \gg |T(\lambda_j)|,\ j > 1$, then a Lanczos-type method such as the IRLM will converge much faster towards the eigenvector q_1 corresponding to λ_1. We use a Chebyshev polynomial T_ℓ of degree ℓ constructed to be as large as possible on λ_1 and as small as possible on an interval containing the remaining eigenvalues of B_α. Convergence of IRLM is often greatly enhanced through this spectral transformation strategy. After convergence, the eigenvalues of B_α are recovered via the Rayleigh quotients with the converged eigenvectors.

Finally, the occurrence of an interior solution when $H = A^T A$ is positive definite in regularization problems deserves a special comment. In this case the solution of the trust-region subproblem corresponds to the least squares solution of the original problem. This solution is contaminated by noise and is of no interest. When we detect an interior solution we have taken the simple approach of reducing the trust-region radius and restarting the method. It is worth noting that if we knew that the noise level in the data was low, then if λ is close to zero when we detect an interior solution, we could approximate the least squares solution by x satisfying (8) since this would be a reasonable approximation to $x = -H^{-1}g$. Note that in this case it would not be necessary to solve a linear system to obtain the solution.

5. Numerical results.

In this section we present numerical experiments to illustrate the performance of LSTRS on regularization problems from different sources, including both test problems and real applications. We used a MATLAB version of LSTRS running under MATLAB 5.3 using Mexfile interfaces to access ARPACK [27] and also the routines to compute matrix-vector products in some of the examples. Notice that the capabilities of ARPACK have been incorporated into MATLAB 6 and are now available through the routine `eigs`. We ran our experiments on a SUN Ultrasparc 2 with a 200 MHz processor and 256 Megabytes of RAM running Solaris 5.6. The floating point arithmetic was IEEE standard double precision with machine precision $2^{-52} \approx 2.2204 \cdot 10^{-16}$.

We present three sets of experiments. In section 5.1 we describe the results obtained on test problems from the Regularization Tools package [19]. In section 5.2 we present an inverse interpolation problem with field data. In section 5.3 we present

Table 1
Results of LSTRS on test problems from the Regularization Tools package.

Problem	Dim.	Δ	$\|x\|$	$\frac{\|x-x_{IP}\|}{\|x_{IP}\|}$	MV Prods.	Iter.
Ill **heat**	300	4.2631	4.2527	3.5684e-01	1721	8
Ill **heat**	1000	7.7829	7.7497	2.6900e-01	967	8
Well **heat**	300	4.2631	4.2958	9.1853e-02	1049	4
ilaplace	195	2.7629	2.7362	1.8537e-01	349	4
parallax	300	5.0000	5.0421	–	869	10
phillips	300	2.9999	2.9869	2.6883e-02	521	6
phillips	1000	3.0000	2.9839	3.3607e-02	575	6
shaw	300	17.2893	17.2467	6.0625e-02	510	6
shaw	1000	31.5659	31.6002	5.2847e-02	423	5

a model seismic inversion problem using a standard data set. The various stopping tolerances on $\frac{|\|x\|-\Delta|}{\Delta}$ were chosen (as they often are in practice) in an ad hoc fashion after some trial runs.

5.1. Problems from the Regularization Tools package. In this section we will present the results of LSTRS on problems from the Regularization Tools package [19]. This package consists of a set of MATLAB routines for the analysis of discrete ill-posed problems along with test problems that are easy to generate. All the test problems come from the discretization of a Fredholm integral equation of the first kind

$$\int_a^b K(s,t)f(t)dt = g(s),$$

and the problem is to compute the unknown function $f(t)$ given $g(s)$ and $K(s,t)$.

In all cases we solved a quadratically constrained least squares problem (4) where A came from the discretization of the kernel $K(s,t)$ and $b = g(s_i)$ at discrete points $s_i \in [a,b]$, where $i = 1,\ldots,n$ and n is the dimension of the problem. For some of the problems the exact solution $f(t)$ was available and in those cases we used $x_{IP} = f(t_i)$ for comparison purposes, where $t_i \in [a,b]$, $i = 1,\ldots,n$. Note that in general Ax_{IP} is different from b. Unless otherwise specified, we used $\Delta = \|x_{IP}\|$ as trust-region radius. In ARPACK, we used nine Lanczos basis vectors with seven shifts on each implicit restart. The required accuracy for the eigenpairs was 10^{-2}. The initial vector for the Lanczos factorization was a randomly generated vector that remained fixed in all the experiments. We solved the trust-region subproblems to a relative accuracy of $|\frac{\|x\|-\Delta}{\Delta}| < 10^{-2}$. We also solved the problems to a higher accuracy but this was computationally more expensive and did not seem to improve the accuracy of the regularized solution x with respect to the exact solution x_{IP} for this particular set of problems. In Table 1, we present the results for a subset of problems from [19].

Several observations are in order concerning Table 1. The third and fourth columns indicate that in all cases LSTRS solved the trust-region subproblem to the prescribed accuracy. The quality of the regularized solution or a measure of how well this solution approximates the exact solution x_{IP} is reported in the fifth column, where a dash indicates that x_{IP} was not available. We see that, generally, there is a reasonable agreement between computed and exact solutions, with relative errors of order 10^{-2}. The number of matrix-vector products is reported in column six, and the last column shows the number of LSTRS iterations.

The primary purpose of these tests was simply to verify that our approach would compute reasonably accurate regularized solutions to well-known examples of discrete forms of ill-posed problems. We cannot draw any conclusions about computational cost or expected number of matrix-vector products from these small examples. However, our limited experience would indicate that the number of matrix-vector products does not increase significantly with the dimension of the problem, and we give examples of this in sections 5.2 and 5.3.

Finally, we remark that some modifications probably would have been possible to improve the accuracy of the computed solutions to this set of examples. For problems ill-conditioned **heat** (inverse heat equation) and **ilaplace** (inverse Laplace transformation), the relative error is probably higher than one would like. It turns out that in these cases the solutions are highly oscillatory. This suggests that we should have solved the trust-region subproblem with a constraint of the form $\|Lx\|$, where L is a discrete form of first derivative. Since our goal here was a basic verification of LSTRS on such problems, we did not analyze each case separately, nor did we pursue more elaborate formulations.

5.2. An inverse interpolation problem. The two-dimensional (2-D) linear interpolation problem consists of using a linear interpolant to find the values of a function at arbitrary points given the values of the function at equally spaced points. A more interesting problem is the inverse interpolation problem: finding the values of the function on a regular grid of points from which we can extract given values of the function at irregularly spaced points by linear interpolation. We can pose the 2-D inverse interpolation problem as a least squares problem,

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|,$$

where $A \in \mathbb{R}^{m \times n}$ is the 2-D linear interpolant and $b \in \mathbb{R}^m$ contains the function values at irregularly spaced points.

To illustrate the performance of LSTRS on this kind of problem we will use the example of constructing a depth map of the Sea of Galilee on a regular grid of points, given depth measurements at irregularly spaced points. The data consists of triplets $v_i,\ w_i,\ b_i, i = 1, \dots, 132044$, representing coordinates on the plane and depth, respectively. The data was collected from a ship using an echo sounder. The data contains noise coming from different sources, including malfunctioning equipment that reported zero depths at points in the middle of the lake and the fact that the measurements were taken at different times of the year and therefore varied greatly from rainy season to dry season. See [1] for a complete description of the data acquisition process. In Figure 5 we show a view from above of a three-dimensional (3-D) plot of the original data. The straight lines we observe in the figure are the tracks of the ship. Therefore, the data acquisition process was an additional source of noise. As Clærbout points out [9], an image of the sea should not include those lines.

In our experiments the size of the grid was $n = 201 \times 201 = 40401$, which is also the number of unknowns when the 2-D grid is represented as a one-dimensional vector. The number of rows in A was $m = 132044$. This matrix was ill-conditioned and was not available explicitly, but we could compute the action of A and A^T on vectors by means of FORTRAN routines. In all the experiments, we solved the trust-region subproblems to a relative accuracy of $|\frac{\|x\|-\Delta}{\Delta}| \leq 10^{-3}$. The size of the Lanczos basis was five, and we applied three shifts on each implicit restart. Therefore, the storage requirement was essentially five vectors of length 40401.

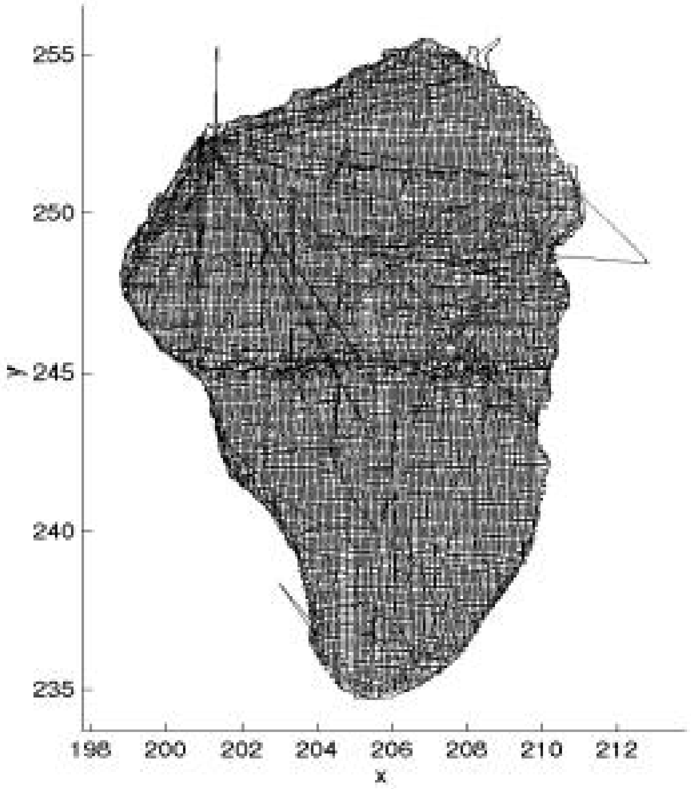


FIG. 5. *Sea of Galilee from original data.*

We posed the trust-region subproblem as

$$\min_{s.t. \|Lx\| \leq \Delta} \frac{1}{2} x^T A^T A x - (A^T b)^T x,$$

where L was either the $n \times n$ identity matrix I, or the matrix M_p, a discretization of the following pth power of the (scaled) 2-D Helmholtz operator

$$(\mathcal{I} - \nabla^T * \text{diag}(s) * \nabla)^p, \tag{9}$$

where $\mathcal{I}$, ∇, and ∇^T are the identity, gradient, and divergence operators, respectively; the vector $s > 0$ is a 2-D vector of scales; the expression diag(s) denotes a diagonal matrix with the components of the vector s on the diagonal; and p is a real scalar.

We ran several experiments for different trust-region radii since in this application we did not have a priori information about the size or smoothness of the desired solution. Figure 6 shows the result for $\Delta = 6000$ and $L = I$. This image still shows the tracks of the ship and does not reveal any known features of the depth distribution of the lake. The contour plot (not shown) is very rough with highly oscillatory contours. This suggested the need to introduce a constraint on the smoothness of the solution.

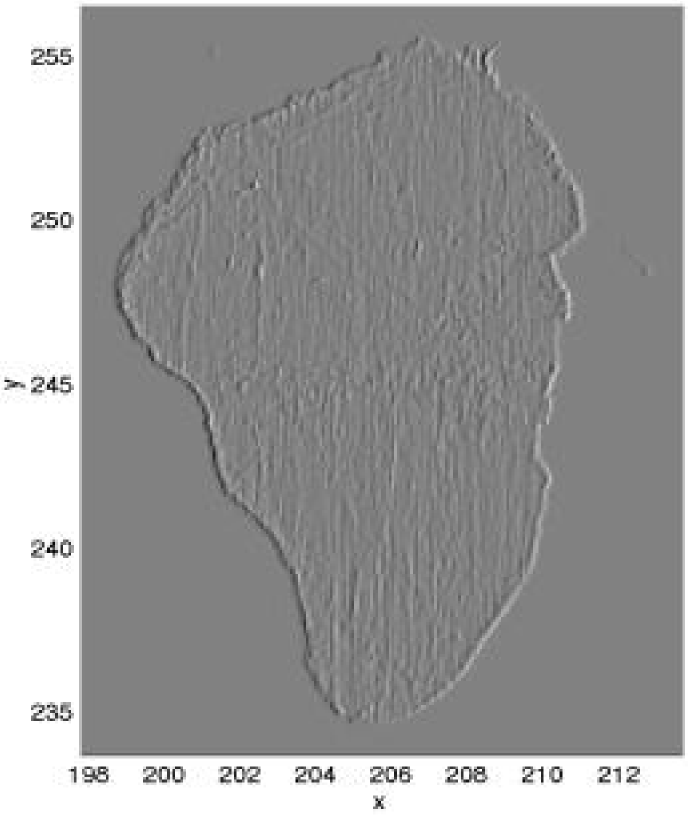


FIG. 6. *Sea of Galilee, regularizing with constraint on size of solution.*

We then solved the trust-region subproblem with $L = M_p$, a discretization of (9), and $\Delta = 26000$. Figure 7 shows the solution for $p = 0.3$. In this image we were able to identify some of the features reported in [1], such as the shallow areas in the northeast, a scarp in the southeast, and a more prominent scarp in the southwest.

We also tried the approach of solving the trust-region subproblem with $L = I$ and applying the Helmholtz operator a posteriori. We call this approach *postsmoothing.* We tried the postsmoothing approach with $p = -1$ for $\Delta = 23423$, which is the norm of x when $\|M_p x\| = 26000$, and we obtained an image very similar to the one in Figure 7. We also used this approach for $\Delta = 6000$, obtaining the image in Figure 8, which clearly shows the features mentioned before. Table 2 shows that the postsmoothing approach is less expensive than using a constraint on the smoothness. There are two reasons for this difference in efficiency. The first one is that the matrix-vector products when $L = M_p$ are more expensive than the matrix-vector products when $L = I$. The second reason is that the smallest eigenvalues of B_α are close to the smallest eigenvalues of $L^{-T}A^TAL^{-1}$, and these are more clustered for $L = M_p$ than for $L = I$. This causes slow convergence of the IRLM. We note, however, that the cost is not too high in either approach relative to the dimension of the problem.

The postsmoothing approach has the drawback that we do not know its physical meaning, and a closer look at the result shows that the postsmoothing is causing a high degree of perturbation on the depths since the lowest point is known to be around

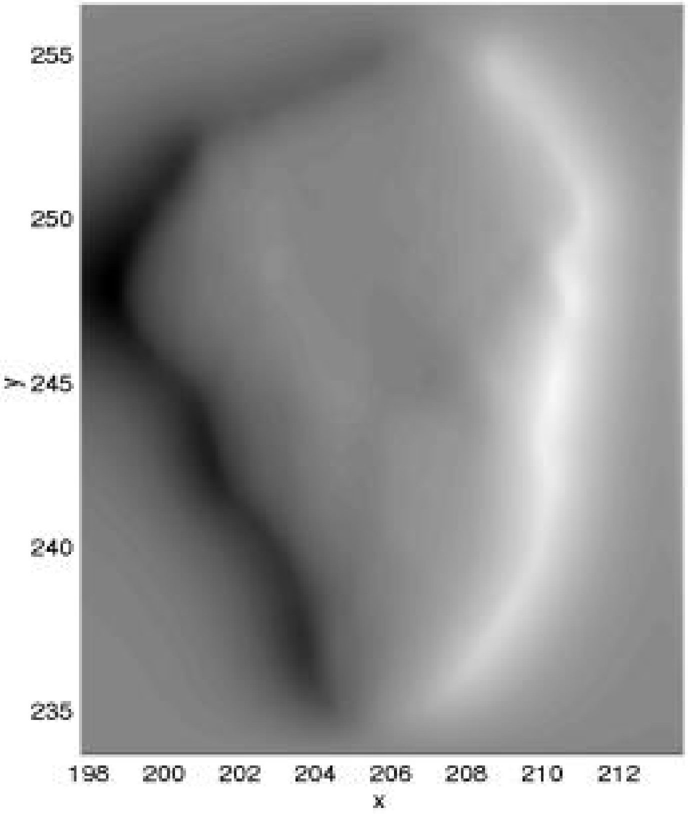


FIG. 7. *Sea of Galilee, regularizing with constraint on smoothness.*

256m below sea level, and the lowest point in Figure 8 is -45m. Another interesting aspect of this particular regularization approach is that it produces very smooth solutions, which was also noted in [21], and this causes, for example, the resulting regular grid to miss the true deepest point located approximately at coordinates $(207, 247)$ and yields a deepest point located at approximately $(205, 245)$ for Figure 8. It is quite remarkable, though, how the known features of the lake are clearly present in this image.

5.3. A model seismic inversion problem. We also solved the quadratically constrained least squares problem (4) where the problem comes from the discretization of the linear viscoacoustic model. As explained in [6], this model describes the behavior of an anelastic fluid, in which the strain response to a change of stress is linear but not completely instantaneous. A relaxation function $G(t, \vec{x})$ is used to express the stress-strain relation. The equations of motion relate $G(t, \vec{x})$, the material density $\rho(\vec{x})$, the pressure (stress) $p(t, \vec{x})$, the particle velocity $\vec{v}(t, \vec{x})$, and the body force source $f(t, \vec{x})$ in the following way:

$$\begin{aligned} p_{,t}(t, \vec{x}) &= -\dot{G}(t, \vec{x}) * \nabla \cdot \vec{v}(t, \vec{x}) + f(t, \vec{x}), \\ \vec{v}_{,t}(t, \vec{x}) &= -\frac{1}{\rho(\vec{x})} \nabla p(t, \vec{x}), \end{aligned} \tag{10}$$

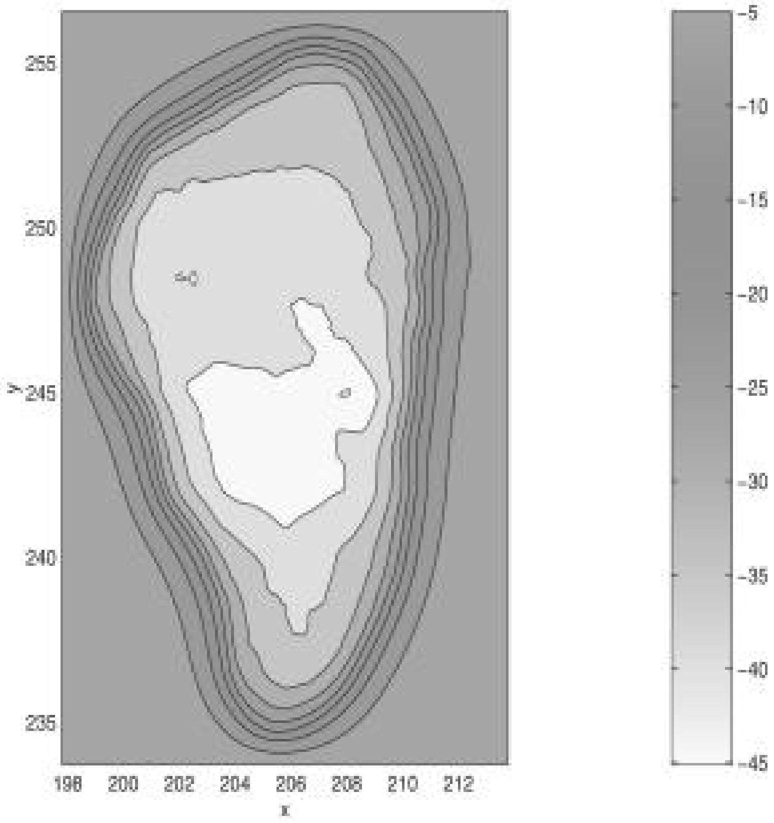


FIG. 8. *Sea of Galilee, regularizing with constraint on size of solution and postsmoothing.*

TABLE 2
Performance of LSTRS on an inverse interpolation problem.

Dimension: 40401 Storage: 5 vectors	Δ	$\|x_*\|$	LSTRS Iter.	MV Prods.	CPU time (min.)
TRS with postsmoothing	23423	23423	4	206	1.40
Constraint on smoothness	26000	25980.61	15	723	22.05

where $p = 0,\ \vec{x} = 0$ for $t \ll 0$. In (10), t denotes time and $\vec{x}$ denotes position.

These equations are used in [6] to model the propagation of waves in marine media using the relaxation function for a standard linear fluid. The matrix A in our quadratically constrained least squares problem corresponds to DF, a linearized version of the forward map or prediction operator. The matrix A^T corresponds to the adjoint of DF denoted by DF^*. The operators DF and DF^* were not explicitly available, but their action on vectors was obtained by solving a simplified (layered) and linearized version of (10). See [5, Chap. 5] and [6] for more details. The data vector $\bar{b}$ is a seismogram containing velocities of waves measured in oil wells in the North Sea. The data is part of the Mobil AVO data set [23], a standard data set for testing inversion methods. The parameters to be estimated in the experiment are the

Table 3
Performance of LSTRS on the viscoacoustic wave equation.

Dimension	Δ	$\|x\|$	LSTRS Iter.	MV Prods.	Storage
121121	0.5	0.5	2	15	4 vectors

stress-strain ratio under simple hydrostatic pressure and the material density. The quadratically constrained least squares problems arise in the context of sophisticated nonlinear inversion strategies [12], where the norm constraint was of the form $\|x\| \leq \Delta$. The quadratically constrained least squares problems were obtained from problems of type

$$\min_{s.t. \|Ax-\bar{b}\| \leq \rho} \|x\|$$

by means of a change of variable made possible by the availability of an effective approximation to the inverse of A. The parameter ρ is an estimate of the noise level in $\bar{b}$ and is known a priori in this case. The reformulation of the problem yields $\Delta = \rho$.

The dimensions of the problem are $m = n = 121121$. Table 3 shows the result obtained when we used LSTRS to solve the trust-region subproblem to an accuracy of $|\frac{\|x\|-\Delta}{\Delta}| < 10^{-5}$ using four Lanczos basis vectors. The method is very efficient for small Δ since in this case the smallest eigenvalue of B_α is well separated from the rest and the IRLM converges rapidly to such eigenvalue. For larger Δ, the eigenvalue of interest belongs to a cluster and the IRLM needs more iterations to compute it.

In Table 3 we can observe the low storage and low number of matrix-vectors products required to solve the problem to a very high accuracy.

6. Conclusions.

We considered the problem of regularizing large-scale discrete forms of ill-posed problems arising in several applications. We posed the regularization problem as a quadratically constrained least squares problem, showed the relationship of this approach to Tikhonov regularization and to the trust-region subproblem, and analyzed the latter in the ill-posed case.

We have presented numerical results obtained when we used the recently developed method LSTRS for the large-scale trust-region subproblem to solve regularization problems from a wide variety of applications including problems with field data. The method requires solving a sequence of large-scale eigenvalue problems, which is accomplished with a variant of the Lanczos method. An important feature of LSTRS is that it computes both the solution and the Tikhonov regularization parameter from the prescribed norm.

LSTRS is particularly suitable for large-scale discrete forms of ill-posed problems, for which it computed regularized solutions close to the desired exact solutions using limited storage and moderate computational effort in general. For real applications the method required a low number of matrix-vector products with respect to the dimension of the problem, and storage comparable to or less than the conjugate gradient method. Our approach also had the desirable feature of providing systematic stopping criteria.

We are currently investigating various approaches to preconditioning, aiming to generate eigenvalue problems that can be solved more efficiently. Although further improvement is needed, LSTRS proved to be a promising method for the numerical treatment of large-scale discrete forms of ill-posed problems in which the norm of the desired solution is prescribed.

Acknowledgments. We would like to thank Chao Yang for making his MATLAB code for the Chebyshev spectral transformation available to us. We thank Ricardo von Borries for his help in recovering what seemed to be a hopelessly lost illustration. We would also like to express our deep gratitude to Bill Symes who provided us with the data codes for the 2-D linear interpolant and interpretation of the results in section 5.2. Bill also provided the model seismic problem in section 5.3 and the AVO data set. We also thank Professor Zvi Ben-Avraham from Tel-Aviv University for making the data of the Sea of Galilee available and for providing important references. Finally, we would like to thank the two anonymous referees for their careful reading of the original manuscript, and for their valuable and relevant suggestions, which helped improve the original version of this paper.

REFERENCES

[1] Z. Ben-Avraham, G. Amit, A. Golan, and Z.B. Begin, *The bathymetry of Lake Kinneret and its structural significance*, Israel J. Earth Sci., 39 (1992), pp. 77–84.

[2] M. Bertero and P. Bocacci, *Introduction to Inverse Problems in Imaging*, Institute of Physics, Bristol, UK, 1998.

[3] Å. Björck, *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, 1996.

[4] Å. Björck, E. Grimme, and P. Van Dooren, *An implicit shift bidiagonalization algorithm for ill-posed systems*, BIT, 34 (1994), pp. 510–534.

[5] J.O. Blanch, *A Study of Viscous Effects in Seismic Modeling, Imaging, and Inversion: Methodology, Computational Aspects, and Sensitivity*, Ph.D. thesis, Department of Geology and Geophysics, Rice University, Houston, 1995.

[6] J.O. Blanch, W.W. Symes, and R.J. Versteeg, *A numerical study of linear viscoacoustic inversion*, in Comparison of Seismic Inversion Methods on a Single Real Data Set, R.G. Keys and D.J. Foster, eds., Society of Exploration Geophysicists, Tulsa, OK, 1998, pp. 13–44.

[7] D. Calvetti, G.H. Golub, and L. Reichel, *Estimation of the L-curve via Lanczos bidiagonalization*, BIT, 39 (1999), pp. 603–619.

[8] D. Calvetti, L. Reichel, and Q. Zhang, *Iterative exponential filtering for large discrete ill-posed problems*, Numer. Math., 83 (1999), pp. 535–556.

[9] J. Clærbout, *Geophysical Estimation by Example: Environmental Soundings Image Enhancement: Multidimensional Autoregression*, available online from http://sepwww.stanford.edu/sep/jon/index.html.

[10] L. Eldén, *Algorithms for the regularization of ill-conditioned least squares problems*, BIT, 17 (1977), pp. 134–145.

[11] D. M. Gay, *Computing optimal locally constrained steps*, SIAM J. Sci. Statist. Comput., 2 (1981), pp. 186–197.

[12] M.S. Gockenbach and W.W. Symes, *Duality for inverse problems in wave propagation*, in Large Scale Optimization, IMA Vol. Math. Appl. 92, L. Biegler, T. Coleman, F. Santosa, and A. Conn, eds., Springer-Verlag, New York, 1997, pp. 37–61.

[13] G.H. Golub and C.F. Van Loan, *Matrix Computations*, 3rd ed., Johns Hopkins University Press, Baltimore, 1996.

[14] G.H. Golub and U. von Matt, *Quadratically constrained least squares and quadratic problems*, Numer. Math., 59 (1991), pp. 561–580.

[15] M. Hanke and P.C. Hansen, *Regularization methods for large-scale problems*, Surveys Math. Indust., 3 (1993), pp. 253–315.

[16] P.C. Hansen, *The discrete Picard condition for discrete ill-posed problems*, BIT, 30 (1990), pp. 658–672.

[17] P.C. Hansen, *Analysis of discrete ill-posed problems by means of the L-curve*, SIAM Rev., 34 (1992), pp. 561–580.

[18] P.C. Hansen and D.P. O'Leary, *The use of the L-curve in the regularization of discrete ill-posed problems*, SIAM J. Sci. Comput., 14 (1993), pp. 1487–1503.

[19] P.C. Hansen, *Regularization Tools: A MATLAB package for analysis and solution of discrete ill-posed problems*, Numer. Algorithms, 6 (1994), pp. 1–35.

[20] P.C. Hansen, *Rank-Deficient and Discrete Ill-Posed Problems*, Doctoral Dissertation, UNI•C, Technical University of Denmark, Lyngby, Denmark, 1996.

[21] P.C. Hansen, M. Jacobsen, J.M. Rasmussen, and H. Sørensen, *The PP-TSVD algorithm*

for image restoration problems, in Methods and Applications of Inversion, Lecture Notes in Earth Sci. 92, P.C. Hansen, B.H. Jacobsen, and K. Mosegaard, eds., Springer-Verlag, Berlin, 2000, pp. 171–186.

[22] L. Kaufman and A. Neumaier, *Image Reconstruction through Regularization by Envelope Guided Conjugate Gradients*, AT&T Bell Laboratories, 4-14, 1994; available online from http:cm.bells-labs.com/cm/cs/doc/94/4-14.ps.gz.

[23] R.G. Keys and D.J. Foster, *A data set for evaluating and comparing seismic inversion methods*, in Comparison of Seismic Inversion Methods on a Single Real Data Set, R.G. Keys and D.J. Foster, eds., Society of Exploration Geophysicists, Tulsa, OK, 1998, pp. 1–12.

[24] M.E. Kilmer and D.P. O'Leary, *Pivoted Cauchy-like preconditioners for regularized solution of ill-posed problems*, SIAM J. Sci. Comput., 21 (1999), pp. 88–110.

[25] M.E. Kilmer and D.P. O'Leary, *Choosing regularization parameters in iterative methods for ill-posed problems*, SIAM J. Matrix Anal. Appl., 22 (2001), pp. 1204–1221 .

[26] C.L. Lawson and R.J. Hanson, *Solving Least Squares Problems*, Classics Appl. Math. 15, SIAM, Philadelphia, 1995.

[27] R.B. Lehoucq, D.C. Sorensen, and C. Yang, *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems by Implicitly Restarted Arnoldi Methods*, SIAM, Philadelphia, 1998.

[28] J.J. Moré and D.C. Sorensen, *Computing a trust region step*, SIAM J. Sci. Statist. Comput., 4 (1983), pp. 553–572.

[29] J. Nagy, V. Pauca, R. Plemmons, and T. Torgersen, *Space-varying restoration of optical images*, Opt. Soc. Amer. A, 14 (1997), pp. 3162–3174.

[30] F. Natterer, *The Mathematics of Computerized Tomography*, John Wiley, New York, 1986.

[31] A. Neumaier, *Solving ill-conditioned and singular linear systems: A tutorial on regularization*, SIAM Rev., 40 (1998), pp. 636–666.

[32] G. Nolet, ed., *Seismic Tomography, with Applications in Global Seismology and Exploration Geophysics*, D. Reidel, Dordrecht, 1987.

[33] B.N. Parlett, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, NJ, 1980.

[34] M. Rojas, *A Large-Scale Trust-Region Approach to the Regularization of Discrete Ill-Posed Problems*, Ph.D. thesis, Technical Report TR98-19, Department of Computational and Applied Mathematics, Rice University, Houston, TX, 1998.

[35] M. Rojas, S.A. Santos, and D.C. Sorensen, *A new matrix-free algorithm for the large-scale trust-region subproblem*, SIAM J. Optim., 11 (2000), pp. 611–646.

[36] D.C. Sorensen, *Newton's method with a model trust region modification*, SIAM J. Numer. Anal., 19 (1982), pp. 409–426.

[37] D.C. Sorensen, *Implicit application of polynomial filters in a k-step Arnoldi method*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 357–385.

[38] D.C. Sorensen, *Minimization of a large-scale quadratic function subject to a spherical constraint*, SIAM J. Optim., 7 (1997), pp. 141–161.

[39] W.W. Symes, *A differential semblance criterion for inversion of multioffset seismic reflection data*, J. Geophysical Research, 98 (1993), pp. 2061–2073.

[40] A.N. Tikhonov, *Regularization of incorrectly posed problems*, Soviet Math., 4 (1963), pp. 1624–1627.

SIAM J. SCI. COMPUT.
Vol. 23, No. 6, pp. 1861–1874

HIDDEN DISCONTINUITIES AND PARAMETRIC SENSITIVITY CALCULATIONS*

JOHN E. TOLSMA[†] AND PAUL I. BARTON[†]

Abstract. Parametric sensitivity analysis is becoming a very important tool when studying differential equations. However, potential pitfalls exist if this analysis is applied naively. In particular, if the model contains discontinuities, then the application of standard numerical codes will typically result in incorrect sensitivity trajectories. The purpose of this paper is to demonstrate the problems associated with "hidden discontinuities" during sensitivity analysis (i.e., discontinuities not handled explicitly) and to present a code analysis and transformation approach through which differential equations containing discontinuities can be solved efficiently and correctly with minimal effort required by the modeler.

Key words. hybrid discrete/continuous simulation, parametric sensitivity analysis, differential-algebraic equations, automatic differentiation, state events, DAEPACK

AMS subject classifications. 65P99, 37N30, 65L05

PII. S106482750037281X

1. Introduction and motivation.

Parametric sensitivity analysis is used to obtain information concerning state variable variations with respect to infinitesimal perturbations in the values of model parameters. Some common applications for parametric sensitivity analysis are experimental design, model reduction, and obtaining gradients and Jacobians in control parameterization approaches for dynamic optimization and parameter estimation. The advent of better algorithms for computing forward sensitivities allows this calculation to be performed on much larger and more complex systems [14, 7]. However, potentially serious problems exist if parametric sensitivity analysis is applied to dynamic models containing discontinuities. These problems are addressed in this paper by considering models containing discontinuities within a hybrid systems framework.

Hybrid discrete/continuous systems (or simply hybrid systems) are dynamical systems that exhibit both discrete and continuous behavior. The continuous behavior of the model is usually described by one or more ordinary differential equation (ODE) or differential-algebraic equation (DAE) systems. The discrete behavior, which occurs at particular points in time known as events, includes phenomena such as nonsmooth forcing, switching of the vector field, and jumps in the state. We can broadly distinguish between two types of events: time events and state events. A time event is an event where the time of occurrence is known a priori. In contrast, a state event occurs when some condition involving the state variables is satisfied (e.g., the level of liquid in a tank reaches a certain height), and the time of occurrence is in general not known a priori.

Hybrid system models appear in a wide variety of disciplines. They appear directly when modeling discrete control actions imposed on a system (e.g., a safety interlock system), disturbances introduced into a continuous system, cell signaling

*Received by the editors May 17, 2000; accepted for publication (in revised form) July 17, 2001; published electronically February 20, 2002. This work was supported by the EPA Center for Airborne Organics at MIT and Alstom Power.

http://www.siam.org/journals/sisc/23-6/37281.html

[†]Department of Chemical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139-4307 (jtolsma@mit.edu, pib@mit.edu).

models, and certain electrical circuit models. In the examples above, the state events and the discontinuities they introduce are an important feature of the phenomena of interest, and it is well known that special action must be taken to simulate such systems [12, 2, 15]. In other, seemingly continuous examples, *hidden discontinuities* can have adverse effects because the special measures mentioned above are not taken. These discontinuities are typically introduced by modeling abstractions; e.g., physical properties of fluids are often represented by piecewise smooth semiempirical relationships. Within a code these may appear as nonsmooth intrinsic functions, such as MIN and MAX, as well as the more obvious IF statements. The problem with models where the discrete behavior is not of primary interest is that often these discontinuities are not handled explicitly (i.e., they remain "hidden" in the code), and the error control mechanism of the integrator is relied on to deal with them. It is well known that numerically integrating systems containing hidden discontinuities is inefficient and sometimes results in integration failures [12]. In the worst case, ignoring discontinuities can lead to incorrect results that may escape the modeler's attention [6]. The situation is far worse and not yet documented in the literature for parametric sensitivity analysis of systems containing hidden discontinuities. Because the sensitivities will often jump at the discontinuities, if the numerical integration is not stopped and the jump computed explicitly [8], the computed sensitivity trajectories will generally be incorrect.

The purpose of this paper is to illustrate the implications of performing parametric sensitivity analysis of systems containing hidden discontinuities and describe automatic techniques that can be applied to code in order to perform parametric sensitivity analysis with hybrid systems correctly.

2. Formalism.

The following hybrid system formalism, adopted from [1, 8], will be used in this paper. State space is divided into a set of modes, $S = \cup_{k=1}^{n_m} S_k$, where each mode S_k is characterized by the following:

1. A set of variables $\{\dot{x}^{(k)}(p,t), x^{(k)}(p,t), y^{(k)}(p,t), u^{(k)}(p,t), p, t\}$, where $x^{(k)}(p,t)$, $\dot{x}^{(k)}(p,t)$ are the differential variables and their time derivatives, which are functions of p and t in an $n_x^{(k)}$-dimensional function space, $y^{(k)}(p,t)$ are the algebraic variables, which are functions of p and t in an $n_y^{(k)}$-dimensional function space, $u^{(k)}(p,t)$ are the controls (or inputs), which are functions of p and t in an $n_u^{(k)}$-dimensional function space, $p \in \mathbb{R}^{n_p}$ are the time invariant parameters, and $t \in \mathbb{R}$ is time, the independent variable.
2. A set of equations, $f^{(k)}(\dot{x}^{(k)}, x^{(k)}, y^{(k)}, u^{(k)}, p, t) = 0$, where $f^{(k)} : \mathbb{R}^{n_x^{(k)}} \times \mathbb{R}^{n_x^{(k)}} \times \mathbb{R}^{n_y^{(k)}} \times \mathbb{R}^{n_u^{(k)}} \times \mathbb{R}^{n_p} \times \mathbb{R} \longrightarrow \mathbb{R}^{n_x^{(k)}} \times \mathbb{R}^{n_y^{(k)}}$.
3. A set of transitions from one mode to another (possibly the same mode), $J^{(k)}$, where each transition is characterized by the following:
   (a) Transition conditions $L_j^{(k)}(\dot{x}^{(k)}, x^{(k)}, y^{(k)}, u^{(k)}, p, t)$, $j \in J^{(k)}$, defining the transition times.
   (b) Transition functions $T_j^{(k)}(\dot{x}^{(k+1)}, x^{(k+1)}, y^{(k+1)}, u^{(k+1)}, \dot{x}^{(k)}, x^{(k)}, y^{(k)}, u^{(k)}, p, t) = 0$, $j \in J^{(k)}$, a system of equations that maps the final values of the variables in the current mode to the initial values in the next mode. (Initial conditions are a special case of these transition functions.) The number of transition functions is determined by the dynamic degrees of freedom of the DAE in the new mode.

It is assumed that the transition conditions, $L_j^{(k)}(\dot{x}^{(k)}, x^{(k)}, y^{(k)}, u^{(k)}, p, t)$, are logical expressions composed of one or more real-valued relational expressions involving relational operators <, ≤, >, or ≥. For example, the logical expression $(x_1 \geq 5) \vee (x_2 > 2 \wedge x_2 \leq 5)$ contains the relational expressions $x_1 \geq 5$, $x_2 > 2$, and $x_2 \leq 5$. Putting these relational expressions into the form

$$g \geq (>) 0 \tag{2.1}$$

defines *discontinuity functions.* The logical proposition of the transition condition may switch value when one or more discontinuity functions cross zero. A switching event is defined by the earliest time at which one of the transition conditions becomes true. Of course, a model may contain more general logical conditions (e.g., integral expressions or expressions involving the equality operator), and these are described below.

This paper focuses on the case where

$$\operatorname{rank}\begin{pmatrix} \dfrac{\partial f^{(k)}}{\partial \dot{x}^{(k)}} & \dfrac{\partial f^{(k)}}{\partial y^{(k)}} \end{pmatrix} = n_x^{(k)} + n_y^{(k)} \tag{2.2}$$

for all k and t. This is sufficient for a differentiation index less than or equal to one.

3. Algorithm. Several approaches have been developed for hybrid simulation, ranging from modifying the integrator itself to detect and handle discontinuities [9] to replacing the discrete aspects of the model with smooth approximations. The approach used in this work is based on the algorithm described in [15]. In this approach, the DAE of the current mode, $f^{(k)}$, is augmented with the discontinuity functions associated with the mode's transition conditions, $L_j^{(k)}$, $j \in J^{(k)}$. For example, let $g_{l,j}^{(k)} : \mathbb{R}^{n_x^{(k)}} \times \mathbb{R}^{n_x^{(k)}} \times \mathbb{R}^{n_y^{(k)}} \times \mathbb{R}^{n_u^{(k)}} \times \mathbb{R}^{n_p} \times \mathbb{R} \longrightarrow \mathbb{R}$ denote the lth discontinuity function of transition condition $L_j^{(k)}$, $j \in J^{(k)}$, where $l = 1, \dots, n_{d,j}^{(k)}$ (where $n_{d,j}^{(k)}$ is the number of discontinuity functions in the jth transition condition). The augmented DAE in mode S_k is given by

$$F^{(k)}(\dot{x}^{(k)}, x^{(k)}, y^{(k)}, \delta^{(k)}, u^{(k)}, p, t) = \begin{pmatrix} f^{(k)} \\ \delta^{(k)} - g^{(k)} \end{pmatrix}, \tag{3.1}$$

where $g^{(k)}$ denotes the vector of the discontinuity functions of all transition conditions in the current mode and $\delta^{(k)}$ is the vector of *discontinuity variables.* Augmenting the original DAE with these additional explicit equations (referred to as *discontinuity equations*) places the discontinuity functions under integration error control. The additional variables added for the discontinuity equations, $\delta^{(k)}$, are algebraic variables and may be appended to the original algebraic variable vector:

$$y \equiv \begin{pmatrix} y \\ \delta \end{pmatrix}.$$

Although including the discontinuity equations increases the size of the model, because the system block decomposes, very little additional computational effort is required to integrate the system [15]. Condition (2.2) along with the assumption the discontinuity functions do not contain time derivatives of y ensures

$$\operatorname{rank}\begin{pmatrix} \dfrac{\partial F^{(k)}}{\partial \dot{x}^{(k)}} & \dfrac{\partial F^{(k)}}{\partial y^{(k)}} \end{pmatrix} = n_x^{(k)} + n_y^{(k)}, \tag{3.2}$$

for all k, where $x^{(k)}$ and $y^{(k)}$ are the differential and algebraic variables, respectively, of the augmented system. (That is, $n_y^{(k)}$ includes the additional discontinuity variables.)

At each integration step, the interpolating polynomials associated with the discontinuity variables are searched for zero crossings using root exclusion tests based on interval arithmetic. If an integrator based on the predictor-corrector method is employed, these interpolating polynomials will, in general, be readily available. Otherwise, polynomials can be fitted to the discontinuity variables during the integration. Using a root exclusion test based on interval arithmetic is efficient and guarantees that the correct transition time (i.e., the earliest) will be identified. The transition time is *polished*, and consistent states are determined by solving the DAE augmented with the discontinuity function triggering the event. (The transition time computed from the interpolating polynomials may not be sufficiently close to the true transition time, resulting in "discontinuity sticking" where the event just found is immediately triggered again [15]. Consequently, the value obtained from the interpolation must be polished by augmenting the DAE with the discontinuity function which triggered the event and solving this augmented system for t^*, $\dot{x}^{(k)}$, $x^{(k)}$, and $y^{(k)}$ where t^* is "close" to the actual transition time.) The model is then switched into the next mode, where the new DAE system is consistently initialized using the appropriate transition function and integration is resumed.

The situation is slightly more complicated when performing parametric sensitivity analysis with hybrid systems. In the kth mode, the augmented DAE and sensitivity equations form an $(n_x^{(k)} + n_y^{(k)})(n_p + 1)$ system given by

$$\begin{aligned}
&F^{(k)}(\dot{x}^{(k)}, x^{(k)}, y^{(k)}, u^{(k)}, p, t) = 0, \\
&\frac{\partial F^{(k)}}{\partial \dot{x}^{(k)}} \dot{s}_{x,1} + \frac{\partial F^{(k)}}{\partial x^{(k)}} s_{x,1} + \frac{\partial F^{(k)}}{\partial y^{(k)}} s_{y,1} = -\left(\frac{\partial F^{(k)}}{\partial u^{(k)}} \frac{\partial u^{(k)}}{\partial p_1} + \frac{\partial F^{(k)}}{\partial p_1} \right), \\
&\vdots \\
&\frac{\partial F^{(k)}}{\partial \dot{x}^{(k)}} \dot{s}_{x,n_p} + \frac{\partial F^{(k)}}{\partial x^{(k)}} s_{x,n_p} + \frac{\partial F^{(k)}}{\partial y^{(k)}} s_{y,n_p} = -\left(\frac{\partial F^{(k)}}{\partial u^{(k)}} \frac{\partial u^{(k)}}{\partial p_{n_p}} + \frac{\partial F^{(k)}}{\partial p_{n_p}} \right),
\end{aligned} \tag{3.3}$$

where $s_{x,i}^{(k)} \equiv \partial x^{(k)}/\partial p_i$, $s_{y,i}^{(k)} \equiv \partial y^{(k)}/\partial p_i$, and $\dot{s}_{x,i}^{(k)} = \partial s_{x,i}^{(k)}/\partial t = \partial \dot{x}^{(k)}/\partial p_i$. Although this system may be quite large, algorithms exist that exploit the structure for efficient solution [14, 7].

In addition to computing consistent initial values for $\dot{x}$, x, and y in the new mode, jumps in the sensitivities must also be computed. However, as shown in [8], these jumps can be readily computed by differentiating the original DAE, discontinuity function, and transition functions associated with the transfer of the state into the new mode. The sensitivity of the transition time with respect to parameter p_i is given by

$$\begin{aligned}
&\frac{\partial \tilde{g}_{k+1}^{(k)}}{\partial \dot{x}^{(k)}} \left(\dot{s}_{x,i}^{(k)} + \ddot{x}^{(k)} \frac{\partial t}{\partial p_i} \right) + \frac{\partial \tilde{g}_{k+1}^{(k)}}{\partial x^{(k)}} \left(s_{x,i}^{(k)} + \dot{x}^{(k)} \frac{\partial t}{\partial p_i} \right) + \frac{\partial \tilde{g}_{k+1}^{(k)}}{\partial y^{(k)}} \left(s_{y,i}^{(k)} + \dot{y}^{(k)} \frac{\partial t}{\partial p_i} \right) \\
&\quad + \frac{\partial \tilde{g}_{k+1}^{(k)}}{\partial u^{(k)}} \left(\frac{\partial u^{(k)}}{\partial p_i} + \frac{\partial u^{(k)}}{\partial t} \frac{\partial t}{\partial p_i} \right) + \frac{\partial \tilde{g}_{k+1}^{(k)}}{\partial p_i} + \frac{\partial \tilde{g}_{k+1}^{(k)}}{\partial t} \frac{\partial t}{\partial p_i} = 0,
\end{aligned} \tag{3.4}$$

where $\tilde{g}_{k+1}^{(k)}$ is the discontinuity function that triggers the event. The equation above is solved for $\partial t/\partial p_i$. The required elements of $\dot{x}$, $\dot{y}$, and $\ddot{x}$ in the equation above can be computed from the first and higher order derivatives of the DAE. The jump in sensitivities can then be computed by solving the following linear system:

$$
\begin{aligned}
&\begin{bmatrix} \frac{\partial T^{(k)}_{k+1}}{\partial \dot{x}^{(k+1)}} & \frac{\partial T^{(k)}_{k+1}}{\partial x^{(k+1)}} & \frac{\partial T^{(k)}_{k+1}}{\partial y^{(k+1)}} \\ \frac{\partial F^{(k+1)}}{\partial \dot{x}^{(k+1)}} & \frac{\partial F^{(k+1)}}{\partial x^{(k+1)}} & \frac{\partial F^{(k+1)}}{\partial y^{(k+1)}} \end{bmatrix}
\begin{bmatrix} \frac{\partial \dot{x}^{(k+1)}}{\partial p_i} \\ \frac{\partial x^{(k+1)}}{\partial p_i} \\ \frac{\partial y^{(k+1)}}{\partial p_i} \end{bmatrix} \\
&= - \begin{bmatrix} \frac{\partial T^{(k)}_{k+1}}{\partial \dot{x}^{(k+1)}} & \frac{\partial T^{(k)}_{k+1}}{\partial x^{(k+1)}} & \frac{\partial T^{(k)}_{k+1}}{\partial y^{(k+1)}} \\ \frac{\partial F^{(k+1)}}{\partial \dot{x}^{(k+1)}} & \frac{\partial F^{(k+1)}}{\partial x^{(k+1)}} & \frac{\partial F^{(k+1)}}{\partial y^{(k+1)}} \end{bmatrix}
\begin{bmatrix} \frac{\partial \dot{x}^{(k+1)}}{\partial t} \\ \frac{\partial x^{(k+1)}}{\partial t} \\ \frac{\partial y^{(k+1)}}{\partial t} \end{bmatrix} \frac{\partial t}{\partial p_i} \\
&\quad - \begin{bmatrix} \frac{\partial T^{(k)}_{k+1}}{\partial \dot{x}^{(k)}} & \frac{\partial T^{(k)}_{k+1}}{\partial x^{(k)}} & \frac{\partial T^{(k)}_{k+1}}{\partial y^{(k)}} & \frac{\partial T^{(k)}_{k+1}}{\partial u^{(k)}} & \frac{\partial T^{(k)}_{k+1}}{\partial u^{(k+1)}} & \frac{\partial T^{(k)}_{k+1}}{\partial p_i} & \frac{\partial T^{(k)}_{k+1}}{\partial t} \\ 0 & 0 & 0 & 0 & \frac{\partial F^{(k+1)}}{\partial u^{(k+1)}} & \frac{\partial F^{(k+1)}}{\partial p_i} & \frac{\partial F^{(k+1)}}{\partial t} \end{bmatrix} \\
&\quad \times \begin{bmatrix} \frac{\partial \dot{x}^{(k)}}{\partial p_i} + \frac{\partial \dot{x}^{(k)}}{\partial t}\frac{\partial t}{\partial p_i} \\ \frac{\partial x^{(k)}}{\partial p_i} + \frac{\partial x^{(k)}}{\partial t}\frac{\partial t}{\partial p_i} \\ \frac{\partial y^{(k)}}{\partial p_i} + \frac{\partial y^{(k)}}{\partial t}\frac{\partial t}{\partial p_i} \\ \frac{\partial u^{(k)}}{\partial p_i} + \frac{\partial u^{(k)}}{\partial t}\frac{\partial t}{\partial p_i} \\ \frac{\partial u^{(k+1)}}{\partial p_i} + \frac{\partial u^{(k+1)}}{\partial t}\frac{\partial t}{\partial p_i} \\ I \\ \frac{\partial t}{\partial p_i} \end{bmatrix}.
\end{aligned}
\tag{3.5}
$$

Additional details about the transfer of sensitivities at state events, including formal existences proofs, are presented in [8].

The hybrid system parametric sensitivity analysis algorithm can be summarized with the pseudocode shown in Figure 1.

A large amount of additional information must be provided by the modeler when using the algorithm described above. First, the modeler must be able to lock the model into the current mode so that the integrator always evaluates a smooth function. For example, if the model code contains IF statements, then the conditional branching must be locked to the current mode, regardless of the values of the logical expressions associated with the IF statements. Second, all of the discontinuity functions contained in the model code must be identified and put into an explicit form. Finally, additional partial derivative information other than that required for smooth integration must be provided (e.g., partial derivatives with respect to discontinuity functions and transition functions). Requiring the modeler to provide this information is a great burden, a reason that contributes greatly to why discontinuities are typically not handled explicitly. However, this additional information can be generated automatically with minimal user intervention using code analysis and transformation techniques similar to those used in automatic differentiation. The paragraphs below describe how this information can be generated when the original model is coded in the form of a program written in some procedural programming language such as C or Fortran.

Automatic, or algorithmic, differentiation (AD) is a technique for obtaining values for analytical derivatives of functions, usually in the form of a computer program (e.g., a collection of one or more subroutines and functions written in C or Fortran). AD is based on applying the chain-rule to the sequence of elementary operations involved in the computation of the dependent variables from the independent variables. (Several

```
Hybrid Sensitivity Algorithm
Compute consistent initial conditions in initial mode
Compute initial sensitivities
while (integration not finished) do
    Take integration step
    Search discontinuity function interpolating polynomials
            for zero crossings
    if (discontinuity function crosses zero and
            transition condition becomes true) then
            Compute event time and consistent values for the states
               and time derivatives in the current mode
            Solve appropriate transition function to transfer final
               conditions in current mode to initial conditions
               in next mode
            Compute all required derivative information (i.e., required
               elements of $\dot{x}$, $\dot{y}$, and $\ddot{x}$)
            Compute sensitivity of event time using (3.4)
            Compute jump in sensitivities using (3.5)
    end if
end
```

Fig. 1. *Pseudocode for hybrid sensitivity algorithm.*

variants of AD exist depending on the how the chain-rule is applied.) The original code is then modified and augmented (either by generating new code or using the *operator overloading* features of many modern programming languages) with additional code to perform the chain-rule operations and compute the values of the partial derivatives of the dependent variables with respect to the independent variables. A detailed description of AD is beyond the scope of this paper, and additional information can be found in [16, 10, 3, 11].

AD tools that generate new code for derivative computation analyze the original source code to identify the dependence of the dependent variables on the independent variables, apply the chain-rule to the elementary operations, and construct new code that computes derivatives of the elementary operations and *accumulates* values of the partial derivatives of the dependent variables. The code analysis and transformation techniques used to construct new code for derivative evaluation can also be extended to generate a wide variety of information, in particular the information required for performing the algorithm for parametric sensitivity analysis of hybrid systems described above. The original source is analyzed and searched for discontinuities. The source is then augmented with additional code that records the current mode. For example, if an IF statement is identified in the original source, then the new code records the current clause of this IF statement (i.e., whether the logical expression evaluates to true or false) and records this value in a bit vector. Similarly, if a non-smooth intrinsic function (e.g., MIN and MAX) is identified, then the currently active branch is recorded in the bit vector. On subsequent model evaluations, the bit vector is analyzed to determine the conditional branching followed on previous evaluations, and this is used to determine the current discrete mode, rather than the actual values of the logical expressions associated with IF statements and other

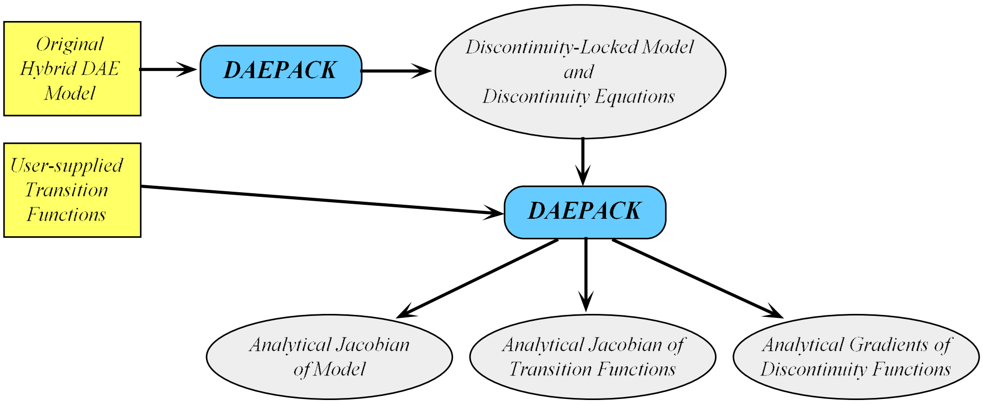

Original Hybrid DAE Model
DAEPACK
Discontinuity-Locked Model and Discontinuity Equations
User-supplied Transition Functions
DAEPACK
Analytical Jacobian of Model
Analytical Jacobian of Transition Functions
Analytical Gradients of Discontinuity Functions

FIG. 2. *Automatic generation of additional information required for hybrid parametric sensitivity analysis with DAEPACK. The squares denote the code that must be supplied by the user. The ovals denote code generated automatically by DAEPACK.*

discontinuities. Using a bit vector to record the current mode typically results in very little additional memory and computational cost required to perform a *locked* model evaluation. In addition to recording the current mode, the original source can also be modified to evaluate the discontinuity functions in the model and augment the original DAE with the discontinuity equations shown in (3.1). To reiterate, all of this can be performed automatically with minimal user intervention. As mentioned above, the transition conditions may contain logical expressions other than real-valued relational expressions (the expressions used to construct discontinuity functions). Although discontinuity functions are not created for these types of logical expressions (e.g., $t = 0$ and $i > 4$ where i is an integer), the new code can be modified to return a flag indicating events containing these types of transition conditions have been triggered. However, unlike events caused solely by zero crossings of discontinuity functions, these other events will be identified only at mesh points in the numerical integration. Once the new model, augmented with the discontinuity equations and modified to allow for locked model evaluations, has been generated, standard AD techniques can be applied to generate the derivatives required for hybrid sensitivity analysis.

The approach described above has been implemented in a software library called DAEPACK [18, 17]. The DAEPACK library consists of two parts: components for analyzing general Fortran code and generating automatically new code for evaluating analytical derivatives, sparsity patterns, discontinuity-locked models, etc. and numeric components which exploit the automatically generated information for performing numerical calculations efficiently, robustly, and correctly. Figure 2 contains a diagram showing the additional code required for hybrid parametric sensitivity analysis generated automatically from the users code representing the original hybrid DAE model. Figure 3 contains a diagram showing how this automatically generated code is used with other DAEPACK numerical components for performing a hybrid parametric sensitivity calculation.

This section describes an automated technique for computing the parametric sensitivities of a hybrid system by explicitly handling the events and transfer of states and sensitivities across these events. Alternatively, the user may compute the sensitivities of a hybrid system by finite differences. For example, using forward finite differences, the sensitivity of a state variable, e.g., $x(t, p)$, at $t = t^*$ is approximately

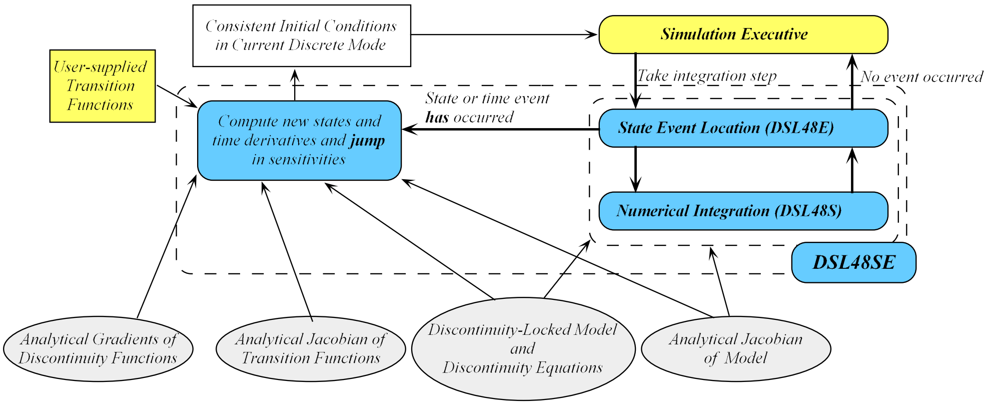

Consistent Initial Conditions in Current Discrete Mode
Simulation Executive
User-supplied Transition Functions
Take integration step
No event occurred
State or time event has occurred
Compute new states and time derivatives and jump in sensitivities
State Event Location (DSL48E)
Numerical Integration (DSL48S)
DSL48SE
Analytical Gradients of Discontinuity Functions
Analytical Jacobian of Transition Functions
Discontinuity-Locked Model and Discontinuity Equations
Analytical Jacobian of Model

FIG. 3. *Hybrid parametric sensitivity analysis algorithm with DAEPACK.*

the difference between this point on the trajectory evaluated at p and this point on the trajectory evaluated at $p + \delta p$, divided by δp. That is,

$$s(t^*, p) \equiv \frac{\partial x(t^*, p)}{\partial p} \approx \frac{x(t^*, p + \delta p) - x(t^*, p)}{\delta p}.$$

There are, however, a number of problems with this. First, the numerical integration of the hybrid system must be performed reliably. If special provisions must be taken to ensure correct hybrid simulation (such as the approach described above), then the additional work required to transfer the sensitivities at the event should be performed. Other problems are cost and accuracy. In order to compute the sensitivity trajectories with respect to n_p parameters, $n_p + 1$ numerical integrations must be performed. Furthermore, each of these integrations should be performed with identical stepsize histories [13]. If an adaptive stepsize algorithm is used, then this can be quite a difficult modification to a third-party library code. Also, the user must ensure that the stepsize history selected is appropriate for each of the $n_p + 1$ integrations. Even if this is done, all the problems associated with computing partial derivatives of algebraic functions with finite differences exist, including selection of appropriate perturbation and loss of accuracy due to round-off error. If the numerical integrations are not performed with same stepsize history, then they must be performed at a much higher tolerance than the accuracy desired in the sensitivities. As described in [13], the error in the finite difference approximated sensitivities due to truncation error is $O(|\delta p|)$, and the error due to the fact the integration is performed with limited accuracy is $O(\epsilon/|\delta p|)$, where δp is the parameter perturbation and ϵ is the numerical integration tolerance. Consequently, if we desire sensitivity trajectories with accuracy δ, we must perform each of the n_p+1 numerical integrations with accuracy δ^2. Unfortunately, the presence of discontinuities in the hybrid system may limit the accuracy of the numerical integration due to calculation failures unless these events are handled explicitly. Even if the numerical integrations can be performed reliably, the tighter tolerances required can significantly increase the cost of the overall calculation. These issues are illustrated with a numerical example in the following section.

The following section contains example problems illustrating the implications of ignoring discontinuities during sensitivity calculation and how the calculation is performed properly using DAEPACK.

4. Examples. This section contains two example problems illustrating the approach described above. The ideas presented in this paper were implemented in the software library DAEPACK [18]. DAEPACK is divided into two main libraries, one containing components that generate automatically symbolic information (e.g., analytical derivatives and sparsity patterns) from Fortran source code and another containing numerical components which exploit this symbolic information. In particular, DAEPACK contains a component that analyzes the user's Fortran source code representing a hybrid system model and writes new code that computes the discontinuity functions and allows the user to perform locked model evaluations. In addition, a component is provided which analyzes Fortran source code and constructs new code for evaluating partial derivatives using AD. The user must simply provide a list of the dependent and independent variables. Three numerical codes in DAEPACK of interest in this paper are DSL48S, DSL48E, and DSL48SE. DSL48S is a large-scale, sparse DAE integrator based on DASSL [4] and employs the parametric sensitivity analysis algorithm described in [7]. DSL48E implements the state event location algorithm described in [15]. DSL48SE combines both DSL48S and DSL48E and computes automatically the sensitivity jump at the state events. All of the information required to perform the hybrid sensitivity calculation properly can be generated with DAEPACK. All the user must provide is the Fortran source code representing the hybrid system model of interest.

The first example is a single ODE where the transition condition, $x^3-5x^2+7x \leq p$, is a function of the time invariant parameter p:

$$\dot{x} = \begin{cases} 4-x & : \quad x^3-5x^2+7x \leq p, \\ 10-2x & : \quad \text{otherwise}, \end{cases} \tag{4.1}$$

where $p = 2.9$. This model was coded into a Fortran subroutine using a normal IF statement to represent the discontinuity. The symbolic components of DAEPACK were applied to this Fortran model to generate new code for the discontinuity-locked model and all of the necessary derivatives. The parametric sensitivity calculation was performed twice, once using only DSL48S (where the discontinuity was hidden) and a second time using DSL48SE to handle properly the discontinuity. The initial condition for this model is

$$x(0) = 0.$$

The transfer functions for this hybrid system are simply $x^{(2)} - x^{(1)} = 0$. Figure 4 shows the state variable, x, and sensitivity trajectories for both cases. Clearly, the sensitivity is wrong (quantitatively and qualitatively) if the discontinuity is not handled properly. In fact, the incorrect sensitivity in Figure 4 would seem to indicate that x is not sensitive at all to the value of p, which it clearly is. Unfortunately, this model is solved without an apparent problem when the discontinuity is ignored; thus, the incorrect sensitivity can easily escape the modeler's attention.

Suppose the user wishes to compute the hybrid sensitivities using finite differences (see the discussion in the previous section). Further, suppose the user naively performs the numerical integrations using an integration tolerance of 10^{-6}. Figure 5 contains a plot of three sensitivity trajectories for the hybrid ODE (4.1) in the vicinity of the first event. The solid line in Figure 5 (the trajectory with a clear "jump") is the sensitivity trajectory computed using the approach described in this paper and has an accuracy of 10^{-6}. The dashed line below this trajectory was computed using finite differences with a parameter perturbation of 10^{-5}. The dotted line above the solid lined trajectory was computed using finite differences with a parameter perturbation

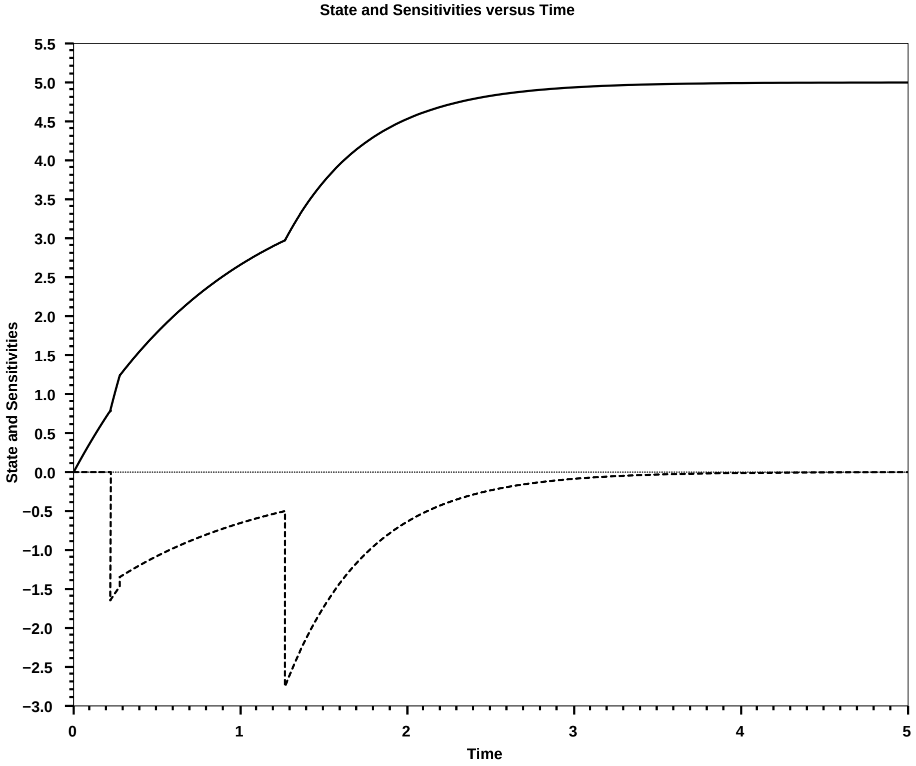


FIG. 4. *State and sensitivity trajectories for Example* 1. *The heavy solid line is the state trajectory, the thin dotted line is the sensitivity trajectory computed with hidden discontinuities, and the dashed line exhibiting jumps is the true sensitivity trajectory.*

of 10^{-4}. Notice that in both of the finite difference trajectories, a sharp, but smooth, transient is computed rather than a jump. The duration of this transient is equal to the shift in time of the event caused by the perturbation of the parameter. Figure 6 contains a plot of the differences between the sensitivity trajectory computed using the approach described in this paper and the trajectories computed by finite differences with various parameter perturbations. (All integrations were performed with a tolerance of 10^{-6}.) The thin solid line is the difference in the finite difference trajectory computed using a parameter perturbation of 10^{-6}. The dashed line is the difference in the finite difference trajectory computed using a parameter perturbation of 10^{-5}. The dotted line is the difference in the finite difference trajectory computed using a parameter perturbation of 10^{-4}. This figure precisely illustrates the description in [13]: the error in the finite difference trajectories is $O(\epsilon/|\delta p|)$. When the parameter perturbation is of the order of the integration tolerance, then the finite difference computed trajectories are completely inaccurate. In this example, we were able to select a parameter perturbation large enough to obtain reasonably accurate sensitivity trajectories for a given integration tolerance. However, if we were interested in the values of the sensitivities near the event, then we would have to choose a much smaller parameter perturbation. Truncation error considerations may also require the use of a much smaller parameter perturbation. If either of these were the case, then we would have to choose a much tighter integration tolerance. In this simple example, the tightest integration tolerance possible was approximately 10^{-13}. Thus, it is quite

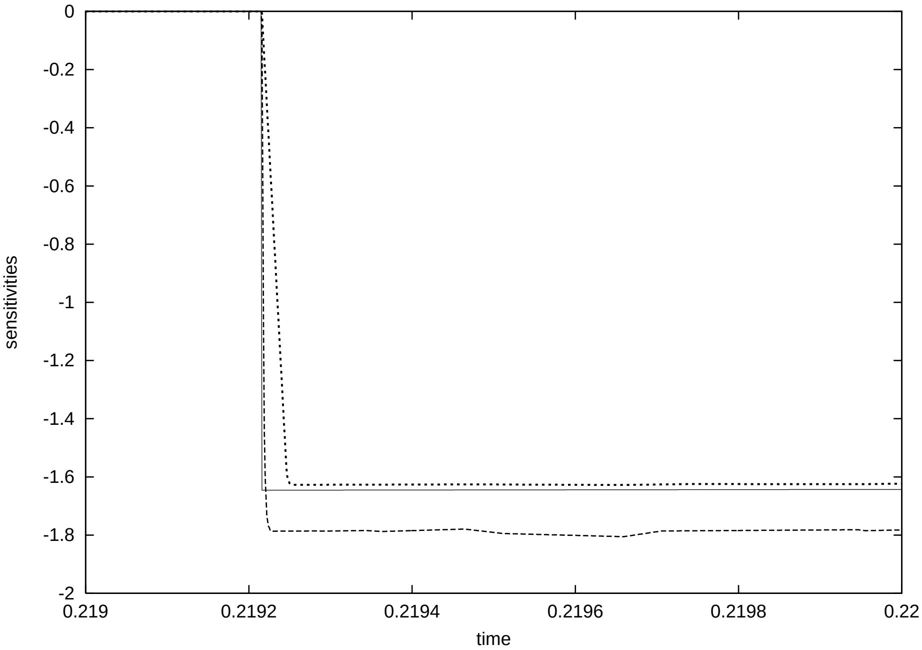


FIG. 5. *Hybrid parametric sensitivity trajectories. The thin solid line is the sensitivity trajectory computed explicitly with a tolerance of* 10^{-6}*. The long dashed line (below solid line trajectory) is the sensitivity trajectory computed using finite differences with a parameter perturbation of* 10^{-5} *and integration tolerance of* 10^{-6}*. The dashed line (slightly above solid line trajectory) is the sensitivity trajectory computed using finite differences with a parameter perturbation of* 10^{-4} *and integration tolerance of* 10^{-6}*.*

possible that the numerical integrations may not be able to be performed at the necessary level of accuracy. Computing derivatives of algebraic functions reliably with finite differences is often a difficult task. As illustrated in this example, using finite differences to compute parametric sensitivities of hybrid dynamic systems is often a far more difficult task.

The second example is a small circuit simulation model adapted from [5]:

$$f_1 = x_1 + x_2 - x_3, \tag{4.2}$$

$$f_2 = r_3(x_1 + x_2) + l_3(\dot{x}_1 + \dot{x}_2) - v_3, \tag{4.3}$$

$$f_3 = 100\cos(100\pi t) - v_1, \tag{4.4}$$

$$f_4 = -v_1 - v_2, \tag{4.5}$$

$$f_5 = v_1 - v_3 - z_3, \tag{4.6}$$

$$f_6 = v_2 - v_3 - z_4, \tag{4.7}$$

(4.8)
$$f_7 = \begin{cases} (v_1 - p_1x_1)/p_2 - \dot{x}_1 & : & (x_1 \geq 0 \vee v_1 \geq v_3) \wedge (x_2 \leq 0 \wedge v_2 \leq v_3), \\ \dot{x}_1 & : & (x_2 \geq 0 \vee v_2 \geq v_3) \wedge (x_1 \leq 0 \wedge v_1 \leq v_3), \\ p_5v_1 + p_6v_2 + p_7x_1 + p_8x_2 - \dot{x}_1 & : & \text{otherwise}, \end{cases}$$

(4.9)
$$f_8 = \begin{cases} \dot{x}_2 & : & (x_1 \geq 0 \vee v_1 \geq v_3) \wedge (x_2 \leq 0 \wedge v_2 \leq v_3), \\ (v_2 - p_3x_2)/p_4 - \dot{x}_2 & : & (x_2 \geq 0 \vee v_2 \geq v_3) \wedge (x_1 \leq 0 \wedge v_1 \leq v_3), \\ p_9v_1 + p_{10}v_2 + p_{11}x_1 + p_{12}x_2 - \dot{x}_2 & : & \text{otherwise}, \end{cases}$$

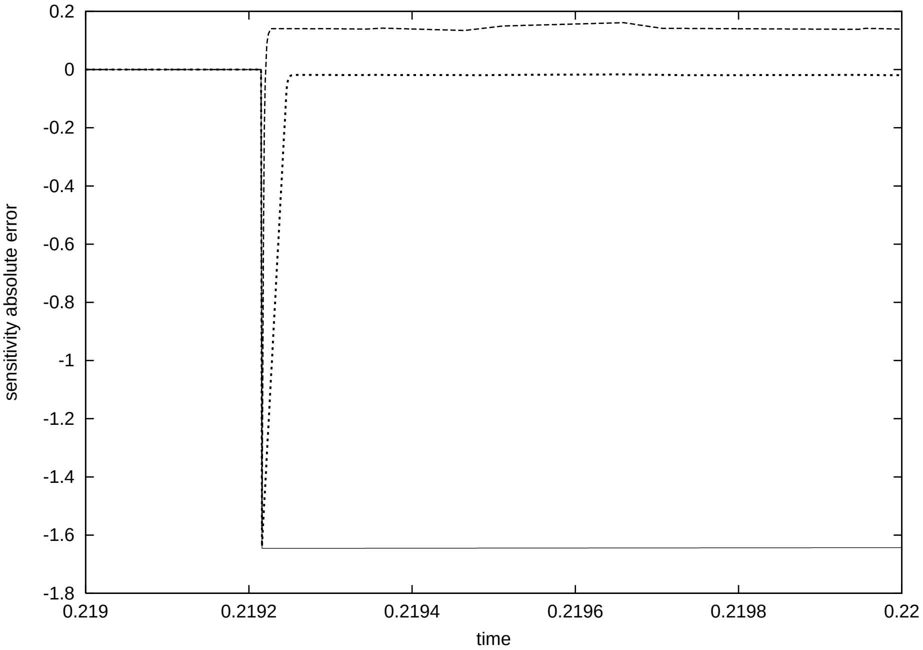


FIG. 6. *Error in hybrid parametric sensitivity trajectories. This figure contains a plot of the difference between the sensitivity trajectory computed explicitly and the trajectories obtained using finite differences. All integrations are performed with a tolerance of* 10^{-6}*. The thin solid line is the difference in the finite difference trajectory computed using a parameter perturbation of* 10^{-6}*, the dashed line is the difference in the finite difference trajectory computed using a parameter perturbation of* 10^{-5}*, and the dotted line is the difference in the finite difference trajectory computed using a parameter perturbation of* 10^{-4}*.*

where $p_1 = 12$, $p_2 = 0.24$, $p_3 = 12$, $p_4 = 0.24$, $p_5 = 13.64$, $p_6 = -11.64$, $p_7 = -50$, $p_8 = 0$, $p_9 = -11.64$, $p_{10} = 13.64$, $p_{11} = 0$, and $p_{12} = -50$. The initial conditions for this model are

$$x_1(0) = 0,$$
$$x_2(0) = 0,$$

and the transition functions are $x_1^{(k+1)} - x_1^{(k)} = 0$ and $x_2^{(k+1)} - x_2^{(k)} = 0$ for all transitions. Again, this model was coded into a Fortran subroutine, and DAEPACK was used to generate the additional information (i.e., the discontinuity-locked model and all partial derivatives). Unlike the first example, this model could not be numerically integrated beyond the second discontinuity when the discontinuities were not handled explicitly. Figure 7 contains the sensitivities of variables x_1, x_2, and x_3 with respect to time invariant parameter p_5. A plot of the state trajectories can be found elsewhere in the literature [15].

5. Conclusions. Parametric sensitivity analysis provides valuable information required to understand a model and is used in a variety of calculations. Unfortunately, applying standard numerical codes for parametric sensitivity analysis to models containing discontinuities can lead to disastrous consequences. This is particularly true for legacy models. Existing models, containing discontinuous relationships, that yielded correct results when performing numerical integration will generally fail or

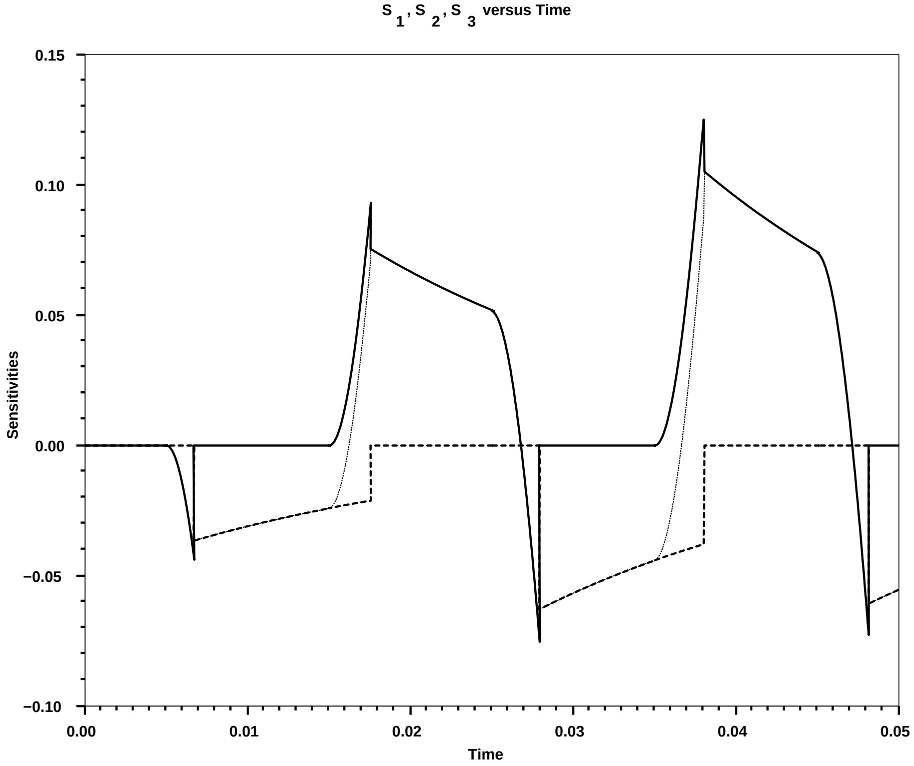


FIG. 7. *Sensitivity trajectories ($s_1 = \partial x_1/\partial p_5$, $s_2 = \partial x_2/\partial p_5$, and $s_3 = \partial x_3/\partial p_5$) for Example 2. The solid line is s_1, the dashed line is s_2, and the thin dotted line is s_3.*

produce incorrect results when performing parametric sensitivity analysis. Fortunately, by extending the techniques of automatic differentiation, code analysis and transformation can be used to automatically generate the information required to perform proper parametric sensitivity analysis.

Acknowledgment. The authors would like to acknowledge Elena Chekhova for the example that brought to our attention the problems of hidden discontinuities in sensitivity analysis.

REFERENCES

[1] A. Back, J. Guckenheimer, and M. Meyers, *A dynamical simulation facility for hybrid systems*, in Hybrid Systems, R. L. Grossman, Lecture Notes in Comput. Sci., 736, A. Nerode, A. P. Ravn, and H. Rischel, eds., Springer-Verlag, New York, 1993, pp. 255–267.

[2] L. G. Birta, T. I. Ören, and K. L. Kettenis, *A robust procedure for discontinuity handling in continuous system simulation*, Trans. Soc. Comput. Sim., 2 (1985), pp. 189–205.

[3] C. Bischof, A. Carle, G. Corliss, A. Griewank, and P. Hovland, *ADIFOR – Generating derivative codes from Fortran programs*, Scientific Programming, 1 (1992), pp. 11–29.

[4] K. E. Brenan, S. L. Campbell, and L. R. Petzold, *Numerical Solution of Initial Value Problems in Differential–Algebraic Equations*, SIAM, Philadelphia, PA, 1995.

[5] M. B. Carver, *Efficient integration over discontinuities in ordinary differential equation simulations*, Math. Comput. Simulation, 20 (1978), pp. 190–196.

[6] F. E. Cellier, *Combined Continuous/Discrete System Simulation by Use of the Digital Computer: Techniques and Tools*, Ph.D. thesis, Swiss Federal Institute of Technology, Zurich, Switzerland, 1979.

[7] W. F. Feehery, J. E. Tolsma, and P. I. Barton, *Efficient sensitivity analysis of large-scale differential-algebraic systems*, Appl. Numer. Math., 25 (1997), pp. 41–54.

[8] S. Galán, W. F. Feehery, and P. I. Barton, *Parametric sensitivity functions for hybrid discrete/continuous systems*, Appl. Numer. Math., 31 (1999), pp. 17–47.

[9] C. W. Gear and O. Osterby, *Solving ordinary differential equations with discontinuities*, ACM Trans. Math. Software, 10 (1984), pp. 23–44.

[10] A. Griewank, *On automatic differentiation*, in Mathematical Programming: Recent Developments and Applications, M. Iri and K. Tanabe, eds., Kluwer Academic Publishers, Dordrecht, The Netherlands, 1989, pp. 83–108.

[11] A. Griewank, *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, SIAM, Philadelphia, PA, 2000.

[12] J. L. Hay and A. W. J. Griffin, *Simulation of discontinuous dynamical systems*, in Proceedings of the 9th IMACS Conference on Simulation of Systems, 1979, L. Dekker, G. Savastano, and G. C. Vansteenkiste, eds., North-Holland, Amsterdam, 1980, pp. 79–87.

[13] M. Kiehl, *Sensitivity Analysis of ODEs and DAEs—Theory and Implementation Guide*, Technical Report TUM–M9801, TUM Fakultät Für Mathematik, Munich, Germany, 1998.

[14] T. Maly and L. R. Petzold, *Numerical methods and software for sensitivity analysis of differential–algebraic systems*, Appl. Numer. Math., 20 (1996), pp. 57–79.

[15] T. Park and P. I. Barton, *State event location in differential algebraic models*, ACM Trans. Model. Comput. Sim., 6 (1996), pp. 137–165.

[16] L. B. Rall, *Automatic Differentiation: Techniques and Applications*, Lecture Notes in Comput. Sci., 120, Springer-Verlag, Berlin, 1981.

[17] J. E. Tolsma and P. I. Barton, *DAEPACK: A Combined Symbolic and Numeric Library for General Numerical Calculations*, Technical Report DAEPACK Web Page, Massachusetts Institute of Technology, Cambridge, MA, 2000.

[18] J. E. Tolsma and P. I. Barton, *DAEPACK: An open modeling environment for legacy models*, Indust. Engrg. Chem. Res., 39 (2000), pp. 1826–1839.

SIAM J. SCI. COMPUT.
Vol. 23, No. 6, pp. 1875–1897

ON THE NUMERICAL SOLUTION OF $(\lambda^2 A + \lambda B + C)\,x = b$ AND APPLICATION TO STRUCTURAL DYNAMICS*

V. SIMONCINI[†] AND F. PEROTTI[‡]

Abstract. In this paper we address the numerical solution of a large linear system depending quadratically on a parameter that varies in a wide range. We analyze a solution method, whose computational cost grows only sublinearly with the number of parameters, that relies on the use of an indefinite inner product. Important implementation aspects are treated in detail. The problem arises in various application areas: we shall report on our experience with cases in structural dynamics.

AMS subject classification. 65F10

Key words. linear systems, iterative methods, inexact methods, complex symmetric matrices, structural dynamics

PII. S1064827501383373

1. Introduction.

We are interested in the numerical solution of the linear system

$$L(\lambda)x = b, \qquad L(\lambda) = \lambda^2 A + \lambda B + C \tag{1.1}$$

as $\lambda \in \mathbb{C}$ varies, where $x = x(\lambda)$. In the following we shall assume that either A or C is nonsingular and that A, B, and C are $n \times n$ complex symmetric matrices, so that $L(\lambda)$ is also complex symmetric. We are interested in the case in which system (1.1) need be solved for several (say hundreds or thousands) values of λ in a wide range.

Equation (1.1) has large application in the solution of difference and ordinary differential equations, where it has been classically studied [14]. Unfortunately, the theoretical spectral tools utilized in such setting are computationally inefficient on large application problems such as electromagnetic scattering [19], wave propagation in porous media [30], and structural dynamics [6]. The quadratic, and in more general contexts nonlinear, parameter λ may be treated using Padé approximation when λ is close enough to a fixed reference value λ_0 [19]. When λ is in a wider range, linearization techniques originating from the eigenvalue setting may be exploited; see, for instance, [25, 16, 39]. In [6] it was shown that for structural dynamics applications, iteratively solving the linearized version of (1.1) is in general more efficient than using a direct sparse solver on (1.1). This is due to the fact that the latter needs to recompute the factors of the sparse factorization for each value of λ, causing the computation cost to grow linearly with the number of parameters. Unfortunately, the iterative methods in [6] did not exploit the symmetry of the problem, so that they suffered from known limitations of the nonsymmetric solvers employed [29]. Although the problem finds application in several important areas, general numerical methods other than those analyzed in [6] do not seem to have been fully investigated (e.g., the methods in [41, 18]

*Received by the editors January 9, 2001; accepted for publication (in revised form) October 15, 2001; published electronically February 20, 2002.
http://www.siam.org/journals/sisc/23-6/38337.html

[†]Dipartimento di Matematica, Università di Bologna, Italy, and Istituto di Analisi Numerica del CNR, Pavia, Italy (val@dragon.ian.pv.cnr.it).

[‡]Dipartimento di Ingegneria Strutturale, Politecnico di Milano, Milano, Italy (perotti@stru.polimi.it).

The research of this author was partially supported by MURST—Italian Ministry for University and Scientific and Technological Research.

are application-oriented), and performance comparisons with obvious techniques seem to be lacking.

In this paper we analyze in detail the iterative solution of (1.1) when a linearization of the problem is carried out, so that system (1.1) is transformed into

$$(\mathcal{A} + \lambda\mathcal{B})z = d$$

of twice the size, with $\mathcal{A}$ and $\mathcal{B}$ complex symmetric and $z = z(\lambda)$. We propose a Krylov subspace method that exploits the symmetry of the problem so as to devise a short-term recurrence whose computational cost per iteration is comparable to that of a Hermitian iterative solver. Collecting $\mathcal{B}$ on the right, we obtain the shifted form

$$(\mathcal{A}\mathcal{B}^{-1} + \lambda I_{2n})\hat{z} = d, \qquad \hat{z} = \mathcal{B}z, \tag{1.2}$$

where I_{2n} is the identity matrix of order $2n$. Due to the shift invariance of Krylov subspaces, the space generated for the approximation of $\hat{z} = \hat{z}(\lambda)$ in (1.2) turns out to be the same for all λ's. This important property allows us to simultaneously handle several λ's at a cost that grows only sublinearly with the number of parameters.

In this paper we give a thorough analysis of the method: relevant aspects are uncovered that make the approach amenable to dealing with very large three-dimensional (3D) problems. To this end, a key step is the solution of a system with the complex symmetric matrix $\mathcal{B}$ at each iteration; the solution is necessary to transform the linearized system into the shifted form (1.2). For large dimension problems, such a solution can be obtained only approximately by means of an iterative method. The inexact solution with $\mathcal{B}$ in general leads to deterioration of performance: a theoretical analysis describes the influence of the inexact procedure on the convergence of the Krylov subspace method and numerical experiments emphasize the difficulties one encounters when using the inexact approach. Most numerical and experimental considerations are carried out on a real application problem, stemming from direct frequency analysis in structural dynamics, in which A is complex symmetric, B is purely imaginary, and C is real. For the sake of generality, however, our numerical experiments allow us to make considerations on the behavior of the method that go beyond the use on our particular application.

We should mention that the idea of combining the symmetry of the problem and the shift invariance of Krylov subspace methods for solving (1.1) first appeared in [33]. This paper aims to investigate the numerous numerical and computational aspects that need to be addressed in order to make a solver of practical value in real application problems.

Following a strategy commonly employed in the nonlinear context, one could solve (1.1) directly for a particular choice of λ and then use the generated information to enhance the solution of (1.1) for the parameter values of interest. Such a strategy would avoid doubling the problem size. Nonetheless, we point to some of the shortcomings of this strategy, while our numerical experiments show that for a large number of parameters, the new method outperforms this type of scheme that directly deals with (1.1).

We next give a synopsis of the paper. In section 2 we give a brief description of the model problem. In section 3 we introduce the linearized matrix formulation and the algebraic tools that are used in the algorithm. In section 4 and its subsection, the Simplified Shifted Lanczos procedure is described. Considerations relating to the solution of the original and linearized problems are discussed in section 5, while acceleration strategies are discussed in section 6. Sections 7 and 7.1 are devoted to the analysis and numerical experiments, respectively, of the exact and inexact solution of the system occurring at each iteration of the Lanczos procedure. In section 8 the method is compared with alternative strategies. Multiple-input systems are discussed in section 9. Finally, our conclusions and comments for future work are summarized in section 10.

All tests were run in Fortran on one CPU of a Sun Enterprise 4500 and plotted using Matlab [21]. Capital letters denote matrices, small Roman letters denote vectors, Greek letters denote scalars, and $\imath$ is the imaginary unit. Matlab notation is used whenever possible. For a complex vector x, $x^T, \bar{x}$ denote the transpose and conjugate of x, respectively, $x^* = \bar{x}^T$, and $(x)_i$ denotes its ith component. For a matrix X, $(X)_{:,i}$ denotes its ith column; unless otherwise stated, its condition number, $\mathrm{cond}(X)$, is the ratio of its largest and smallest nonzero singular values. Moreover, $\|x\|^2 = \sum_{i=1}^{n} |(x)_i|^2$ for $x \in \mathbb{C}^n$, and the induced norm is used for matrices. I_n is the identity matrix of dimension n and e_j is its jth column, whose number of components is usually clear from the context. The pair (λ, x) with $0 \neq x \in \mathbb{C}^n$ is an eigenpair (or latent root and vector) of L if $L(\lambda)x = 0$. Finally, $\mathrm{diag}(\alpha_1, \ldots, \alpha_n)$ is a diagonal matrix of size n with diagonal entries $\alpha_1, \ldots, \alpha_n$.

2. Frequency analysis of linearized dynamical systems.

Let us consider an n-DOF discretized linear system, oscillating with small amplitudes in the neighborhood of a stable equilibrium configuration (see, e.g., [4, 26] for a matrix-oriented treatment). Assuming linear elastic restoring forces and viscous dissipation, the equations of motion can be written, by Lagrange equations, in the following matrix form:

$$M\ddot{q} + D\dot{q} + Kq = p. \tag{2.1}$$

In (2.1) q is the vector of Lagrangian coordinates, p the vector of generalized components of dynamic forces, M and K the kinetic and potential energy matrices, respectively, while D is the damping matrix. In the case of complex harmonic excitation at frequency f, system (2.1) takes the form

$$M\ddot{q} + D\dot{q} + Kq = be^{\imath 2\pi f t}. \tag{2.2}$$

Equation (2.2) admits the steady-state solution $q(t) = \tilde{q}e^{\imath 2\pi f t}$, where $\tilde{q}$ can be obtained as the solution of the linear system

$$\left(-(2\pi f)^2 M + \imath 2\pi f D + K\right)\tilde{q} = b. \tag{2.3}$$

Within the context of steady-state harmonic analysis, a "mixed" damping, encompassing viscous (D_V matrix) and hysteretic (D_H matrix) contributions, can be introduced as $D = D_V + \frac{1}{2\pi f}D_H$. Substitution in the linear system (2.3) leads to

$$\left(-(2\pi f)^2 M + \imath 2\pi f D_V + K_\star\right)\tilde{q} = b, \qquad K_\star = K + \imath D_H. \tag{2.4}$$

We shall assume that the dynamic system under study encompasses p hysteretic subsystems, each of them characterized by a contribution $K^{(j)}$ to the global stiffness matrix and by a hysteretic damping factor $\eta^{(j)}$, so that $D_H = \sum_{j=1}^{p} \eta^{(j)} K^{(j)}$.

The obtained system (2.4) fits our matrix form (1.1) by defining

$$\lambda = 2\pi f, \quad A = -M, \quad B = \imath D_V, \quad C = K_\star, \quad x = \tilde{q}.$$

As an alternative to the above formulation, the linear system can be rewritten in terms of acceleration amplitudes $\tilde{a} = -(2\pi f)^2\tilde{q}$, leading to the "inverse" form

$$\left(M + \frac{1}{\imath 2\pi f}D_V + \frac{1}{-(\imath 2\pi f)^2}K_\star\right)\tilde{a} = b.$$

Table 2.1

Relevant data for our test problems. Matrices B, C are diagonal. Frequencies are given as $\lambda^{-1} \in 2\pi[\alpha, \beta]$. $\#(\cdot)$ = number of entries, considering symmetry.

Pb.	Pb.	$\#(A)$	cond(A)	$\#(B)$	$\lVert B\rVert$	$\#(C)$	$\lVert C\rVert$	freq. (Hz)
	size		(†)					$[\alpha, \beta]$
B	3 627	102 378	$9.7 \cdot 10^4$	211	381	3 627	0.3	$[0.1, 60.1]$
C	2 472	24 340	$3.6 \cdot 10^7$	36	20243	1475	48	$[0.1, 60.1]$
F	11 957	419 160	$2.6 \cdot 10^{12}$	3 243	212	11 907	0.4	$[10, 50]$
F1	11 907	416 855	$2.0 \cdot 10^7$	3 243	212	11 857	0.4	$[10, 50]$

† Matlab `condest` estimate of the 1-norm condition number of the matrix.

Such a formulation can be made consistent with (1.1) by defining

$$\lambda = (2\pi f)^{-1}, \quad C = -M, \quad B = \imath D_V, \quad A = K_\star, \quad x = -\tilde{a}. \tag{2.5}$$

Throughout the paper we shall adopt this formulation whenever addressing the structural dynamics application.

When nonharmonic independent forces are applied to the system, the right-hand side of equations (2.1) can be written as $p(t) = Fg(t)$, with F an $n \times m$ matrix and $g(t) = [\gamma_1(t), \ldots, \gamma_m(t)]$. The Fourier transform of $p(t)$ yields the frequency domain load description $\hat{p}(f) = F\hat{g}(f)$. In the first part of this paper the "single-input" case will be considered ($m = 1$); in this situation a single right-hand side vector $b = F$ is present in (1.1). The solution, however, must be repeated for all frequencies of interest in the spectrum of $g(t)$; a similar situation arises in the case of perfectly correlated stationary random loads. The case $m \neq 1$ (independent deterministic loads or multicorrelated random loads) will be addressed in section 9.

All numerical experiments described in the paper were done using a subset of the test cases first considered in [6]; in there, a description of the originating problem is also presented. Case **F1** is a variant of case **F** , as described below. Table 2.1 reports a summary of the relevant features for our analysis: for each test case, the problem size, the number of nonzero elements of A, B, and C (considering symmetry), an estimate for the condition number of A, and the frequency interval of interest is reported. Note that problem cases **B** and **C** have nonuniform hysteretic damping. In cases **F** and **F1** it is $K_\star = (1 + \imath\eta)K$, with $0 < \eta < 1$, $\eta \in \mathbb{R}$. The ill-conditioning of matrix A in case **F** is due to the unfavorable stiffness properties of the mechanical system, which was originally composed of an elastic body restrained by viscous dampers only, this causing the stiffness matrix to be singular. To face this problem, highly deformable springs were placed parallel to the dampers, leading, in case **F** , to a very high condition number; in case **F1** stiffer springs were inserted which, along with other slight modifications to the model, provided a significant improvement of the matrix conditioning.

The case of uniform hysteretic damping is particularly attractive, since spectral information on the viscous problem can be obtained from the purely hysteretic problem. Let us thus assume that $K_\star \equiv (1 + \imath\eta)K$, with K real symmetric and positive definite and $\eta > 0$. The natural frequencies of the problem lie on a line of the complex plane. Indeed, let $(\xi^2(1 + \imath\eta)K - M)x = 0$ and set $\theta = \xi^2(1 + \imath\eta)$. The eigenvalues θ of the pencil (M, K) are real; therefore

$$\xi^2 = \frac{\theta}{|1 + \imath\eta|^2}(1 - \imath\eta) \tag{2.6}$$

and the eigenvalues ξ are on the line through the origin containing $\sqrt{1 - \imath\eta}$.

The viscous problem can be seen as a perturbation of the uniformly hysteretic case. We next provide an estimate of the magnitude of the perturbed eigenvalues, which is based on Geršgorin theorem and follows the analysis in [20, section 9.3]. A general perturbation analysis can be done using classical theory; see [38] for a study with pseudospectra. In the following, we shall use the notation introduced in (2.5).

PROPOSITION 2.1. *Let (ξ, x) solve the problem $\left((1+\imath\eta)K\xi^2 + \imath D_V\xi - M\right)x = 0$, with K, D_V and M real symmetric, K positive definite and $\|D_V\| \le \varepsilon$. Also let $\sigma_1 \ldots, \sigma_n$ be the eigenvalues of the problem $Mq = \sigma^2(1+\imath\eta)Kq$. Then the eigenvalues ξ are in the union of the disks*

$$\left|\xi - \left(\pm\sigma_k - \frac{\nu}{2}(\hat{D}_V)_{kk}\right)\right| = \left|\frac{\nu}{2}(\hat{D}_V)_{kk}\right| + \sum_{j\neq k}|\nu(\hat{D}_V)_{kj}|, \qquad k = 1, \ldots, n, \tag{2.7}$$

where $\|\hat{D}_V\| \le \varepsilon$ and $\nu = \imath(1+\imath\eta)^{-1}$.

Proof. Let Q be the real orthogonal eigenvector matrix in the eigenvalue problem $Mq = \sigma^2(1+\imath\eta)Kq$ such that $Q^TKQ = I$ and $Q^TMQ = (1+\imath\eta)\Sigma^2$ with $\Sigma =$diag$(\sigma_1, \ldots, \sigma_n)$. Let $x = Qv$. Then

$$\left(\xi^2 I + \xi\nu\hat{D}_V - \Sigma^2\right)v = 0,$$

where $\hat{D}_V = Q^TD_VQ$. The eigenvalues of this problem are the same as the eigenvalues of the matrices

$$\begin{pmatrix} -\nu\hat{D}_V & \Sigma \\ \Sigma & \end{pmatrix}, \qquad \begin{pmatrix} \Sigma - \frac{\nu}{2}\hat{D}_V & \frac{\nu}{2}\hat{D}_V \\ \frac{\nu}{2}\hat{D}_V & -\Sigma - \frac{\nu}{2}\hat{D}_V \end{pmatrix}, \tag{2.8}$$

where the second matrix is obtained by similarity [20, section 9.3].

Applying the Geršgorin theorem to the kth row (or column) of the second matrix in (2.8) yields

$$\left|\xi - \left(\pm\sigma - \frac{\nu}{2}(\hat{D}_V)_{kk}\right)\right| \le \left|\frac{\nu}{2}(\hat{D}_V)_{kk}\right| + \sum_{j\neq k}|\nu(\hat{D}_V)_{kj}|. \qquad \square$$

We note that since $0 \le \eta < 1$ and $\nu = (1+\eta^2)^{-1}(\eta + \imath)$, then the major change in the spectrum is observed in the imaginary part. Not unexpectedly, our analysis predicts that for the class of problems considered, the eigenvalues of the viscous problem tightly distribute about the eigenvalues of the purely hysteretic problem. A typical spectrum is reproduced in Figure 2.1. When we consider the coefficient matrix in the shifted form (1.2) we can readily observe that its eigenvalues are obtained by shifting the eigenvalues ξ's of the original viscous problem.

3. Matrix formulation.

Different matrix formulations can be derived when linearizing (1.1) in λ. The companion algebraic form yields (see, e.g., [14])

$$\left(\begin{bmatrix} 0 & I_n \\ -C & -B \end{bmatrix} - \lambda\begin{bmatrix} I_n & 0 \\ 0 & A \end{bmatrix}\right)\begin{bmatrix} x \\ \lambda x \end{bmatrix} = \begin{bmatrix} 0 \\ -b \end{bmatrix}.$$

Most theoretical results are derived for the formulation above. A symmetric equivalent formulation can be obtained by simply multiplying the first block row by C^T and accommodating signs and rows, so as to obtain

$$\left(\begin{bmatrix} B & C \\ C^T & 0 \end{bmatrix} + \lambda\begin{bmatrix} A & 0 \\ 0 & -C^T \end{bmatrix}\right)\begin{bmatrix} \lambda x \\ x \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}. \tag{3.1}$$

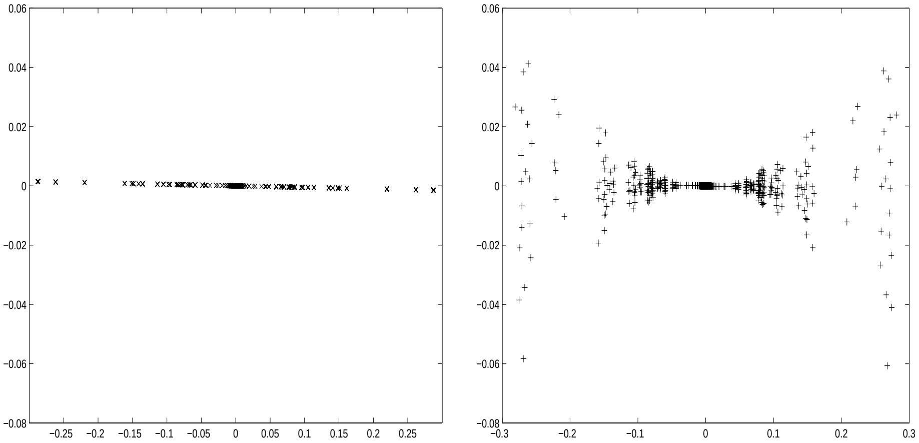


FIG. 2.1. *Typical eigenvalue distribution for the purely hysteretic problem (left) and for the viscous problem (right).*

Such symmetric structure has often been used in the context of eigenvalue computation; see [23, 25] and the recent survey [39]. With obvious notation, system (3.1) can be written in a more compact form as

$$(\mathcal{A} + \lambda \mathcal{B})z = d. \tag{3.2}$$

When A, B, and C are complex symmetric, then $\mathcal{A}$ and $\mathcal{B}$ are also complex symmetric and so is $(\mathcal{A} + \lambda\mathcal{B})$. Throughout the paper we shall assume that $\mathcal{B}$ is nonsingular. Singularity of $\mathcal{B}$ arises if C is singular. This does happen in structural dynamics applications, assuming $C = -M$, where Lagrangian coordinates are often present in the model (e.g., rotation components in framed or plate structures), which are useful for modeling elastic properties but do not appear in the kinetic energy definition. The problem of singular C can be solved in a computationally efficient manner when, for instance, C is diagonal [33]. If instead A is singular and $\lambda \neq 0$, the formulation with coefficient matrix $\sigma^2 C + \sigma B + A$ should be used, with $\sigma = \lambda^{-1}$, and the considerations above still apply.

In the complex symmetric setting, the *indefinite* inner product

$$(x, y) = x^T y \qquad \text{for } x, y \in \mathbb{C}^n \tag{3.3}$$

is used so that $(x, Ax) = (Ax, x)$ for a complex symmetric matrix A: variants of short-term recurrence methods such as conjugate gradients can thus be implemented that employ (3.3) rather than the usual definite inner product in $\mathbb{C}^n$ [40]. It may be that $(x, x) = 0$ for some $x \neq 0$ (called isotropic [13]), so that breakdown can occur; an iterative method employing this inner product should therefore have safeguard strategies, although breakdown is rarely encountered in practice. Because of our application, the derivation and matrix analysis will be done for complex symmetric matrices and the inner product above. Nevertheless, most statements hold for Hermitian matrices by replacing T with the usual transposition and conjugation, and also hold for real symmetric matrices.

Collecting $\mathcal{B}^{-1}$ from the right yields the shifted linear system

$$(\mathcal{A}\mathcal{B}^{-1} + \lambda I_{2n})\hat{z} = d, \qquad \hat{z} = \mathcal{B}z. \tag{3.4}$$

Isolating the frequency parameter in (3.4) is very convenient: if Krylov subspace iterative methods are employed for solving the system, computational costs may be substantially lowered. We remark here that if $\mathcal{B}$ were to have a positive definite symmetric part, then the factorization $\mathcal{B} = LL^T$ ([17]) would lead to a complex symmetric linearized problem with shifted coefficient matrix $L^{-1}\mathcal{A}L^{-T} + \lambda I_{2n}$. To make the treatment sufficiently general, in the rest of the paper we shall not restrict to the case where $\mathcal{B}$ has a positive definite symmetric part. Symmetry properties are lost with matrix $\mathcal{AB}^{-1}$, and nonsymmetric solvers should be applied. Nevertheless, $\mathcal{AB}^{-1}$ is symmetric with respect to a (nonstandard) indefinite inner product. Indefinite inner products have been used in the past for theoretical purposes (see [13] for a general treatment) in connection with the generalized eigenvalue problem [25, 23] as well as in the linear system setting (cf., e.g., [12]).

Definition 3.1. *Given a complex Hermitian (symmetric) matrix $\mathcal{X}$, an indefinite inner product with respect to $\mathcal{X}$ is a bilinear form defined by $[x,y]_{\mathcal{X}} = x^*\mathcal{X}y$ ($[x,y]_{\mathcal{X}} = x^T\mathcal{X}y$) for complex vectors x, y.*

Definition 3.2. *Matrix A is $\mathcal{X}$-Hermitian (-symmetric) if $[x,Ay]_{\mathcal{X}} = [Ax,y]_{\mathcal{X}}$.*

Matrix $\mathcal{B}^{-1}\mathcal{A}$ is $\mathcal{B}$-symmetric and $\mathcal{A}$-symmetric, while $\mathcal{AB}^{-1}$ is $\mathcal{A}^{-1}$- and $\mathcal{B}^{-1}$-symmetric [25]. We shall see in section 4 that Lanczos-type approaches can be implemented so as to properly exploit the indefinite inner product $[\cdot,\cdot]_{\mathcal{X}}$, resulting in a computationally advantageous short-term recurrence method. Since the matrices in our application are complex symmetric, we shall focus our discussion on the (conjugation-free) indefinite inner product $[x,y]_{\mathcal{X}} = x^T\mathcal{X}y$.

Here and in the following we opt for inverting matrix $\mathcal{B}$ because of its computationally convenient block diagonal structure. Unless $\mathcal{A}$ is singular, the following alternative linearized formulation can be used:

$$(I + \lambda\mathcal{B}\mathcal{A}^{-1})\tilde{z} = d. \tag{3.5}$$

Most algorithmic considerations can easily be adapted to the case in which the formulation (3.5) is used. The preference between the two versions depends both on the spectral properties of the problem (cf. [6]) and on the computational effort involved in solving with $\mathcal{B}$ or $\mathcal{A}$.

4. Iterative solution.

In this section we describe the method we use to solve the system (1.1) through the linearization (3.2). We briefly recall the general properties of the Lanczos recurrence, and then we specialize it to symmetric (with respect to the indefinite inner product of Definition 3.2) matrices and to the linearized system.

Given a matrix $G \in \mathbb{C}^{n\times n}$ and the linear system $Gz = d$, the coupled two-term recurrence Lanczos method generates two pairs of vectors $\{q_j\}, \{p_j\}$ and $\{\tilde{q}_j\}, \{\tilde{p}_j\}$ that are pairwise biorthogonal, that is, $\tilde{q}_i^T q_j = 0$, $i \neq j$, and pairwise G-biorthogonal, $\tilde{p}_i^T G p_j = 0$, $i \neq j$. Setting $Q = [q_1, \dots, q_k]$, $P_k = [p_1, \dots, p_k]$, and $\tilde{Q}_k, \tilde{P}_k$ accordingly, we have

$$GP_k = Q_{k+1}L_k, \qquad G^T\tilde{P}_k = \tilde{Q}_{k+1}D_{k+1}^{-1}L_kD_k, \tag{4.1}$$

where $L_k \in \mathbb{C}^{(k+1)\times k}$ and $D_k \in \mathbb{C}^{k\times k}$ are lower bidiagonal and diagonal matrices, respectively. The vectors q_j, p_j and $\tilde{q}_j, \tilde{p}_j$ can be determined via coupled two-term recurrences by explicitly writing the relations above. Moreover, $Q_k = P_kU_k$ with $U_k \in \mathbb{C}^{k\times k}$ upper bidiagonal, which gives the usual three-term Lanczos recurrence $GQ_k = Q_{k+1}L_kU_k$ [11]. The first basis vector, q_1, is the normalized starting residual, $r_0 = d - Gz_0$, where z_0 is the starting approximate solution. The auxiliary starting vector $\tilde{q}_1$ is arbitrary. If a QMR procedure is applied [11], then an approximate solution is obtained as $z_k = z_0 + Q_k y_k$, where y_k solves the problem

$$\min_{y \in \mathbb{C}^k} \| \, \|d\| e_{k+1} - (L_k U_k) y \|. \tag{4.2}$$

The minimization process can be done "on the fly" and the matrix $L_k U_k$ need not be explicitly assembled [11].

In our case, the system to be solved is $(\mathcal{A}\mathcal{B}^{-1} + \lambda I_{2n})\hat{z} = d$. Let us for the moment omit the shift λI_{2n} and concentrate on the matrix $\mathcal{A}\mathcal{B}^{-1}$, so that $G = \mathcal{A}\mathcal{B}^{-1}$ is $\mathcal{B}^{-1}$-symmetric (cf. Definition 3.2). If $\tilde{q}_1$ is chosen to be $\tilde{q}_1 = \gamma_1 \mathcal{B}^{-1} q_1$, for some $\gamma_1 \in \mathbb{C}$, then all subsequent vectors $\tilde{q}_j$ satisfy the relation $\tilde{q}_j = \gamma_j \mathcal{B}^{-1} q_j$, $j = 2, \ldots, k$, for some $\gamma_j \in \mathbb{C}$ [12]. For this choice of auxiliary vector $\tilde{q}_1$, the left Krylov subspace is simply $\mathcal{B}^{-1}$ times the right Krylov subspace; therefore, the recurrence for $\tilde{q}_j$ may be obtained by solving a system with $\mathcal{B}$.

It is also worth noticing that the vector $\mathcal{B}^{-1} q_j$ can also be used to generate the next right iterate q_{j+1}; therefore a multiplication by $G^T \equiv \mathcal{B}^{-1}\mathcal{A}$ per iteration is avoided. The resulting algorithm is a $\mathcal{B}^{-1}$-symmetric Lanczos procedure, where only a set of basis vectors need be recursively computed while the second set of vectors can be recovered at no additional cost [25, 12].

To deal with the shift, we recall that a Krylov subspace is invariant under shift; that is, given $G \in \mathbb{C}^{n \times n}$ and $d \in \mathbb{C}^n$, we have $K_k(G + \lambda I_{2n}, d) = K_k(G, d) \equiv \text{span}\{d, Gd, \ldots, G^{k-1}d\}$, so that $(G + \lambda I_{2n}) Q_k = Q_{k+1}(L_k U_k + \lambda \tilde{I}_k)$, with $\tilde{I}_k = [I_k; 0]$. Note that the generating vector, d, must be the same. For this reason, the right-hand side should be the same for all linear systems; therefore a starting zero approximation $z_0 = 0$ is considered.

Shift invariance allows us to approximate the solution of all systems by generating only the Krylov subspace associated with one of them. Once the space $K_k(\mathcal{A}\mathcal{B}^{-1}, d)$ is constructed, for each λ_j $j = 1, \ldots, s$, an approximate solution $z_k(\lambda_j) = Q_k y_k(\lambda_j)$ may be obtained using a QMR procedure by solving

$$\min_{y \in \mathbb{C}^k} \| \, \|d\| e_{k+1} - (L_k U_k + \lambda_j \tilde{I}_k) y \|.$$

The minimization problem is carried out for each parameter, while the expensive step of generating the Krylov subspace is done once for all. The approximate solution to the original system is determined as $z_k = \mathcal{B}^{-1}\hat{z}_k$. From $\hat{z}_k \in K_k(\mathcal{A}\mathcal{B}^{-1}, d)$ it follows $z_k \in \mathcal{B}^{-1} K_k(\mathcal{A}\mathcal{B}^{-1}, d)$; therefore z_k is a linear combination of the vectors $\tilde{p}_i$, $i = 1, \ldots, k$, and so it may be updated directly as $z_k = \tilde{P}_k g_k$. This consideration will be useful in section 7, where an inexact solution with $\mathcal{B}$ is discussed.

4.1. The algorithm.

The algorithm for solving the linearized system (3.2), which includes symmetry with respect to the indefinite inner product and the shifting procedure, is given below. Key recursions for the simplified and shifted approaches are marked by boldface. Here we report the algorithm with the QMR procedure to generate the approximate solution. Other techniques could also be employed.

ALGORITHM: **Simplified Shifted Lanczos method** with QMR procedure.

```
Given $\mathcal{A}, b, z_0, \mathcal{B}, \lambda_1, \ldots, \lambda_s$
  $\mathcal{I} = \{1, 2, \ldots, s\}$
  $q_0 = b/\|b\|$     $p_0 = q_0$,     $\tilde{p}_0 = \mathcal{B}^{-1} p_0$,     $\tilde{q}_0 = \tilde{p}_0/\|\tilde{p}_0\|$
  $\gamma_0 = \|q_0\|/\|\tilde{q}_0\|$     $\omega_0 = \tilde{q}_0^T q_0$
```

```
$t_0 = \mathcal{A}\tilde{p}_0, \qquad \tau_0 = \tilde{p}_0^T t_0, \qquad v_0^{(j)} = v_{-1}^{(j)} = 0 \quad j \in \mathcal{I}$
$c_0^{(j)} = c_{-1}^{(j)} = 1 \qquad s_0^{(j)} = s_{-1}^{(j)} = 0, \quad \rho^{(j)} = \|b\|$
for $i = 0, 1, \ldots$
    $\alpha_i = \frac{\omega_i}{\tau_i}$
    $q_{i+1} = q_i - \alpha_i t_i$
    $\tilde{\mathbf{u}} = \mathcal{B}^{-\mathbf{1}}\mathbf{q}_{i+1}, \qquad \mathbf{u} = \boldsymbol{\gamma}_i\tilde{\mathbf{u}} \qquad \tilde{\mathbf{q}}_{\mathbf{i+1}}{=}\frac{\mathbf{1}}{\|\mathbf{u}\|}\mathbf{u}$        $(*)$
    $\omega_{i+1} = \tilde{q}_{i+1}^T q_{i+1}$
    $\boldsymbol{\gamma}_{i+1} = \frac{\boldsymbol{\gamma}_i}{\|\mathbf{u}\|} \quad \chi_{i+1} = \|\mathbf{u}\|\,\|p_i\|\,\omega_{i+1}\omega_i^{-1}$
    $p_{i+1} = q_{i+1} + \chi_{i+1}p_i \quad \tilde{\mathbf{p}}_{i+1} = \tilde{\mathbf{u}} + \chi_{i+1}\tilde{\mathbf{p}}_i$
    for each $j \in \mathcal{I}$
        $\nu = \|p_i\| + \chi_{i+1} + \boldsymbol{\lambda_j\alpha_i}, \quad \mu = \overline{s_i}^{(j)}c_{i-1}^{(j)}\chi_i + c_i^{(j)}\nu$
        $\theta = -\|p_{i+1}\|\mu^{-1}, \quad c_{i+1}^{(j)} = (1 + |\theta|^2)^{-\frac{1}{2}}, \quad s_{i+1}^{(j)} = c_{i+1}^{(j)}\overline{\theta}$
        $v_{i+1}^{(j)} = [\tilde{p}_i\alpha_i - v_i^{(j)}(-c_i^{(j)}c_{i-1}^{(j)}\chi_i + s_i^{(j)}\nu) + v_{i-1}^{(j)}s_{i-1}^{(j)}\chi_i]c_{i+1}^{(j)}\mu^{-1}$        $(a)$
        $z_{i+1}^{(j)} = z_i^{(j)} + v_{i+1}^{(j)}\rho^{(j)}c_{i+1}^{(j)}$        $(b)$
        $\rho^{(j)} = -c_{i+1}^{(j)}\theta\rho^{(j)}$
    endfor
    Eliminate converged indexes from $\mathcal{I}$. If $\mathcal{I} = \emptyset$, then stop.
    $t_{i+1} = \mathcal{A}\tilde{p}_{i+1}/\|p_{i+1}\|$
    $\tau_{i+1} = \gamma_{i+1}\tilde{p}_{i+1}^T t_{i+1}$
    $\tilde{p}_{i+1} = \frac{\tilde{p}_{i+1}}{\|p_{i+1}\|}$
endfor
```

The algorithm (in short, SS-Lanczos) requires six vectors of $2n$ components and two matrices of size $2n \times s$. We shall see in section 5 that only some of the rows of these matrices need be stored in a practical implementation. For large problems, the major computational effort is the solution with $\mathcal{B}$ at each iteration, while other operations in the algorithm, such as `daxpy`'s and `dots`' with vectors of $2n$ components, represent a low percentage of the total computational cost.

Convergence could be monitored by explicitly computing the residual norm. However, an upper bound is readily available as (see also [11])

$$\|d - (\mathcal{A} + \lambda_j\mathcal{B})z_k\| \leq \|Q_{k+1}\| \min_{y \in \mathbb{C}^k} \|\,\|d\|e_1 - (L_kU_k + \lambda_j\tilde{I})y\|. \tag{4.3}$$

The norm of Q_{k+1} can be bounded by the square root of the sum of its column's norm. Moreover, in the algorithm above, $\|\,\|d\|e_1 - (L_kU_k + \lambda_j\tilde{I})y_k\| = |\rho^{(j)}|$. We explicitly remark that in all our experiments the relative residual norm was always monitored (also in the inner solver; cf. section 7). The Lanczos procedure may break down if, for instance, $\omega_i \approx 0$ for some i. A look-ahead procedure has been devised to overcome breakdown, and near breakdown, and can be implemented in this case as well [11].

Since the parameters λ_j's are selected in a wide range, convergence curves can be quite different depending on the value of the parameter. In Figure 4.1 we report typical convergence histories for test case **B** when 23 parameter values are considered, with the values of λ uniformly distributed in the relevant interval. We recall that the eigenvalues of the coefficient matrix $\mathcal{A}\mathcal{B}^{-1} + \lambda I_{2n}$ correspond to the eigenvalues of the original viscous problem, shifted by the parameter λ. The least translation of the original spectrum away from zero (cf. Figure 2.1) appears when λ^{-1} is close to the right end of the interval. As observed in our experiments, convergence is in general much slower for those values of the parameter.

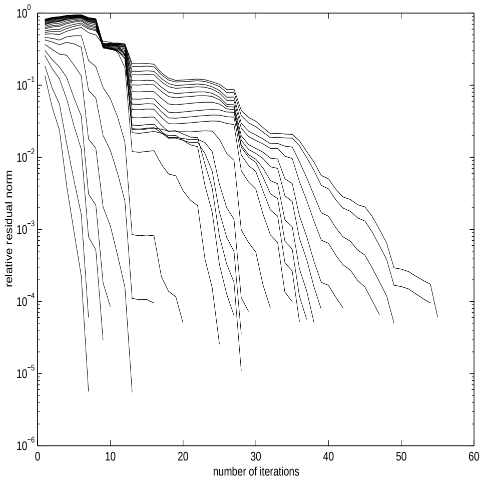


FIG. 4.1. *Case* **B** *. Convergence history for several different parameters. Convergence curves from left (fast) to right (slow) are relative to increasing values of* λ^{-1} *in* $2\pi[\alpha, \beta]$.

When no viscous damping is present in the model, then SS-Lanczos can be applied to the original problem, $(\lambda^2 A + C)x = b$, which is already in linearized form, as a function of λ^2. It should be mentioned that if the Lanczos process were used on the double size problem with $B = 0$, then breakdown would occur at the very first iteration: from $p_0 = [p_0^{(1)}; 0]$ and $\tilde{p}_0 = [\tilde{p}_0^{(1)}; 0]$ it follows that

$$\tau_0 = \tilde{p}_0^T \mathcal{A}\mathcal{B}^{-1} p_0 = [(\tilde{p}_0^{(1)})^T, 0^T] \begin{bmatrix} 0 & C \\ C & 0 \end{bmatrix} \begin{bmatrix} A^{-1}p_0^{(1)} \\ 0 \end{bmatrix} = 0.$$

A similar but more harmful problem is encountered when wishing to solve (3.5), that is, $(I + \lambda\mathcal{B}\mathcal{A}^{-1})\tilde{z} = d$. Indeed, in this case it is easy to show that

$$\tilde{p}^T \mathcal{A}^{-1} p = [\tilde{v}^T, 0^T] \begin{bmatrix} B & C \\ C & 0 \end{bmatrix}^{-1} \begin{bmatrix} v \\ 0 \end{bmatrix} = 0 \qquad \text{for any } v, \tilde{v}.$$

The algorithm breaks down at the first iteration, and it may break down in subsequent iterations in case such an operation is encountered. In this case, a look-ahead procedure should be implemented in order to step ahead in the recurrence [9]; cf. section 6 for alternative strategies.

5. Relations to the original problem. In the described procedure, (1.1) is linearized, and an approximate solution to (3.2) is determined as $z_k = [y_k; x_k]$. At termination, x_k provides an approximate solution to the original problem. However, it would be desirable to directly monitor the goodness of the approximate solution during the Lanczos recurrence. The residual $r_k = d - (\mathcal{A} + \lambda\mathcal{B})z_k$ does not need to be calculated, but it can be approximated via (4.3). On the other hand, $\|r_k\|$ and the norm of the residual $b - (\lambda^2 A + \lambda B + C)x_k$ may be very different [33], yielding an uncertain evaluation of the final sought approximation. Let us instead consider $\lambda^{-1}y_k$ as approximate solution. In theory, for large enough k, $x_k = \lambda^{-1}y_k$; equality, however, does not hold for smaller values of k. Let $R_k = b - (\lambda^2 A + \lambda B + C)\lambda^{-1}y_k$.

Then [33]

$$R_k = \left[I_n, \frac{1}{\lambda} I_n\right] r_k.$$

Clearly, $\|R_k\|$ may be easily estimated through the bound for $\|r_k\|$ during the Lanczos recurrence, whereas the residual $b - (\lambda^2 A + \lambda B + C)x_k$ would have to be computed explicitly. The choice between x_k and $\lambda^{-1} y_k$ may be done a posteriori at termination.

A further consideration from an application viewpoint is that in several situations the quantity of interest is not the solution x but only some of its components. This often happens in structural dynamics applications; in finite-element soil-structure interaction analyses, for example, response parameters within the ground mesh, where most of the elements are located, are usually of little interest. In some extreme cases, such as in the evaluation of rigid foundations impedance functions, only very few (six for a rigid foundation) components of the displacement vector are of interest. When only some of the solution components are requested, major computational and memory savings may be obtained by referencing only such entries. Let $N_\ell = [e_{k_1}, \ldots, e_{k_\ell}]$, where $k_1, \ldots, k_\ell$ are the requested component indexes. Then

$$X^{(j)} \equiv N_\ell^T (\lambda_j^2 A + \lambda_j B + C)^{-1} b.$$

In general, $\ell \ll n$. In the special case where $\ell = \mathcal{O}(1)$, the tall matrix N_ℓ can be used to construct an ad hoc Lanczos process [5].

We can thus use the approximation $X^{(j)} \approx X_k^{(j)} = \lambda_j^{-1} N_\ell^T y_k^{(j)}$. Setting $\hat{N}_\ell = [N_\ell; 0]$ and $f_i^{(j)} = \hat{N}_\ell^T v_i^{(j)}$, lines (a)–(b) in the algorithm can be modified as

$$f_{i+1}^{(j)} = [\hat{N}_\ell^T \tilde{p}_i \alpha_i - f_i^{(j)}(-c_i^{(j)} c_{i-1}^{(j)} \chi_i + s_i^{(j)} \nu) + f_{i-1}^{(j)} s_{i-1}^{(j)} \chi_i] c_{i+1}^{(j)} \mu^{-1} \qquad (a')$$

$$X_{i+1}^{(j)} = X_i^{(j)} + \lambda_j^{-1} f_{i+1}^{(j)} \rho^{(j)} c_{i+1}^{(j)}. \qquad (b')$$

The s vectors $f_i^{(j)}$'s, $j = 1, \ldots, s$, have only ℓ components, with $\ell \ll 2n$, while computing $\hat{N}_\ell^T \tilde{p}_i$ does not entail any floating point operation. To make the presentation and the computational experiments more general, in our implementation we have updated the entire upper portion of the approximate solution, that is, vector $y_k^{(j)}$, at each iteration, together with the first n entries of vectors $v_i^{(j)}, v_{i-1}^{(j)}$, $j = 1, \ldots, s$.

6. Acceleration strategies. Convergence may be slow if the coefficient matrix of the linearized problem has unfavorable spectral properties. The major problem encountered when seeking an efficient preconditioner for $(\mathcal{A}\mathcal{B}^{-1} + \lambda I_{2n})\hat{z} = d$ is that the matrix $\mathcal{A}\mathcal{B}^{-1}$ cannot be formed explicitly, so that usual incomplete factorizations are in general inapplicable. Moreover, preconditioning a system while maintaining the shifted structure is by itself a challenging problem. Attempts have been made in the past by using polynomial preconditioning, which preserves the shifted form [7]. Polynomial preconditioning consists of applying to a recurrence iterate a linear combination of powers of the coefficient matrix at each iteration. Unfortunately, in our case any additional multiplication by the system matrix entails a solution with $\mathcal{B}$, which makes the whole polynomial preconditioning procedure very expensive.

The location of the parameter λ in the spectral region of the coefficient matrix may highly influence the convergence of the method. An alternative acceleration strategy consists of first fixing a reference value λ_0 and then solving the problem with respect to the parameter $\omega = \lambda - \lambda_0$, varying ω. This can be done on either the original problem or the linearized equation.

In the original problem (1.1) let $\lambda = \lambda_0 + \omega$. Substituting in $L(\lambda)$ we get

$$\begin{aligned} L(\lambda_0 + \omega) &= \omega^2 A + \omega(2\lambda_0 A + B) + (\lambda_0^2 A + \lambda_0 B + C) \\ &= \omega^2 A + \omega L'(\lambda_0) + L(\lambda_0) \equiv H_{\lambda_0}(\omega). \end{aligned} \tag{6.1}$$

Here L' stands for the derivative of L with respect to λ. Note that the two matrices have the same eigenvectors. Matrix $H_{\lambda_0}(\omega)$ can be linearized as

$$\begin{bmatrix} L'(\lambda_0) & L(\lambda_0) \\ L(\lambda_0) & 0 \end{bmatrix} + \omega \begin{bmatrix} A & 0 \\ 0 & -L(\lambda_0) \end{bmatrix} \equiv \mathcal{A}_{\lambda_0} + \omega \mathcal{B}_{\lambda_0}. \tag{6.2}$$

We next give an explicit relation between the eigenpairs of the two linearizations.

Remark 6.1. Let (θ, z) and (ρ, t), with θ, ρ similarly ordered, be the eigenpairs of $\mathcal{A}_{\lambda_0}(\mathcal{B}_{\lambda_0})^{-1} + \omega I_{2n}$ and $\mathcal{A}\mathcal{B}^{-1} + \lambda I_{2n}$, respectively, with $z = [x; y]$. Assume that λ_0 is not an eigenvalue of $L(\lambda)$. Then

$$\theta = \rho \quad \text{and} \quad t = \begin{bmatrix} I & \\ & CL(\lambda_0)^{-1} \end{bmatrix} z - \lambda_0 \begin{bmatrix} AL(\lambda_0)^{-1} y \\ 0 \end{bmatrix}.$$

Proof. The pair (θ, z) with $z = [x; y]$ is an eigenpair of $\mathcal{A}_{\lambda_0}(\mathcal{B}_{\lambda_0})^{-1} + \omega I_{2n}$ if and only if $((\omega-\theta)^2 A + (\omega-\theta)L'(\lambda_0) + L(\lambda_0))\hat{y} = 0$ with $\hat{y} = L(\lambda_0)^{-1} y$ and $x = -(\omega-\theta)A\hat{y}$. By explicitly writing L' and L, the last quadratic eigenvalue problem can be rewritten as $((\lambda - \theta)^2 A + (\lambda - \theta)B + C)\hat{y} = 0$; that is, (θ, t) with $t = [-(\lambda - \theta)A\hat{y}; C\hat{y}]$ is an eigenpair of $\mathcal{A}\mathcal{B}^{-1} + \lambda I_{2n}$ from which the assertion easily follows. □

Numerical experiments not reported here showed that this strategy did not lead to better performance.

Acting on the linearized version of the matrix $L(\lambda)$ does change the spectrum. Writing once more $\lambda = \lambda_0 + \omega$, then (cf., for instance, [5, 35, 22])

$$\mathcal{A} + (\lambda_0 + \omega)\mathcal{B} = (\mathcal{A} + \lambda_0 \mathcal{B}) + \omega \mathcal{B}.$$

If $(\mathcal{A} + \lambda_0 \mathcal{B})$ is nonsingular, right multiplication with $(\mathcal{A} + \lambda_0 \mathcal{B})^{-1}$ yields a linear system with coefficient matrix

$$I_{2n} + \omega \mathcal{B}(\mathcal{A} + \lambda_0 \mathcal{B})^{-1}. \tag{6.3}$$

If $\mathcal{A}u = -\theta \mathcal{B}u$, then shifting and inverting we obtain

$$\mathcal{B}(\mathcal{A} + \lambda_0 \mathcal{B})^{-1} v = \frac{1}{\lambda_0 - \theta} v \qquad \text{with} \quad v = (\mathcal{A} + \lambda_0 \mathcal{B})u.$$

If λ_0 is close to some of the eigenvalues of the pencil $(\mathcal{A}, -\mathcal{B})$, then the corresponding eigenvalues of $\mathcal{B}(\mathcal{A} + \lambda_0 \mathcal{B})^{-1}$ and of matrix (6.3) are magnified. If $|\omega| = |\lambda - \lambda_0| \ll 1$, all remaining eigenvalues of matrix (6.3) conveniently cluster around one. However, for larger ω, eigenvalues distribute in a less predictable way. This strategy works best for $|\omega|$ small, that is, for λ in a neighborhood of λ_0. For this reason, the spectrum of the matrix in (6.3) may or may not be more appealing than that of $\mathcal{A}\mathcal{B}^{-1} + \lambda I_{2n}$ in our iterative solver context. A priori information for choosing the most appropriate shift is thus required; see also the discussion in [5, section IV.B] and [22]. Regardless of performance, this approach may be very convenient to avoid possible ill-conditioning or even singularity of $\mathcal{A}$ or $\mathcal{B}$ (e.g., as in our case $\mathbf{F}$), since λ_0 may be chosen to make $(\mathcal{A} + \lambda_0\mathcal{B})$ nonsingular.

Unfortunately, the difficulty in the choice of the reference value λ_0 is not the only shortcoming of the acceleration strategies discussed above. Indeed, at each iteration a linear system with matrix $\mathcal{B}_{\lambda_0}$ and $\mathcal{A}+\lambda_0\mathcal{B}$ need be solved, respectively. In particular, $\mathcal{A} + \lambda_0\mathcal{B}$ and $\mathcal{B}_{\lambda_0}$ may not inherit the sparsity patterns of matrices A, B, and C, resulting in a possibly much denser coefficient matrix. Similar concerns apply if an iterative solver is used (cf. section 7). Such a disadvantage need be taken into account since the overall performance of the iterative scheme strongly depends on the cost of these solves.

In all cases, the iterative solver described in section 4 may be adapted. For the sake of simplicity, however, in the following we shall refer only to the solution of $(\mathcal{A}\mathcal{B}^{-1} + \lambda I_{2n})\hat{z} = d$.

7. Exact and inexact solution of the system with matrix $\mathcal{B}$. At each iteration of the simplified Lanczos method a system solution with matrix $\mathcal{B}$ is required. Due to the structure of $\mathcal{B}$, this actually means solving with the symmetric nonsingular matrices A and C. In our application C is diagonal; therefore the complexity of solving with $\mathcal{B}$ is mostly due to that of solving with A. Medium to large 3D application problems yield matrices of type A that may produce quite dense factors if sparse direct solvers are used, so that most computational time is spent solving with A at each iteration. An example is shown in Table 7.1, where for our test cases the following information is reported for one parameter value of λ: elapsed time (Fortran function etime) to compute the factor[1] of A, total elapsed time to solve with the computed factor during the whole Lanczos process, total elapsed time for the iterative solver (including the time for computing and using the factors of $\mathcal{B}$), and memory allocations for the factor (complex and integers). The sparse direct solver ME47 [1] was used to solve with $\mathcal{B}$. The outer tolerance of the iterative solver was set to $\texttt{tol}_{outer} = 10^{-4}$. Though quite loose, this tolerance is considered satisfactory for this kind of application. The factor is computed once and for all, while it is clear that most time is spent in the solve with A at each iteration.

The large amount of memory allocation required by the factors warns that for larger 3D problems direct (exact) methods may not be affordable. Iterative (inexact) methods should be employed to solve with $\mathcal{B}$, leading to the commonly used term inner-outer iteration for the whole Lanczos recurrence. We shall see that the accuracy with which the inner system is solved influences not only the overall performance of the method (and this is typical of inner-outer procedures), but also the level of final accuracy of the outer iterative solver. In order to overcome the problem arising with the direct solver, the iterative solver should require much less memory allocation. On the other hand, memory allocation may be high if iteratively solving with A requires strong preconditioning. Therefore, the overall effectiveness of the inner-outer procedure depends on the effectiveness of solving with the stiffness and hysteretic damping matrix.

The modification occurring in the outer iterative algorithm is minimal and only involves the step that applies $\mathcal{B}^{-1}$. The resulting variant shares the algebraic properties of usual inner-outer procedures (cf., e.g., [15]) and of flexible versions of known algorithms [27, 37], although in our case it is not the preconditioner that is inaccurately applied, but part of the matrix itself. The subspace generated by the Lanczos recur-

[1] For the complex symmetric matrix A, the LDL^T factorization is carried out [1].

Table 7.1

Splitting of computational cost and memory allocations for the test cases in Table 2.1. $\lambda^{-1} = 10 \cdot 2\pi$. *Elapsed time is in seconds.*

Test case	# its	E–time factorization	E–time solves	E–time iterative solver	Mem. alloc. (cmplxM/intM)
B	12	2.1	0.8	3.2	0.52/0.21
C	92	0.5	1.6	2.9	0.15/0.05
F	105	342	158	509	12/0.86
F1	36	315	52	371	11/0.86

sion is no longer a Krylov subspace. The relation $z_k \in \mathcal{B}^{-1}K_k(\mathcal{A}\mathcal{B}^{-1}, d)$ does not hold, in general, whereas the relation $z_k \in \text{span}\{\tilde{p}_1, \dots, \tilde{p}_k\}$ holds, where $\text{span}\{\tilde{p}_1, \dots, \tilde{p}_k\}$ is not a Krylov subspace. Nonetheless, the important matrix equality

$$\mathcal{A}\tilde{P}_k = Q_{k+1}L_k \tag{7.1}$$

still holds, with $\tilde{P}_k \approx \mathcal{B}^{-1}P_k$, where the approximation depends on how accurately the inner system is solved. The approximate solution z_k can still be updated as in the exact case by using the generated basis vectors $\tilde{p}_i$. Iteratively solving with $\mathcal{B}$ with low accuracy implies that relation $(*)$ in the algorithm does not hold even in exact arithmetic. Instead, $\mathcal{B}\tilde{u} = q_k - t_k$, where t_k is the residual of the inner solver. The coupling between the $\tilde{Q}$- and $\tilde{P}$-recurrences becomes

$$\tilde{Q}_k\Sigma_k^{-1} = \tilde{P}_kU_k, \qquad \Sigma_k = \text{diag}(\|\tilde{u}_1\|, \dots, \|\tilde{u}_k\|). \tag{7.2}$$

We next show that the approximate solution of the system with $\mathcal{B}$ affects the final attainable accuracy of the outer procedure.

Proposition 7.1. *Let $T_k = [t_1, \dots, t_k]$ be the matrix of inner residuals, that is, $\mathcal{B}\tilde{u} = q_i - t_i$ for $i = 1, \dots, k$. Then the residual $r_k = d - (\mathcal{A} + \lambda\mathcal{B})z_k$ satisfies the relation $r_k = Q_{k+1}h_k + \lambda T_k y_k$, so that*

$$\|r_k\| \leq \|Q_{k+1}h_k\| + |\lambda|\|T_k y_k\|, \tag{7.3}$$

where y_k, h_k are the solution and residual of the least squares problem (4.2), *respectively.*

Proof. We have

$$\mathcal{B}\tilde{Q}_k\Sigma_k^{-1} = Q_k - T_k. \tag{7.4}$$

Moreover, $z_k = \tilde{P}_k g_k = \tilde{P}_k U_k y_k$. Therefore,

$$\begin{aligned} r_k &= d - (\mathcal{A} + \lambda\mathcal{B})z_k = d - \mathcal{A}\tilde{P}_k g_k - \lambda\mathcal{B}\tilde{P}_k g_k \\ &\overset{(7.1)-(7.2)}{=} d - Q_{k+1}L_k g_k - \lambda\mathcal{B}\tilde{Q}_k\Sigma_k^{-1}U_k^{-1}g_k \\ &\overset{(7.4)}{=} Q_{k+1}(\|d\|e_1 - L_k g_k - \lambda[U_k^{-1}; 0]g_k) + \lambda T_k U_k^{-1} g_k \\ &= Q_{k+1}\left(\|d\|e_1 - (L_kU_k + \lambda\tilde{I}_k)y_k\right) + \lambda T_k y_k, \end{aligned}$$

where $\tilde{I}_k = [I_k; 0]$. The upper bound thus follows from taking norms. □

In exact arithmetic, the norm of h_k in the first right addend in (7.3) converges to zero as $k \to 2n$, so that $\|Q_{k+1}h_k\|$ goes to zero if Q_{k+1} is not too ill-conditioned. The second right addend in (7.3) measures how well the inner system is solved. For

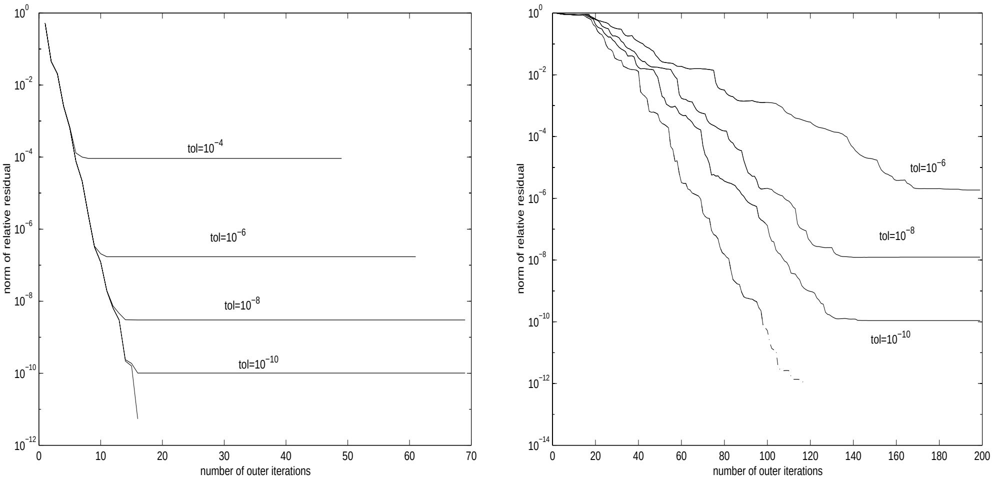


FIG. 7.1. *Inner-outer iteration. Convergence history for different values of the inner stopping tolerance. Left: Example* 1. *Right: Example* 2.

large enough k, $\|r_k\| \approx |\lambda| \, \|T_k y_k\|$, where $\|T_k y_k\|$ in general remains approximatively at the level of the inner tolerance.

Neither Q_{k+1} nor h_k are the same as if the inner system were solved exactly. More precisely, the inner solver not only influences the level of final attainable accuracy, but also the number of iterations to reach that level, delaying convergence in some cases. Inexact methods are known to suffer of this shortcoming (cf. [24, 29] and the references therein); an analysis in the symmetric case has been recently done by Golub and Ye [15], while the real nonsymmetric problem has been studied, for instance, in the context of flexible methods [27, 37]. A more precise analysis directly related to the linearized problem (3.2) would be desirable.

7.1. Numerical experiments with the inner-outer method. We illustrate the behavior of the inner-outer procedure with some numerical tests. We start with two built-up examples.

Example 1. Let $\mathcal{A}$ be the centered finite difference discretization of the two-dimensional Laplacian on the unit square with homogeneous boundary conditions and let $\mathcal{B}$ be a diagonal matrix, $\mathcal{B} = I_{2n} + \imath D$, with D real diagonal with normally distributed diagonal values (Matlab function `randn`). We have $\text{cond}(\mathcal{A}\mathcal{B}^{-1}) \approx 56$ and $\text{cond}(\mathcal{B}) \approx 2.4$. The performance of the inner-outer procedure is shown in Figure 7.1 (left) for various values of the inner stopping tolerance. (Random right-hand side with normally distributed entries and zero starting approximation were considered.) Here and in Example 2, $s = 1$ and $\lambda_1 = 10$. For this example, the inner tolerance provides quite a sharp approximation of the final attainable accuracy, while the convergence curve of the inner-outer solver is indistinguishable from that of the exact scheme, as long as the outer residual is larger than the inner tolerance.

Example 2. Let us consider $\mathcal{B} = (0.001+\imath)D$, with D as in Example 1. In this case $\text{cond}(\mathcal{A}\mathcal{B}^{-1}) \approx 5.9 \cdot 10^5$ and $\text{cond}(\mathcal{B}) \approx 5 \cdot 10^4$. The convergence curves for several inner tolerance values are shown in Figure 7.1 (right). Convergence starts deteriorating much earlier than at the final accuracy level. Similar behavior is observed in realistic problems, as in our test case **B** (cf. Figure 7.2).

Convergence deterioration depends on the conditioning of $\mathcal{A}\mathcal{B}^{-1}$ (cf. [15] in the

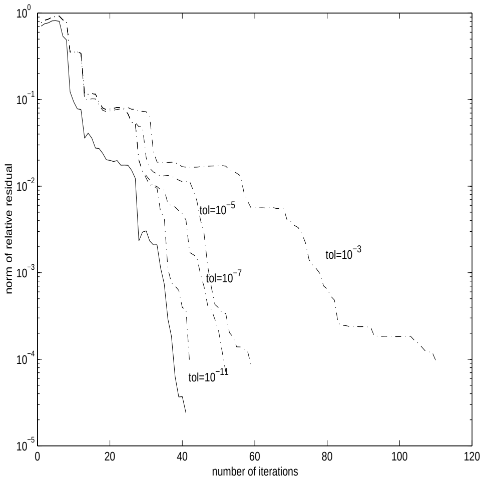


FIG. 7.2. *Test Case* **B** *. Exact (solid) and inexact (dashed) $\mathcal{B}$-symmetry for various stopping inner tolerance values* `tol`*.*

symmetric case); therefore the inner tolerance should be small if the conditioning of $\mathcal{A}\mathcal{B}^{-1}$ is large. In our numerical experiments, ill-conditioning of $K_\star$ seemed to strongly affect the performance of the inner-outer solver, yielding very slow convergence unless a very strict inner tolerance was used (cf. section 8).

To enhance convergence, preconditioning should be employed when solving with $\mathcal{B}$. In our experiments, we have used the preconditioned conjugate gradients method for complex symmetric matrices, which appropriately exploits the indefinite inner product (3.3); see [40]. Preconditioning has been carried out by using a complex symmetric incomplete factorization with fill-in and threshold: a manual complexification of the real symmetric ICT algorithm by Chow and Saad [3] worked satisfactorily on our problem. The preconditioner requires two parameters: the number of nonzero elements per row in the factor (fill-in) in addition to the original entries of the matrix and a threshold value for discarding small entries in the factor [3]; see also [28]. This last parameter was set to the experimentally observed best value, 10^{-4}, in our experiments.

8. Comparison with other techniques.

Solving directly with $L(\lambda)$ would be more convenient than turning to the linearized problem of twice the size if the multiple parameters could be handled efficiently. Efforts in this direction were made in [41, 18], where, however, the approximation was constructed starting from spectral information on the unviscous problem.

To exploit the structure in the solution of (1.1), one could use the factors of the full factorization of $L(\lambda_0)$ as the preconditioner for a value of λ_0 for which $L(\lambda_0)$ is nonsingular. More precisely, for each λ, the preconditioned system of size n

$$L(\lambda)L(\lambda_0)^{-1}\hat{x} = b, \qquad x = L(\lambda_0)^{-1}\hat{x}, \tag{8.1}$$

is solved with an iterative method. Note that $L(\lambda_0)$ may be denser than its addends A, B, and C. In our application problem this is not the case, however, since B and C are always diagonal; this special structure is very favorable for strategies attacking the original problem, as we shall see in our numerical experiments. The goodness of the preconditioner is measured in terms of the magnitude of the error norm $\|L(\lambda)L(\lambda_0)^{-1} - I_n\|$ (cf., e.g., [29]). The following lower bound can be easily proved.

Remark 8.1. The preconditioner matrix $L(\lambda_0)$ satisfies

$$\|L(\lambda)L(\lambda_0)^{-1} - I_n\| \geq \\ |\omega| \max\left\{ \frac{\|(\lambda+\lambda_0)A + B\|}{\|L(\lambda_0)\|}, \sigma_{\min}((\lambda+\lambda_0)A + B) \cdot \|L(\lambda_0)^{-1}\| \right\},$$

where $\sigma_{\min}(\cdot)$ is the smallest singular value of the argument matrix.

Proof. Using (6.1), the preconditioned matrix can be written as

$$L(\lambda)L(\lambda_0)^{-1} = \omega(\omega A + L'(\lambda_0))L(\lambda_0)^{-1} + I_n. \tag{8.2}$$

Therefore,

$$\begin{aligned}\|L(\lambda)L(\lambda_0)^{-1} - I_n\| &\geq \sigma_{\min}(L(\lambda_0)^{-1})|\omega|\|(\lambda+\lambda_0)A + B\| \\ &= \frac{|\omega|}{\|L(\lambda_0)\|}\|(\lambda+\lambda_0)A + B\|.\end{aligned}$$

The other lower bound similarly follows. □

It follows that the condition $|\omega| \ll 1$ (that is, λ close to λ_0) is not sufficient for ensuring a small preconditioning error norm. Indeed, if, e.g., $\|A\|$ is much larger than the norm of B and C, as is usually the case for stiffness matrices in structural dynamics application, and if $|\lambda_0| \ll 1$, then $\|(\lambda+\lambda_0)A + B\| \gg \|L(\lambda_0)\|$ and $L(\lambda_0)$ will be a poor preconditioner for $L(\lambda)$, regardless of the distance between λ_0 and λ. Numerical experiments (see Example 3) seemed to agree with this observation.

When an exact factorization of $L(\lambda_0)$ is not feasible, then an incomplete factorization can be used. Since the cost of generating the incomplete (and sparser) factor is much lower than for the exact factor, the computation can be done for each λ of interest. In practice, this corresponds to iteratively solving $L(\lambda)x = b$ independently for each value of λ with an incomplete factorization of the coefficient matrix as the preconditioner. Obviously, no advantage is taken of the special structure of the problem, and cost grows approximately linearly with the number of parameters. Memory requirements are low since n-vectors are needed, compared with the $2n$-vectors of the linearized approach. If the solution is carried out sequentially, then only the n-vectors of the iterative method are required, together with the preconditioner, while in SS-Lanczos vectors of s components are also required. Nevertheless, the major memory requirements are due to the (incomplete) factorization of $L(\lambda)$, which needs at least as much memory as that for A, depending on the application.

Example 3. We numerically compare the SS-Lanczos method (exact and inner-outer versions) with the described approaches. We experimented with the following:

- PQMR-ME47: complex symmetric QMR [10] on $L(\lambda)x = b$ preconditioned by a complete factorization of $L(\lambda_0)$ with $\lambda_0 = (60\pi)^{-1}$;
- SSL: SS-Lanczos with QMR procedure;
- PQMR-ICT: complex symmetric QMR on $L(\lambda)x = b$ preconditioned by an incomplete factorization of $L(\lambda)$ with no fill-in;
- ISSL: inner-outer SS-Lanczos with QMR procedure.

Note that fill-in in the PQMR-ICT preconditioner provided an overall worse performance of the method than no fill-in.

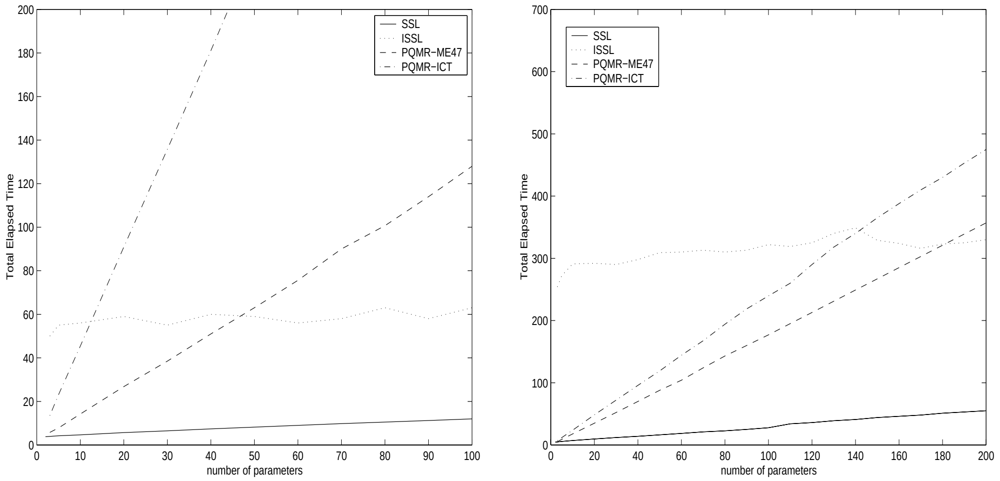


Fig. 8.1. *Elapsed time for the methods as a function of the number of parameters. Case* **B** *(left) and* **C** *(right).*

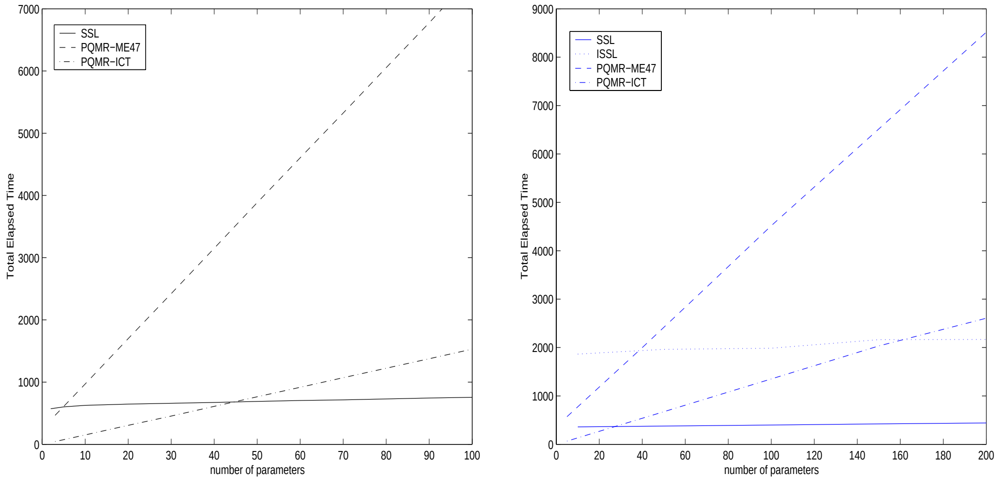


Fig. 8.2. *Elapsed time for the methods as a function of the number of parameters. Cases* **F** *(left) and* **F1** *(right); In case* **F** *the curve for the inner-outer SS-Lanczos method is out (top) of the plot.*

In Figures 8.1 and 8.2 the total elapsed time for cases **B** , **C** , **F** , and **F1** is reported, versus the number of parameters uniformly distributed in the interval. The outer tolerance $\texttt{tol}_{outer} = 10^{-4}$ was used.

All plots consistently show that the exact SS-Lanczos method provides the overall best performance for several frequency values. (In this application problem, a few hundred distinct frequency values would be needed [6].)

Solving with the inner-outer method is much more expensive than with the exact SS-Lanczos procedure. The performance discrepancy highly depends on the difficulty of iteratively solving with $K_\star$ in the inner-outer method. High preconditioning fill-in (fill-in=100 for cases **B** and **C**, fill-in=30 for cases **F** and **F1**) and strict inner tolerances (tol=10^{-6} for case **B** and tol=10^{-8} for the rest) were employed in order to maintain a performance, in terms of iterations and elapsed time, as close as possible to that of the exact method. Nonetheless, in the highly ill-conditioned case **F** the inner-outer method is still too expensive (cf. section 7) compared to all methods: its convergence curve is out of the plot axes. In spite of the high fill-in, memory allocations for the preconditioner were always less than one million, also for cases **F** and **F1** (cf. Table 7.1). As expected, the cost for solving with the exact and inexact SS-Lanczos method grows very little with the number of parameters compared to the other approaches.

In order to make a fair performance evaluation against the techniques that directly attach $L(\lambda)$, SSL should be compared to PQMR-ME47, since they both use a direct solver, while ISSL should be compared to PQMR-ICT. In the latter comparison, we can notice that the sparsity pattern of $L(\lambda)$ is in our application the same as that of $K_\star$. Solving with $L(\lambda)$ is thus not much worse than solving with $K_\star$, so we expect that the inner-outer method will not do better than PQMR-ICT if the number of iterations to solve the linearized problem is larger than the number of frequencies. On the other hand, unless $K_\star$ is severely ill-conditioned as in case **F** , ISSL outperforms PQMR-ICT for a large enough number of parameters. In other applications where the sparsity pattern of B and C is different from that of A, the performance of the method may significantly change. Our experiments thus show that solving the system by sharing the common information of all parameters is in general convenient, even in the inexact case.

We finally notice that PQMR-ME47 remains competitive, though always much worse than SS-Lanczos, as long as dealing with the factor of $K_\star$ is not too expensive, which is the case for the larger cases **F** and **F1** . It is also worth mentioning that in terms of number of iterations, preconditioning with $L(\lambda_0)$ was completely analogous to using ICT on each system with the mentioned parameters; the computational cost and memory requirements, however, were much higher.

9. Multiple-input systems. So far we have considered the case of one right-hand side b in (1.1) (the forcing load in our application). In the multiple-input case the vector of generalized dynamical loads in (2.1) can be written as $p(t) = Fg(t)$, with F an $n \times m$ matrix and $g = [\gamma_1; \cdots ; \gamma_m] \in \mathbb{C}^m$. In the frequency space, this yields the load

$$b(\lambda) = F\hat{g}(\lambda), \qquad F \in \mathbb{C}^{n\times m} \tag{9.1}$$

which thus depends on the frequency parameter. We next consider two alternatives to cope with the presence of a right-hand side in the system (1.1) that also depends on the parameter λ:

(i) Solve the linear system $(\lambda_j^2 A + \lambda_j B + C)X = F$ for $j = 1, \dots, s$ for all right-hand sides represented by the m columns of F, with $X \in \mathbb{C}^{n\times m}$.

(ii) Solve $(\lambda_j^2 A + \lambda_j B + C)x = b(\lambda_j)$ for $j = 1, \dots, s$, with $x = x(\lambda_j)$.

We shall see that the choice between the two approaches depends on the relative size of s (the number of parameters) and m (the number of different loading histories).

Case (i). Constant multiple right-hand sides. The linearized problem in this case can be written as

$$(\mathcal{A} + \lambda_j \mathcal{B})Z = \widehat{D}, \qquad \widehat{D} = [F; 0] \in \mathbb{C}^{2n\times m},\ Z = Z(\lambda_j). \tag{9.2}$$

This system can be solved by using a generalization of the SS-Lanczos algorithm for multiple right-hand sides. Using a block approach, for instance, one constructs the *block* Krylov subspace[2] $\mathcal{K}_{mk}(\mathcal{A}\mathcal{B}^{-1}, \widehat{D})$ and then approximates all shifted systems using the shift invariance of the space (cf., e.g., [34]). The block version of the two-term recurrence Lanczos method generates pairs of matrices $\mathcal{Q}_k, \tilde{\mathcal{Q}}_k$ and $\mathcal{P}_k, \tilde{\mathcal{P}}_k$ that satisfy the relations $\mathcal{A}\mathcal{B}^{-1}\mathcal{P}_k = \mathcal{Q}_{k+1}\mathcal{L}_k$ and $\mathcal{Q}_k = \mathcal{P}_k\mathcal{U}_k$, where $\mathcal{L}_k$,$\mathcal{U}_k$ are block lower and upper bidiagonal. An approximate solution to the linearized problem (9.2) is found as $\hat{Z}_k = \tilde{\mathcal{P}}_k Y_k$, where $Y_k \in \mathbb{C}^{km\times m}$ solves

$$\min_{Y\in\mathbb{C}^{km\times m}} \|E_1\chi - (\mathcal{L}_k\mathcal{U}_k + \lambda_j\tilde{I})Y\|, \qquad j = 1,\dots,s, \tag{9.3}$$

where $\widehat{D} = \mathcal{Q}_{k+1}E_1\chi$, $E_1 = [I_m; 0] \in \mathbb{R}^{(k+1)m\times m}$ and $\tilde{I} = [I_{mk}; 0] \in \mathbb{R}^{(k+1)m\times km}$ [32].

Note that the block tridiagonal matrix $(\mathcal{L}_k\mathcal{U}_k + \lambda_j\tilde{I})$ is of size mk. The least squares solution matrix Y_k is updated at each iteration k with a computational cost of $\mathcal{O}(m^2)$ [32]; see also [8].

Case (ii). Single frequency-dependent right-hand side. In this case (1.1) becomes

$$(\lambda_j^2 A + \lambda_j B + C)x = b(\lambda_j), \quad j = 1,\dots,s,$$

where $b(\lambda_j)$ may depend on λ_j nonlinearly. The linearized problem takes the form

$$(\mathcal{A} + \lambda_j\mathcal{B})z = d(\lambda_j), \quad d(\lambda_j) = [b(\lambda_j); 0].$$

Solving all systems independently will make the cost grow approximately linearly with the number of parameters, and it will not make use of the common matrices $\mathcal{A}$ and $\mathcal{B}$. In [2] a method was devised, for real symmetric positive definite matrix $\mathcal{A}\mathcal{B}^{-1}$, that deals with the multiple right-hand side and associated shift. It would be interesting to see whether a generalization to our setting would be possible.

Alternatively, all systems may be treated simultaneously, yielding the following Sylvester equation:

$$\mathcal{A}\mathcal{B}^{-1}\widehat{Z} + \widehat{Z}\Lambda = \widehat{D}, \qquad \widehat{D} = [d(\lambda_1),\dots,d(\lambda_s)] \in \mathbb{C}^{2n\times s}, \qquad \Lambda = \mathrm{diag}(\lambda_1,\dots,\lambda_s),$$

with $\widehat{Z} = \mathcal{B}^{-1}[z_1,\dots,z_s]$. Block versions of Krylov subspace methods may be adapted to solve the problem [31]. More precisely, the block Krylov subspace $\mathcal{K}_{sk}(\mathcal{A}\mathcal{B}^{-1}, \widehat{D})$ is generated, and the approximation to the Sylvester equation is carried out on the projected problem [31]. The implementation is completely analogous to that for constant multiple loads. However, the least squares problem to be solved for each frequency is

$$\min_{y\in\mathbb{C}^{ks}} \|(E_1\chi)_{:,j} - (\mathcal{L}_k\mathcal{U}_k + \lambda_j\tilde{I}_{ks})y\|, \qquad y = y(\lambda_j), \qquad j = 1,\dots,s. \tag{9.4}$$

Contrary to (9.3), the computational cost for updating the solution y_k at each iteration k depends on the number of parameters and is proportional to s^2.

[2] Given $G \in \mathbb{C}^{n\times n}$, $\widehat{D} \in \mathbb{C}^{n\times m}$, a block Krylov subspace is given by $\mathcal{K}_{mk}(G, \widehat{D})$ =span$\{\widehat{D}, G\widehat{D}, \dots, G^{k-1}\widehat{D}\}$ and its dimension is at most mk.

The preference between the two approaches depends on the magnitude of s and m. If very many parameter values are treated simultaneously, then one may want to consider, if possible, splitting the load vector b as in (9.1) and solving with F as the right-hand side matrix. This may be particularly convenient when the system is characterized by few loading points, so that $m = \mathcal{O}(1)$.

10. Conclusions.

In this paper we have proposed a method for solving a linear system with a quadratic parameter that varies in a wide range. The method relies on the symmetry of the linearized problem with respect to an indefinite inner product. Shift invariance of the subspace in which the problem is projected is exploited in order to limit the computational cost growth as the number of parameters increases.

We have addressed the problem of inexactly solving the linear system arising at each iteration of the solver and uncovered some of the major difficulties related to its efficient manipulation. The performance of the inner-outer method seems to depend on the conditioning of matrix A in the original problem. Strategies for optimizing the inner system solution would certainly lead to better performance; for instance, we have not explored ordering strategies of the coefficient matrix entries, other than lexicographic, which may lead to more efficient preconditioning procedures for the inner system.

In summary, our numerical experience shows that if systems with the leading matrix A can be solved with a sparse direct solver, then the proposed method greatly outperforms other currently available techniques. When memory limitations force us to use an inner-outer procedure, then the performance of the method will in general degrade, the amount of degradation depending on the conditioning of A. We have shown, however, that when A is not too severely ill-conditioned, the inner-outer method remains highly competitive in case several parameter values need be considered. Ill-conditioning or singularity of A may be overcome by shifting the linearized formulation, although such an approach may entail considerable computational cost overhead, due to a less convenient sparsity pattern of the matrix.

It would be interesting to explore whether an efficient method that directly attacks the original problem, without passing through the linearization process, could be devised (cf., e.g., [36] for the eigenvalue setting). Such a method would allow us to deal with higher orders in λ.

Acknowledgments. All Fortran tests were done using the facilities of the Istituto di Analisi Numerica del CNR, Pavia, Italy. Stimulating discussions with Françoise Tisseur are acknowledged. We thank the referees for their constructive criticism.

REFERENCES

[1] AEA Technology, *Harwell Subroutine Library*, Harwell Laboratory, Oxfordshire, England, 1995.

[2] T. F. Chan and M. K. Ng, *Galerkin projection methods for solving multiple linear systems*, SIAM J. Sci. Comput., 21 (1999), pp. 836–850.

[3] E. Chow and Y. Saad, *Private communication*, 1998.

[4] A. R. Collar and A. Simpson, *Matrices and Engineering Dynamics*, Ellis Horwood, Chichester, Halsted Press, New York, 1987.

[5] P. Feldmann and R. W. Freund, *Efficient linear circuit analysis by Padé approximation via the Lanczos process*, IEEE Trans. Computer-Aided Design, 14 (1995), pp. 639–649.

[6] A. Feriani, F. Perotti, and V. Simoncini, *Iterative system solvers for the frequency analysis of linear mechanical systems*, Comput. Methods Appl. Mech. Engrg., 190 (2000), pp. 1719–1739.

[7] R. Freund, *On conjugate gradient type methods and polynomial preconditioners for a class of complex non-Hermitian matrices*, Numer. Math., 57 (1990), pp. 285–312.

[8] R. W. Freund and Z. Bai, *A symmetric band Lanczos process based on coupled recurrences and some applications*, SIAM J. Sci. Comput., 23 (2001), pp. 542–562.

[9] R. W. Freund, M. H. Gutknecht, and N. M. Nachtigal, *An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices*, SIAM J. Sci. Comput., 14 (1993), pp. 137–158.

[10] R. W. Freund, *Conjugate gradient-type methods for linear systems with complex symmetric coefficient matrices*, SIAM J. Sci. Stat. Comput., 13 (1992), pp. 425–448.

[11] R. W. Freund and N. M. Nachtigal, *An implementation of the QMR method based on coupled two-term recurrences*, SIAM J. Sci. Comput., 15 (1994), pp. 313–337.

[12] R. W. Freund and N. M. Nachtigal, *Software for simplified Lanczos and QMR algorithms*, Appl. Numer. Math., 19 (1995), pp. 319–341.

[13] I. Gohberg, P. Lancaster, and L. Rodman, *Matrices and Indefinite Scalar Products*, Oper. Theory Adv. Appl. 8, Birkhäuser Verlag, Basel, 1983.

[14] I. Gohberg, L. Rodman, and P. Lancaster, *Matrix Polynomials*, Academic Press, New York, 1982.

[15] G. H. Golub and Q. Ye, *Inexact preconditioned conjugate gradient method with inner-outer iteration*, SIAM J. Sci. Comput., 21 (1999), pp. 1305–1320.

[16] P. Guillaume, *Nonlinear eigenproblems*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 575–595.

[17] N. J. Higham, *Factorizing complex symmetric matrices with positive definite real and imaginary parts*, Math. Comput., 67 (1998), pp. 1591–1599.

[18] A. Ibrahimbegovic, H. C. Chen, E. L. Wilson, and R. L. Taylor, *Ritz method for dynamic analysis of large discrete linear systems with non-proportional damping*, Earthquake Eng. and Struct. Dyn., 19 (1990), pp. 877–889.

[19] M. Kuzuoglu and R. Mittra, *Finite element solution of electromagnetic problems over a wide frequency range via the Padé approximation*, Comput. Methods Appl. Mech. Engrg., 169 (1999), pp. 263–277.

[20] P. Lancaster, *Lambda-Matrices and Vibrating Systems*, Pergamon Press, New York, 1966.

[21] The MathWorks, Inc., *MATLAB User's Guide*, Natick, MA, 1998.

[22] K. Meerbergen, *The Solution of Parametrized Linear Systems from Mechanical Systems. Part* I: *Linear Parameter*, Tech. Rep. KM-2000-2, Free Field Technologies, 2000, pp. 1–20.

[23] V. Mehrmann and D. Watkins, *Structure-preserving methods for computing eigenpairs of large sparse skew-Hamiltonian/Hamiltonian pencils*, SIAM J. Sci. Comput., 22 (2001), pp. 1905–1925.

[24] G. Meurant, *Computer Solution of Large Linear Systems*, North-Holland, Amsterdam, 1999.

[25] B. N. Parlett and H. C. Chen, *Use of indefinite pencils for computing damped natural modes*, Linear. Algebra Appl., 140 (1990), pp. 53–88.

[26] F. Perotti, *Analytical and numerical techniques for the dynamic analysis of non-classically damped linear systems*, Soil Dynamics and Earthquake Eng., 13 (1994), pp. 197–212.

[27] Y. Saad, *A flexible inner-outer preconditioned GMRES*, SIAM J. Sci. Comput., 14 (1993), pp. 461–469.

[28] Y. Saad, *ILUT: A dual threshold incomplete ILU factorization*, Numer. Linear Algebra Appl., 1 (1994), pp. 387–402.

[29] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS, Boston, 1996.

[30] J. E. Santos and P. M. Gauzellino, *Parallel algorithms for wave propagation in fluid-saturated porous media*, in Computational Mechanics, New Trends and Applications, CINME, Barcelona, Spain, 1998, pp. 1–13.

[31] V. Simoncini, *On the numerical solution of $AX - XB = C$*, BIT, 36 (1996), pp. 814–830.

[32] V. Simoncini, *A stabilized QMR version of block BiCG*, SIAM J. Matrix Analysis and Appl., 18 (1997), pp. 419–434.

[33] V. Simoncini, *Linear systems with a quadratic parameter and application to structural dynamics*, in Iterative methods in Scientific Computation II, IMACS Ser. Computational and Appl. Math. 5, IMACS, New Brunswick, NJ, 1999, pp. 451–461.

[34] V. Simoncini and E. Gallopoulos, *A hybrid block gmres method for nonsymmetric systems with multiple right-hand sides*, J. Comput. Appl. Math., 66 (1996), pp. 457–469.

[35] D. Skoogh, *A Rational Krylov Method for Model Order Reduction*, Tech. Rep. 1998-47, Department of Mathematics, Chalmers University, Göteborg, Sweden, 1998.

[36] G. Sleijpen, A. Booten, D. Fokkema, and H. V. der Vorst, *Jacobi-Davidson type methods for generalized eigenproblems and polynomial eigenproblems*, BIT, 36 (1996), pp. 595–633.

[37] D. B. Szyld and J. A. Vogel, *FQMR: A flexible quasi-minimal residual method with inexact preconditioning*, SIAM J. Sci. Comput., 23 (2001), pp. 363–380.

[38] F. Tisseur and N. J. Higham, *Structured pseudospectra for polynomial eigenvalue problems, with applications*, SIAM J. Matrix Anal. Appl., 23 (2001), pp. 187–208.

[39] F. Tisseur and K. Meerbergen, *The quadratic eigenvalue problem*, SIAM Rev., 43 (2001), pp. 235–286.

[40] H. A. van der Vorst and J. B. M. Melissen, *A Petrov-Galerkin type method for solving Ax=b, where A is symmetric complex*, IEEE Trans. Magnetics, 26 (1990), pp. 706–708.

[41] E. L. Wilson, M.-W. Yuan, and J. M. Dickens, *Dynamic analysis by direct superposition of Ritz vectors*, Earthquake Eng. and Struct. Dyn., 10 (1982), pp. 813–821.

SIAM J. SCI. COMPUT.
Vol. 23, No. 6, pp. 1898–1923

RESIDUAL AND BACKWARD ERROR BOUNDS IN MINIMUM RESIDUAL KRYLOV SUBSPACE METHODS*

CHRISTOPHER C. PAIGE† AND ZDENĚK STRAKOŠ‡

Abstract. Minimum residual norm iterative methods for solving linear systems $Ax = b$ can be viewed as, and are often implemented as, sequences of least squares problems involving Krylov subspaces of increasing dimensions. The minimum residual method (MINRES) [C. Paige and M. Saunders, *SIAM J. Numer. Anal.*, 12 (1975), pp. 617–629] and generalized minimum residual method (GMRES) [Y. Saad and M. Schultz, *SIAM J. Sci. Statist. Comput.*, 7 (1986), pp. 856–869] represent typical examples. In [C. Paige and Z. Strakoš, *Bounds for the least squares distance using scaled total least squares*, Numer. Math., to appear] revealing upper and lower bounds on the residual norm of any linear least squares (LS) problem were derived in terms of the total least squares (TLS) correction of the corresponding scaled TLS problem. In this paper theoretical results of [C. Paige and Z. Strakoš, *Bounds for the least squares distance using scaled total least squares*, Numer. Math., to appear] are extended to the GMRES context. The bounds that are developed are important in theory, but they also have fundamental practical implications for the finite precision behavior of the modified Gram–Schmidt implementation of GMRES, and perhaps for other minimum norm methods.

Key words. linear equations, eigenproblem, large sparse matrices, iterative solution, Krylov subspace methods, Arnoldi method, generalized minimum residual method, modified Gram–Schmidt, least squares, total least squares, singular values

AMS subject classifications. 65F10, 65F20, 65F25, 65F50, 65G05, 15A42

PII. S1064827500381239

1. Introduction.

Consider a system of linear algebraic equations $Ax = b$, where A is a given n by n (unsymmetric) nonsingular matrix and b an n-dimensional vector. Given an initial approximation x_0, one approach to finding x is to first compute the initial residual $r_0 = b - Ax_0$. Using this, derive a sequence of Krylov subspaces $\mathcal{K}_k(A, r_0) \equiv \text{span}\{r_0, Ar_0, \ldots, A^{k-1}r_0\}$, $k = 1, 2, \ldots$, in some way, and look for approximate solutions $x_k \in x_0 + \mathcal{K}_k(A, r_0)$. Various principles are used for constructing x_k which determine various Krylov subspace methods for solving $Ax = b$. Similarly, Krylov subspaces for A can be used to obtain eigenvalue approximations or to solve other problems involving A.

Krylov subspace methods are useful for solving problems involving very large sparse matrices, since these methods use these matrices only for multiplying vectors, and the resulting Krylov subspaces frequently exhibit good approximation properties. The Arnoldi method [4] is a Krylov subspace method designed for solving the eigenproblem of unsymmetric matrices. The generalized minimum residual method (GMRES) [27] uses the Arnoldi iteration and adapts it for solving the linear system $Ax = b$. GMRES can be computationally more expensive per step than some other methods; see, for example, Bi-CGSTAB [30], QMR [8, 9] for unsymmetric A, and LSQR [20, 19] for unsymmetric or even rectangular A. However, GMRES is widely

*Received by the editors November 15, 2000; accepted for publication (in revised form) October 15, 2001; published electronically February 20, 2002.
http://www.siam.org/journals/sisc/23-6/38123.html

†School of Computer Science, McGill University, Montreal, Quebec, Canada, H3A 2A7 (paige@cs.mcgill.ca). This author's work was supported by NSERC of Canada grant OGP0009236.

‡Institute of Computer Science, Academy of Sciences of the Czech Republic, Pod Vodárenskou věží 2, 182 07 Praha 8, Czech Republic (strakos@cs.cas.cz). This author's work was supported by the GA AV CR under grant A1030103. Part of this work was performed during the academic years 1998/1999 and 1999/2000 while this author was visiting Emory University, Atlanta, GA.

used for solving linear systems arising from discretization of partial differential equations, and it is also interesting to study, since it *does* in theory minimize the 2-norm of the residual $\|r_k\| = \|b - Ax_k\|$ over $x_k \in x_0 + \mathcal{K}_k(A, r_0)$ at each step. Thus, theoretical results on GMRES can, for example, provide lower bounds for the residuals of other methods using the same Krylov subspaces. GMRES is also interesting to study computationally, especially since a strong relationship has been noticed between convergence of GMRES and loss of orthogonality among the Arnoldi vectors computed via (finite precision) modified Gram–Schmidt (MGS) orthogonalization; see [11, 24]. An understanding of this will be just as important for the practical use of the Arnoldi method as it will be for GMRES itself.

This project is complicated, so we give an introduction involving simplified results. Given an initial approximation x_0 to the solution x of $Ax = b$, we form the residual

$$r_0 = b - Ax_0, \qquad \rho_0 = \|r_0\|, \qquad v_1 = r_0/\rho_0,$$

and use v_1 to initiate the Arnoldi process [4]. In theory, after k steps this produces

$$V_{k+1} = [v_1, v_2, \ldots, v_{k+1}], \quad V_{k+1}^T V_{k+1} = I_{k+1}, \quad \operatorname{span}\{v_1, \ldots, v_{k+1}\} = \mathcal{K}_{k+1}(A, r_0).$$

At each step GMRES takes $x_k = x_0 + V_k y_k$ as the approximation to the solution x, which gives the residual $r_k = b - Ax_k$. GMRES uses that y_k which in theory minimizes the 2-norm of this residual, so

$$\|r_k\| = \min_y \|r_0 - AV_k\, y\| = \min_y \|[v_1\rho_0, AV_k] \begin{bmatrix} 1 \\ -y \end{bmatrix} \|.$$

So far this is rigorous and well known, but now we give some ideas in approximate form, so that they will be easier to follow. It is the purpose of this paper to show for the ratio of the largest to smallest singular value (condition number) $\kappa([v_1\rho_0, AV_k])$, which increases with k, and the normwise relative backward error

$$\beta(x_k) \equiv \frac{\|r_k\|}{\|b\| + \|A\| \cdot \|x_k\|}, \tag{1.1}$$

which tends to decrease with k until it is eventually zero, that with exact arithmetic we have something like the intriguing relationship

$$\beta(x_k)\, \kappa([v_1\rho_0, AV_k]) = O(1). \tag{1.2}$$

In later sections we will develop rigorous theory for the more precise version of this. There the columns of $[v_1\rho_0, AV_k]$ in $\kappa(\cdot)$ are scaled, and a certain condition must be satisfied. We will argue that the precise version probably also holds even in finite precision arithmetic and present convincing numerical examples supporting this hypothesis.

Now we explain why (1.2) is important. An efficient, and the most usual way of computing the Arnoldi vectors $v_1, v_2, \ldots, v_{k+1}$ for large sparse unsymmetric A, is to use the MGS orthogonalization. Unfortunately, in finite precision computations this leads to loss of orthogonality among these MGS Arnoldi vectors. If these MGS Arnoldi vectors are used in GMRES we have MGS GMRES. We want to show that MGS GMRES succeeds despite the loss of orthogonality among the computed MGS Arnoldi vectors. A similar hypothesis was published in [11, 24] with a justification based on the link between loss of orthogonality among the Arnoldi vectors and the size of the GMRES relative residual. Here is how we hope to prove a significantly stronger statement in [17] by using what is essentially the result (1.2) of this paper as a fundamental intermediate step.

Following the important work [5] of Björck, and that of Walker [32], the papers [7] and [11] showed a relationship between the *finite precision* loss of orthogonality in the MGS Arnoldi vectors and the condition number $\kappa([v_1\rho_0, AV_k])$. In particular, unless A is extremely ill-conditioned (close to numericaly singular), for computed quantities

$$\text{(1.3)} \quad \|I - V_{k+1}^T V_{k+1}\|_F \leq \kappa([v_1\rho_0, AV_k])\, O(\epsilon), \quad \epsilon \text{ the computer roundoff unit}$$

(where subscript F denotes the Frobenius norm). Combining it with a finite precision version of (1.2) would show

$$\text{(1.4)} \qquad \frac{\|r_k\| \cdot \|I - V_{k+1}^T V_{k+1}\|_F}{\|b\| + \|A\| \cdot \|x_k\|} \leq \beta(x_k)\, \kappa([v_1\rho_0, AV_k])\, O(\epsilon) \;=\; O(\epsilon).$$

This would imply that it is impossible to have a significant loss of orthogonality until the normwise relative backward error is very small. It could then be shown that there would be no meaningful deterioration in the rate of convergence, and significant loss of orthogonality would imply convergence and backward stability of MGS GMRES. These results would then be somewhat analogous to those shown for the Lanczos method for the symmetric eigenproblem, where significant loss of orthogonality implied that at least one eigenvalue had been found to about machine precision, and the first eigenvalues to converge did so with no meaningful deterioration in rate of convergence; see [16]. Perhaps the ideas here could be combined with some of those from [16] to prove how the MGS Arnoldi method is affected by rounding errors.

If we can prove a result like (1.4), we will be able to justify theoretically the well-known observation that, unless the matrix A is extremely ill-conditioned, MGS GMRES competes successfully in both the rate of convergence and the final accuracy with the more expensive GMRES implementation based on the Householder reflections (HH GMRES)[31]. HH GMRES was proved backward stable in [7]. That proof relied upon the fact that the Householder reflections keep the loss of orthogonality among the computed Arnoldi vectors close to the machine precision. Orthogonality among the Arnoldi vectors can be lost using MGS GMRES finite precision computations. Therefore the results from [7] could not be extended to MGS GMRES, and a different approach had to be used.

Despite its backward stability, HH GMRES is not widely used. A popular justification for this is based on the *numerical stability* versus *computational efficiency* argument: It is generally believed that HH GMRES is favorable numerically, but the cheaper MGS GMRES is accepted (sometimes with a fear of a possible unspecified loss of accuracy) as a standard for practical computations. One aim of our work is to eliminate that fear.

This paper is the third of a sequence starting with [22], which revised the fundamentals of the scaled total least squares theory. The subsequent paper [21] produced general purpose bounds we will use here and in [17]. The present paper proves theoretical results *motivated* by the abovementioned finite precision behavior of MGS GMRES but assumes *exact* arithmetic in all the proofs. Finite precision analogies of the statements proven here will require detailed rounding error analyses, and these are intended for the planned paper [17]. Thus, when completed, we think the work in [21], in here, and in [17] will represent a substantial step forward in our understanding of MGS orthogonalization in Krylov subspace methods and will also lead to a full justification for MGS GMRES computations. We also hope it will produce tools that will help in the analysis of MGS Arnoldi computations. We would like to investigate whether the MGS Arnoldi method still gives accurate approximations to eigenvalues, but we will not consider this here.

Since the results in this paper assume exact arithmetic, they are independent of any particular implementation of the GMRES method. They apply to any mathematically equivalent residual minimizing Krylov subspace method (such as the MINRES method for symmetric indefinite systems). Some mathematically equivalent variants of the GMRES method are described in [15, 25]. In most practical applications some acceleration technique must be applied to improve convergence of the basic method. For historical reasons such acceleration techniques are frequently and imprecisely called preconditioning. Assuming exact arithmetic, preconditioning of a given method is equivalent to the application of the (basic) method to some modified (preconditioned) system. In this paper we assume, with no loss of generality, that A represents the matrix and b the right-hand side of the preconditioned system. For simplicity of notation we assume that A and b are real. Reformulation to the general complex case is obvious.

The paper is organized as follows. In section 2 we will give the necessary mathematics of GMRES, while in section 3, which represents the main connection with the preceding papers [22] and [21], we will present bounds for the GMRES residual (Theorem 3.1). Section 4 will give an extreme example which shows that the assumption (3.5) required in Theorem 3.1 need not hold up until the very last step of the GMRES iteration. This is, of course, a highly contrived situation and not indicative of any realistic problem we have encountered. Section 5 will explain in more detail just why the bounds from section 3 are so important for our understanding of GMRES and related methods. We will prove Theorem 5.1, which is the precise version of (1.2) and represents the main result of this paper. Section 6 will discuss its consequences in light of possible scalings. Section 7 will display some computational results and section 8 will present concluding remarks.

In the paper we will use $\sigma_i(X)$ to denote the ith largest singular value of X, use $\kappa(X)$ to be the ratio of the largest to the smallest singular value of X, and refer to $\kappa(X)$ briefly as the condition number of X. The vector of elements i to j of a vector y will be denoted $y_{i:j}$, and e_j denotes the jth column of the unit matrix I. We will use $\|\cdot\|$ to denote the 2-norm and $\|\cdot\|_F$ to denote the Frobenius norm. Several quantities used in our bounds will depend on the iteration step k. For simplicity of notation we sometimes omit the explicit reference to the iteration step when the dependence is clear from the context and need not be stressed for any particular reason.

As explained above, this paper proves the precise version of (1.2), which is the fundamental intermediate step of the whole project, and it assumes exact arithmetic in all the proofs. However, the underlying discussion of MGS GMRES finite precision behavior motivates the whole work and affects most of the particular considerations in this paper. Though we separate the exact arithmetic results from the finite precision arithmetic discussion as much as possible, we cannot split them entirely. Scaling, for example, affects both (exact precision) bounds for the GMRES residual norm developed in this paper and finite precision bounds for loss of orthogonality in the Arnoldi process. Any discussion of scaling must consider both aspects, which are generally in conflict. When it will be helpful, we will use the word “ideally” to refer to a result that would hold using exact arithmetic, and “computationally” or “numerically” to a result of a finite precision computation.

2. The GMRES method. For a given n by n (usually unsymmetric) nonsingular matrix A and n-vector b, we wish to solve $Ax = b$. Given an initial approximation x_0 we form the residual

$$r_0 = b - Ax_0, \qquad \rho_0 = \|r_0\|, \qquad v_1 = r_0/\rho_0, \tag{2.1}$$

and use v_1 to initiate the Arnoldi process [4]. At step k this forms Av_k, orthogonalizes it against $v_1, v_2, \ldots, v_k$, and if the resulting vector is nonzero, normalizes it to give v_{k+1}, giving ideally

$$AV_k = V_{k+1}H_{k+1,k}, \quad V_{k+1}^T V_{k+1} = I_{k+1}, \quad V_{k+1} = [v_1, v_2, \ldots, v_{k+1}]. \tag{2.2}$$

Here $H_{k+1,k}$ is a $k+1$ by k upper Hessenberg matrix with elements h_{ij}, where $h_{j+1,j} \neq 0$, $j = 1, 2, \ldots, k-1$. If at any stage $h_{k+1,k} = 0$ we would stop with $AV_k = V_k H_{k,k}$. In this case all the eigenvalues of $H_{k,k}$ are clearly eigenvalues of A. When $h_{k+1,k} \neq 0$ the eigenvalues of $H_{k,k}$ are approximations to some of those of A, and this gives the Arnoldi method [4]. Computationally, we are unlikely to reach a k such that $h_{k+1,k} = 0$, and for solution of equations we stop when we assess the norm of the residual (ideally given as below in (2.7)) is small enough.

In general, at each step we take $x_k = x_0 + V_k y_k$ as our approximation to the solution x, which gives the residual

$$\begin{aligned} r_k &= b - Ax_k = r_0 - AV_k y_k = v_1\rho_0 - V_{k+1}H_{k+1,k}\, y_k \\ &= V_{k+1}(e_1\rho_0 - H_{k+1,k}\, y_k). \end{aligned} \tag{2.3}$$

GMRES seeks y_k which minimizes this residual by solving the linear least squares problem

$$\|r_k\| = \min_y \|r_0 - AV_k\, y\| = \min_y \|v_1\rho_0 - AV_k\, y\|. \tag{2.4}$$

Using (2.2) and (2.3), (2.4) can be formulated as the least squares problem with the upper Hessenberg matrix $H_{k+1,k}$

$$\|r_k\| = \min_y \|e_1\rho_0 - H_{k+1,k}\, y\|. \tag{2.5}$$

To solve (2.5) we apply orthogonal rotations (J_i being the rotation in the $i, i+1$ plane through the angle θ_i) sequentially to $H_{k+1,k}$ to bring it to upper triangular form S_k:

$$J_k \cdots J_2 J_1 H_{k+1,k} = Q_k^T H_{k+1,k} = \begin{pmatrix} S_k \\ 0 \end{pmatrix}.$$

The vectors y_k and r_k ideally then satisfy

$$S_k y_k = (Q_k^T e_1\rho_0)_{1:k}, \tag{2.6}$$

$$\begin{aligned} \|r_k\| &= |e_{k+1}^T Q_k^T e_1 \rho_0| \\ &= |\xi_1\xi_2 \cdots \xi_k|\, \|r_0\|, \qquad \xi_i = \sin\theta_i. \end{aligned} \tag{2.7}$$

The measure (2.7) of the (nonincreasing) residual norm is available without determining y_k, and since y_{k+1} will usually differ in every element from y_k, it would seem preferable to avoid determining y_k or x_k until we decide the residual norm (2.7) is small enough to stop. Computationally, however, it is not clear that we can base the stopping criterion on (2.7) alone. The step from (2.4) to (2.5) requires orthogonality of the columns of V_{k+1}. However, even if orthogonality of the Arnoldi vectors computed using finite precision arithmetic is well preserved (as in HH GMRES), (2.7) will not hold for the computed quantities after the residual norm drops near the final accuracy level; see [7].

Finally, little has been published about the choice of the initial approximation x_0. In many cases $x_0 = 0$ is recommended or considered. For $x_0 = 0$ we have $r_0 = b$ and trivially $\|r_0\| \le \|b\|$. This last condition seems very natural and should always be imposed. For a nonzero x_0 it may easily happen that $\|r_0\| > \|b\|$ (even $\gg$ for some problems), and any such x_0 is a poor initial approximation to the solution x. Hegedüs [13] suggested that a simple way around this difficulty is to rescale the initial approximation. Given a preliminary initial guess x_p, it is easy to determine the scaling parameter $\zeta_{\min}$ such that

$$\|r_0\| = \|b - Ax_p\zeta_{\min}\| = \min_{\zeta} \|b - Ax_p\zeta\|, \qquad \zeta_{\min} = \frac{b^T A x_p}{\|Ax_p\|^2}\,. \tag{2.8}$$

Thus, by setting $x_0 = x_p\zeta_{\min}$ we ensure $\|r_0\| \le \|b\|$. The extra cost for implementing this little trick is negligible; it should be used in GMRES computations whenever a nonzero x_0 is considered. For some related comments see the discussion concerning the experiments in section 7.

We point out that the previous paragraph does not mean that an arbitrary x_p with (2.8) gives a proper initial approximation x_0. Our general feeling is that, even with (2.8), a nonzero x_0 should not be used unless there is a good reason for preferring it over $x_0 = 0$. It has been observed that without such additional justification, a choice of nonzero x_0 satisfying $\|r_0\| \le \|b\|$ can significantly slow down GMRES convergence [28].

3. Bounds for the GMRES residuals.

From the previous section it is clear that GMRES can be seen as a sequence of least squares problems (2.4) involving Krylov subspaces of increasing dimensions. In [21] we considered the overdetermined approximate linear system $Bu \approx c$ and bounded the least squares (LS) residual

$$\text{LS residual} \quad \equiv \quad \min_{r,y} \|r\|_2 \quad \text{subject to} \quad By = c - r \tag{3.1}$$

from above and from below in terms of the scaled total least squares (STLS) distance

$$\text{STLS distance} \quad \equiv \quad \min_{s,E,z} \|[s, E]\|_F \quad \text{subject to} \quad (B + E)z\gamma = c\gamma - s, \tag{3.2}$$

where $\gamma > 0$ is the scaling parameter. The bounds from [21] say nothing about an iterative method, or where B or c come from, and so they are general results. In order to apply the results from [21] to GMRES we have to identify B, c, and γ with the proper quantities in GMRES. We have several choices, but as yet there is no choice which is clearly superior to the others. Therefore we will formulate the bounds in the following theorem and in section 5 in a general way. Particular scalings (γ and D_k in the theorem) will be discussed in section 6.

To obtain useful bounds for the kth step of GMRES, we consider $c = r_0 = v_1\rho_0$ and $B = B_k = AV_kD_k$, where D_k is a diagonal matrix of positive scaling coefficients ($D_k > 0$). Note that the column scaling by the diagonal matrix D_k does not change the optimal residual r_k (see (2.4)) and

$$\|r_k\| = \min_y \|v_1\rho_0 - AV_k\, y\| = \min_{D_k^{-1}y} \|c - B_k\,(D_k^{-1}y)\|\,. \tag{3.3}$$

Clearly, for this c and B_k the solution of (3.1) is $D_k^{-1}y_k$, where y_k is the solution of the LS problem (2.4). The column scaling matrix D_k will prove useful later. Note that, by construction, B_k has full column rank.

We now give bounds on the $\|r_k\|$ in GMRES, together with bounds on an important ratio δ_k.

THEOREM 3.1. *Given a scalar $\gamma > 0$ and a positive diagonal matrix D_k, use $\sigma(\cdot)$ to denote singular values and $\|\cdot\|$ to denote 2-norms. Let the n by n nonsingular matrix A, the vectors r_0, y_k, and r_k, the scalar ρ_0, and the matrix V_k be as in the GMRES algorithm* (2.1)–(2.5) *using exact arithmetic, and let AV_k have rank k. Denote $B_k = AV_kD_k$, $c = v_1\rho_0$, and define*

$$\delta_k \equiv \delta_k(\gamma, D_k) \equiv \sigma_{k+1}([c\gamma, B_k])/\sigma_k(B_k) = \sigma_{k+1}([v_1\rho_0\gamma, AV_kD_k])/\sigma_k(AV_kD_k). \tag{3.4}$$

If

$$v_1 \not\perp \{\textit{left singular vector subspace of } B_k \textit{ corresponding to } \sigma_{min}(B_k)\}, \tag{3.5}$$

then $\delta_k < 1$ and

$$\begin{aligned} \mu_L \equiv \sigma_{k+1}([c\gamma, B_k])\,\{\gamma^{-2} + \|D_k^{-1}y_k\|^2\}^{\frac{1}{2}} \quad &\le \quad \|r_k\| \\ \le\; \mu_U \equiv \sigma_{k+1}([c\gamma, B_k])\,\{\gamma^{-2} + (1-\delta_k^2)^{-1}\|D_k^{-1}y_k\|^2\}^{\frac{1}{2}}&, \end{aligned} \tag{3.6}$$

$$\frac{\|r_k\|}{\left\{\gamma^{-2} + \frac{\|D_k^{-1}y_k\|^2}{1-\delta_k^2}\right\}^{\frac{1}{2}}\sigma_k(B_k)} \le \delta_k \le \frac{\|r_k\|}{\{\gamma^{-2} + \|D_k^{-1}y_k\|^2\}^{\frac{1}{2}}\sigma_k(B_k)}, \tag{3.7}$$

$$\frac{\gamma\|r_k\|}{\|[c\gamma, B_k]\|} \le \delta_k \le \frac{\gamma\|r_k\|}{\sigma_k([c\gamma, B_k])} \le \frac{\gamma\|r_k\|}{\sigma_k(B_k)} \le \frac{\gamma\|r_k\|}{\sigma_n(A)\sigma_k(D_k)}. \tag{3.8}$$

Proof. We see $c\gamma = v_1\rho_0\gamma$ and $B_k = AV_kD_k$ satisfy the conditions and assumptions of Theorem 4.1 of [21] for any $\gamma > 0$, and from (3.3) we see that r_k and $D_k^{-1}y_k$ correspond to r and y in (3.1); so the theorem holds with [21, (4.4)] giving (3.6) and its equivalent (3.7), while Corollary 6.1 of [21] gives all but the last inequality in (3.8), which holds since $V_k^HV_k = I$. □

Note that apart from the last inequality in (3.8) the result does not depend on orthogonality of the columns of V_k, since Theorem 4.1 of [21] requires nothing of $B = B_k = AV_kD_k$ here except that it has full column rank. The only requirement is for $\|r_k\|$ to be a minimum (see (2.4), (3.1), and (3.3)) at each step. It should also be pointed out that due to monotonicity of $\|r_k\|$ from GMRES, possible oscillations in the upper bound (3.6) can be eliminated by taking the minimum

$$\|r_k\| \;\le\; \min_{j=1,\dots,k}\{\sigma_{j+1}([v_1\rho_0\gamma, B_j])\,\{\gamma^{-2} + (1-\delta_j^2)^{-1}\|D_j^{-1}y_j\|^2\}^{\frac{1}{2}}\}. \tag{3.9}$$

In the paper [21] we compared the bounds for the LS residual used here with other existing bounds. For example, [21, Corollary 5.1] gives

$$\begin{aligned} \gamma\|r_k\| &\le \delta_k\,\{\|c\|^2\gamma^2 + \sigma_k^2(B_k) - \sigma_{k+1}^2([c\gamma, B_k])\}^{\frac{1}{2}} \\ &\le \delta_k\,\{\|c\|^2\gamma^2 + \sigma_k^2(B_k)\}^{\frac{1}{2}}. \end{aligned} \tag{3.10}$$

As stated in [21, section 5], our bounds in (3.6) can be significantly better than those from (3.10). They are also easily applicable to the problem investigated in this paper. We will therefore not examine (3.10) and the other possible bounds which can be derived from (3.10) here.

It will be important to examine the tightness of the bounds (3.6). The following corollary is an immediate consequence of [21, Corollary 4.2].

COROLLARY 3.2. *Under the conditions and assumptions of Theorem* 3.1*, and using the notation there together with*

$$\eta \equiv \frac{\|r_k\| - \mu_L}{\|r_k\|}, \qquad \zeta \equiv \frac{\mu_U - \mu_L}{\|r_k\|}, \tag{3.11}$$

we have the following bound on η *and* ζ*:*

$$0 \leq \eta \ \leq \ \zeta \ \leq \ \frac{\gamma^2\|D_k^{-1}y_k\|^2}{2+\gamma^2\|D_k^{-1}y_k\|^2} \cdot \frac{\delta_k^2}{1-\delta_k^2} \to 0 \quad as \quad \gamma \to 0, \tag{3.12}$$

where the upper bound goes to zero at least as fast as $O(\gamma^4)$ *(see* (3.8)*).* □

The assumption (3.5) is not necessary for proving the bounds (3.6)–(3.8) and (3.12). From the proof of [21, Theorem 4.1] it is clear that these bounds require only $\delta_k < 1$, and, moreover, the lower bound in (3.6), the upper bound in (3.7) and the bounds in (3.8) also hold if $\delta_k = 1$. (The upper bound in (3.6) and the lower bound (3.7) become ∞ and 0 when $\delta_k = 1$, and so hold trivially.) Using (3.5), however, makes the theory clean and consistent. The assumption (3.5) is independent of scaling and it ensures that the bounds do not contain irrelevant quantities; see [22, Remark 4.3].

From (3.12) and (3.8) we see that small δ_k, γ, $\|r_k\|$ or $\|D_k^{-1}y_k\|/(1-\delta_k^2)$ ensures that the bounds (3.6) are not only very tight, but very tight in a relative sense. The tightness of the bounds depends in an important way on δ_k; for $\delta_k \ll 1$ we get the strong relationship from (3.6)

$$\|r_k\| \ \approx \ \sigma_{min}([v_1\rho_0\gamma, AV_kD_k])\,\{\gamma^{-2} + \|D_k^{-1}y_k\|^2\}^{\frac{1}{2}}. \tag{3.13}$$

We know $0 \leq \delta_k \leq 1$ from (3.4). If $\delta_k \approx 1$ the bounds in (3.6) and (3.7) become weak, so we need to see if $\delta_k \approx 1$ is possible. In the GMRES context δ_k will necessarily be small as $\|r_k\| \to 0$ (see (3.8)). Proper scaling can always ensure $\delta_k \ll 1$. (For a fixed D_k it was shown in [22, Corollary 4.1] that if (3.5) holds, then $\delta_k < 1$, δ_k increases and decreases with γ, and (3.8) shows $\gamma \to 0 \Rightarrow \delta_k \to 0$.) Using this argument, it appears at first that the disturbing case $\delta_k \approx 1$ can easily be eliminated from our discussion. It turns out, however, that this is not entirely true because the use of scaling also has disadvantages. We will see that we cannot use an arbitrarily small γ to ensure $\delta_k \ll 1$ without (potentially) damaging the tightness of the bounds for the loss of orthogonality among the Arnoldi vectors (the tightness of the scaled version of (1.3)). On the other hand, a scaling which might be appropriate from the point of view of the formulation of the main result (a scaled version of (1.2); see the following section) might at the same time increase the value of δ_k. The choice of scaling therefore represents a delicate task. Despite these subtle details, we will see that $\delta_k \approx 1$ represents a technical problem but not a serious conceptual difficulty. We will return to the detailed discussion of this point in section 6.

4. Delayed convergence of GMRES.

It is possible for convergence of GMRES to be very slow and stagnate entirely even with exact arithmetic. Suppose

$$A = [e_2\gamma_2, e_3\gamma_3, \ldots, e_n\gamma_n, e_1\gamma_1], \qquad b = e_1\|b\|, \qquad x_0 = 0,$$

for some $\gamma_i \neq 0$, $i = 1, \dots, n$; then in (2.1) and (2.2) for $k < n$

$$V_{k+1} = [e_1, e_2, \dots, e_{k+1}], \qquad H_{k+1,k} = [e_2\gamma_2, e_3\gamma_3, \dots, e_{k+1}\gamma_{k+1}],$$

and in (2.3) and (2.5)

$$y_k = 0, \qquad x_k = 0, \qquad r_k = r_0, \qquad k = 1, 2, \dots, n-1;$$

so any convergence at all is delayed until the solution is obtained at step $k = n$. Here we have $v_1 = e_1 \perp \mathcal{R}(AV_k)$ for $k < n$, so (3.5) does not hold and $\delta_k = 1$ for $k = 1, 2, \dots, n-1$. In fact (3.6) degenerates to $\|r_k\| = \|r_0\|$ for $k < n$.

5. Backward error theorem.

Now we show why we consider the bounds from Theorem 3.1 to be so important. This provides the scaled versions of (1.2)–(1.4). Remember that the scaled equivalents of the finite precision results (1.3)–(1.4) are only for motivation here, and the full proofs of these will be left to [17].

As noticed in [32] and used in [7] (see also [3]), the Arnoldi process (2.2) with (2.1) ideally gives the QR factorization of $[r_0, AV_k]$, since on defining upper triangular $R_{k+1} \equiv [e_1\rho_0, H_{k+1,k}]$ we see

$$[r_0, AV_k] = V_{k+1}[e_1\rho_0, H_{k+1,k}] = V_{k+1}R_{k+1}, \qquad V_{k+1}^T V_{k+1} = I_{k+1}. \tag{5.1}$$

By comparing this with (2.1) and (2.2), we see we may now refer to (5.1) as the Arnoldi process.

If the orthogonalization in (2.2) is carried out by the MGS technique, then it is straightforward to show that this MGS Arnoldi process provides V_{k+1} and R_{k+1}, which are *computationally* identical to those produced by the QR factorization of $[r_0, \widetilde{AV}_k]$ by MGS. Here, $\widetilde{AV}_k$ indicates that the multiplications Av_j, $j = 1, \dots, k$, are computed numerically. A parallel statement holds when classical Gram–Schmidt orthogonalization is used in (2.2).

With a computer using finite precision with unit roundoff ϵ, the computed vectors $v_1, v_2, \dots$ tend to lose orthogonality. It was shown by Björck [5] that using MGS in the QR factorization $C = QR$ computationally leads to Q such that

$$\|I - Q^T Q\|_F \leq \kappa(C)\, O(\epsilon).$$

(For convenience in numerical experiments we use the Frobenius norm.)

Thus from the discussion following (5.1), for the finite precision version of (2.2) using MGS we have (see (1.3))

$$\|I - V_{k+1}^T V_{k+1}\|_F \leq \kappa([v_1\rho_0, AV_k])\, O(\epsilon). \tag{5.2}$$

Note that $\kappa([v_1\rho_0, AV_k])$ is used here instead of $\kappa([r_0, \widetilde{AV}_k])$. Using $\kappa([v_1\rho_0, AV_k])$ simplifies further considerations; the difference between $\kappa([v_1\rho_0, AV_k])$ and $\kappa([r_0, \widetilde{AV}_k])$ is absorbed in the multiplicative factor $O(\epsilon)$. For the detailed justification see [7] and [11].

When MGS is used with exact arithmetic in (5.1), the resulting matrix V_{k+1} is invariant with respect to the column scaling in $[v_1\rho_0\gamma, AV_kD_k]$, where $\gamma > 0$ and D_k is a positive diagonal k by k matrix. It appears that, ignoring a small additional error of $O(\epsilon)$, the matrix V_{k+1} resulting from the *finite precision* MGS Arnoldi process (5.1) is invariant with respect to positive column scaling. This important result was noticed in [11, p. 711], and was partially exploited there. It can be justified by the following argument (which is a variant of the argument attributed to Bauer; see [33, pp. 129–130]). If the scaling factors are always powers of the base of the floating point arithmetic (powers of 2 for the IEEE FP arithmetic), then the resulting V_{k+1} computed in finite precision arithmetic using the MGS Arnoldi process (5.1) will be exactly the same as the V_{k+1} computed in finite precision arithmetic using the same MGS Arnoldi process for the scaled data $[r_0\gamma, AV_kD_k]$. If the scaling factors are not powers of the base of the floating point arithmetic, then there will be additional rounding errors proportional to unit roundoff ϵ. Apparently no formal proof of the last part has been given, so we hope to include one in [17].

If all the above are true, the loss of orthogonality among the MGS Arnoldi vectors computed via (5.1) with a computer using finite precision arithmetic with unit roundoff ϵ is bounded by

$$\|I - V_{k+1}^T V_{k+1}\|_F \le \kappa([v_1\rho_0\gamma, AV_kD_k])\, O(\epsilon) \tag{5.3}$$

for all $\gamma > 0$ and positive diagonal k by k matrices D_k. One possibility is to scale the columns of $[v_1\rho_0\gamma, AV_kD_k]$ so they have unit length. That is, take

$$\gamma = \rho_0^{-1}, \qquad D_k = \mathrm{diag}\,(\|Av_1\|^{-1}, \ldots, \|Av_k\|^{-1}) \equiv \mathrm{diag}\,(\|Av_j\|^{-1}). \tag{5.4}$$

The corresponding condition number and the bound (5.3) would then be no more than a factor $\sqrt{k+1}$ away from its minimum (see [29]), so this is nearly optimal scaling. Other convenient choices will be discussed in the next section. Extensive experimental evidence suggests that for the nearly optimal scaling (5.4), the bound (5.3) is tight, and usually

$$\|I - V_{k+1}^T V_{k+1}\|_F \approx \kappa([v_1\rho_0\gamma, AV_kD_k])\, O(\epsilon). \tag{5.5}$$

It was observed that when MGS was used in (2.2), leading to the MGS GMRES method (2.1)–(2.6), loss of orthogonality in V_{k+1} was accompanied by a small relative residual norm $\|r_k\|/\rho_0$; see [11]. That is, significant loss of orthogonality in MGS GMRES apparently did not occur before convergence measured by $\|r_k\|/\rho_0$ occurred. This fortuitous behavior was analyzed numerically in [11] and a partial explanation was offered there. A much stronger and more complete theoretical explanation of the observed behavior can be derived from the bounds (3.6)–(3.8). As a first step, $\|r_k\|/\rho_0$ must be replaced by a more appropriate convergence characteristic.

We will use the terminology (such as *normwise*) and results reported in [14, section 7.1]. The *backward error* for x_k as an approximate solution for $Ax = b$ is a measure of the amounts by which A and b have to be perturbed so that x_k is the exact solution of the perturbed system $(A + \Delta A)x_k = b + \Delta b$. The *normwise relative backward error* of x_k defined by

$$\beta(x_k) \equiv \min_{\beta,\Delta A,\Delta b}\{\beta \ :\ (A+\Delta A)x_k = b + \Delta b,\ \|\Delta A\| \le \beta\|A\|,\ \|\Delta b\| \le \beta\|b\|\}$$

was shown by Rigal and Gaches [23] (see [14, Theorem 7.1, p. 132]), to satisfy

$$\beta(x_k) = \frac{\|r_k\|}{\|b\| + \|A\| \cdot \|x_k\|} = \frac{\|\Delta A_{\min}\|}{\|A\|} = \frac{\|\Delta b_{\min}\|}{\|b\|}. \tag{5.6}$$

We strongly believe that if no other (more relevant and more sophisticated) criterion is available (such as in [1]), this relative backward error should always be preferred to the (relative) residual norm $\|r_k\|/\|r_0\| = \|r_k\|/\rho_0$ in (2.1) when measuring convergence of iterative methods. In practice $\|A\|$ has to be replaced by its approximation—when available—or simply by the Frobenius norm of A. The theoretical reasons for preferring the relative backward error are well known; see, for example, [2] and [14]. We will add some more practical arguments in section 7. In particular the residual norm can be very misleading and easily misinterpreted. It is surprising and somewhat alarming that $\|r_k\|/\rho_0$ remains in use as the main (and usually the only) indicator of convergence of iterative processes. This statement applies to the majority of computational results published by numerical analysts. Our results will put a new emphasis on the importance of the backward error. For GMRES and the other residual minimizing methods, this raises a key question. If the residual norm is somewhat in doubt as a measure of convergence, how does this affect the position of the minimal residual principle as one of the main principles on which practical Krylov subspace methods are based? The answer needs work, and its further discussion is beyond the scope of this paper. However, we do not expect that the position of the minimal residual principle will be considerably shaken by such an analysis; rather we think it will be reaffirmed. It seems that GMRES, though based on the minimal residual principle, also produces a very good (nearly optimal) backward error.

We will now describe our main observation. This illustrates and supports the main goal of our work on MGS GMRES, which is to prove a scaled version of (1.4). Consider a plot with two lines obtained from the MGS GMRES finite precision computation. One line represents the relative backward error $\|r_k\|/(\|b\| + \|A\| \cdot \|x_k\|)$ and the other the loss of orthogonality $\|I - V_{k+1}^T V_{k+1}\|_F$ (both plotted on the same logarithmic scale) as a function of the iteration step k. We have observed that these two lines are always very nearly reflections of each other through the horizontal line defined by their intersection. For a clear example of this, see the dashed lines in Figure 7.1. In other words, *in finite precision MGS GMRES computations, the product of the normwise relative backward error and the loss of orthogonality is (as a function of the iteration step) almost constant and equal to the order of the machine precision* ϵ. The goal of this paper and [17] is to present a theoretical proof of this observed fact, and its fundamental consequences, which are that orthogonality among the computed MGS Arnoldi vectors is effectively maintained until convergence and total loss of orthogonality implies convergence of the normwise relative backward error to $O(\epsilon)$, which is equivalent to (normwise) backward stability of MGS GMRES.

Using the results presented in [21] the main ideas are simple and elegant. The proof itself (as yet incomplete) is, however, technical and tedious. Therefore in this paper we restrict ourselves to proving and discussing exact arithmetic results about the product of the normwise relative backward error and the condition number $\kappa([v_1\rho_0\gamma, AV_kD_k])$; with finite precision arithmetic this condition number controls the numerical loss of orthogonality via (5.5). A detailed rounding error analysis, together with the results relating the genuine loss of orthogonality $\|I - V_{k+1}^T V_{k+1}\|_F$ to the relative backward error, is intended for [17].

In the following theorem the product of the normwise relative backward error of GMRES and the condition number of the scaled matrix $[v_1\rho_0\gamma, AV_kD_k]$ is bounded from below and from above. Note that the theorem assumes exact arithmetic and therefore the result holds for GMRES in general. The theorem is formulated for any $\gamma > 0$ and any positive diagonal D_k; bounds corresponding to the specific choices of γ and D_k will be given in section 6.

THEOREM 5.1. *Under the conditions and assumptions of Theorem* 3.1, *and using the notation there, let* $\sigma_1 \equiv \sigma_1([v_1\rho_0\gamma, AV_kD_k]) = \|[v_1\rho_0\gamma, AV_kD_k]\|$, $\kappa_k \equiv \kappa([v_1\rho_0\gamma, AV_kD_k])$. *Then*

$$\begin{aligned}
(5.7)\qquad \frac{\sigma_1}{\sqrt{2}} \cdot \frac{\{\gamma^{-2} + \|D_k^{-1}y_k\|^2\}^{\frac{1}{2}}}{\{\|b\|^2 + \|A\|^2\|x_k\|^2\}^{\frac{1}{2}}} &\le \sigma_1 \frac{\{\gamma^{-2} + \|D_k^{-1}y_k\|^2\}^{\frac{1}{2}}}{\|b\| + \|A\| \cdot \|x_k\|} \\
&\le \kappa_k \frac{\|r_k\|}{\|b\| + \|A\| \cdot \|x_k\|} \\
\le \sigma_1 \frac{\{\gamma^{-2} + (1-\delta_k^2)^{-1}\|D_k^{-1}y_k\|^2\}^{\frac{1}{2}}}{\|b\| + \|A\| \cdot \|x_k\|} &\le \sigma_1 \frac{\{\gamma^{-2} + (1-\delta_k^2)^{-1}\|D_k^{-1}y_k\|^2\}^{\frac{1}{2}}}{\{\|b\|^2 + \|A\|^2\|x_k\|^2\}^{\frac{1}{2}}}.
\end{aligned}$$

Proof. The tighter lower and upper bounds follow immediately from (3.6) in Theorem 3.1. However,

$$(5.8)\qquad \frac{1}{\sqrt{2}} \le f\left(\frac{\|A\| \cdot \|x_k\|}{\|b\|}\right) = \frac{\{\|b\|^2 + \|A\|^2\|x_k\|^2\}^{\frac{1}{2}}}{\|b\| + \|A\| \cdot \|x_k\|} \le 1,$$

since for $\omega \ge 0$, $f(\omega) \equiv (1+\omega^2)^{\frac{1}{2}}/(1+\omega)$ satisfies $f(0) = 1$, $f(\omega) < 1$ for $\omega > 0$, $f(\omega) \to 1$ for $\omega \to \infty$, and $f(\omega)$ has for $\omega > 0$ a single minimum $f(1) = \sqrt{2}/2$. This gives the weaker lower and upper bounds in (5.7). □

Note that the ratio of the tighter upper and lower bounds is (exactly as in (3.6))

$$(5.9)\qquad \nu \equiv \frac{\{\gamma^{-2} + (1-\delta_k^2)^{-1}\|D_k^{-1}y_k\|^2\}^{\frac{1}{2}}}{\{\gamma^{-2} + \|D_k^{-1}y_k\|^2\}^{\frac{1}{2}}}$$

and the corresponding ratio of the weaker bounds is $\sqrt{2}\,\nu$. We will prefer the weaker bounds because they are convenient for the discussion of the particular scalings in the next section, and the factor $\sqrt{2}$ does not affect our considerations.

6. Scaling choices.

There is no easy preference for the choice of scaling, since we have to consider several aspects that are unfortunately in conflict.

As described before, our ultimate goal is to relate the loss of orthogonality among the Arnoldi vectors to the convergence of MGS GMRES measured by the normwise relative backward error by obtaining a scaled version of (1.4). Considering (5.3) it seems that the role of scaling is to minimize $\kappa([v_1\rho_0\gamma, AV_kD_k])$, and the nearly optimal scaling (5.4) seems to be the right choice. Scaling decreasing $\kappa([v_1\rho_0\gamma, AV_kD_k])$ may, however, increase the value of $\delta(\gamma, D_k)$ and therefore act against the tightness of the bounds in Theorem 5.1; see (3.8) and (3.12). While decreasing γ decreases δ_k [22, Corollary 4.1], decreasing entries in D_k increase the upper bounds in (3.8) and potentially also δ_k. In order to describe this in more detail we denote, for the moment, $\vartheta \equiv (\sigma_k(D_k))^{-1}, D_k' \equiv \vartheta D_k, \sigma_k(D_k') = 1$. Now $\vartheta\sigma_1([v_1\rho_0\gamma, AV_kD_k]) = \sigma_1([v_1\rho_0\gamma\vartheta, AV_kD_k'])$, $\kappa([v_1\rho_0\gamma, AV_kD_k]) = \kappa([v_1\rho_0\gamma\vartheta, AV_kD_k'])$, and for δ_k in (3.4)

$$\begin{aligned}
\delta_k \equiv \delta_k(\gamma, D_k) &= \frac{\sigma_{k+1}([v_1\rho_0\gamma, AV_kD_k])}{\sigma_k(AV_kD_k)} \\
(6.1)\qquad &= \frac{\sigma_{k+1}([v_1\rho_0\gamma\vartheta, AV_kD_k'])}{\sigma_k(AV_kD_k')} = \delta_k(\gamma\vartheta, D_k').
\end{aligned}$$

This shows the bounds in Theorem 5.1 rescale trivially, giving the same results for the scaling γ, $D_k = \vartheta^{-1}D_k'$, as for the scaling $\gamma\vartheta$, D_k'. It is clear from [22, Corollary

4.1] that, for a fixed $D_k^{'}$, $\delta_k(\gamma\vartheta, D_k^{'})$ increases monotonically with $\gamma\vartheta$, and in some circumstances it can be close to unity. (We assume that the assumptions of Theorem 3.1 hold and therefore $\delta_k < 1$ always.) It follows that if ϑ is very large (resulting in large $\gamma\vartheta$), then $\delta_k(\gamma\vartheta, D_k^{'})$ can be close to unity. This negatively affects the tightness of the bounds in Theorem 5.1. Consequently, the near optimal tightness in (5.3) might be achieved at the cost of weakening (5.7). Similarly, weakening (5.3) may result in a tighter (5.7).

Please notice that varying ϑ (for a fixed $D_k^{'}$) has, due to (6.1), the same effect on $\delta_k(\gamma, D_k) = \delta_k(\gamma, \vartheta^{-1}D_k^{'}) = \delta_k(\gamma\vartheta, D_k^{'})$ as varying the scaling parameter γ. It therefore need not be considered here.

To study further the effects of scaling, we will discuss three specific cases: no scaling ($\gamma = 1$, $D_k = I$), the nearly optimal column scaling $\gamma = \rho_0^{-1}$, $D_k = \mathrm{diag}(\|Av_j\|^{-1})$, and the norm scaling $\gamma = \|b\|^{-1}$, $D_k = \|A\|^{-1}I$. We will consider only the weaker bounds given by Theorem 5.1.

PROPOSITION 6.1. *Under the conditions and assumptions of Theorem* 3.1 *and using the notation of Theorem* 5.1*, we have the following bounds:*
With no scaling ($\gamma = 1$, $D_k = I$) we have $\delta_k \equiv \delta_k(1, I)$, $\sigma_1 \equiv \sigma_1([r_0, AV_k])$, $\kappa_k \equiv \kappa([r_0, AV_k])$, and the weaker bounds from (5.7) *give*

$$\chi_{L1} \equiv \frac{\sigma_1}{\sqrt{2}} \cdot \frac{\{1 + \|y_k\|^2\}^{\frac{1}{2}}}{\{\|b\|^2 + \|A\|^2\|x_k\|^2\}^{\frac{1}{2}}} \leq \kappa_k \frac{\|r_k\|}{\|b\| + \|A\| \cdot \|x_k\|} \leq \sigma_1 \frac{\{1 + (1-\delta_k^2)^{-1}\|y_k\|^2\}^{\frac{1}{2}}}{\{\|b\|^2 + \|A\|^2\|x_k\|^2\}^{\frac{1}{2}}} \equiv \chi_{U1}. \tag{6.2}$$

The nearly optimal column scaling $\gamma = \rho_0^{-1}$, $D_k = \mathrm{diag}(\|Av_j\|^{-1})$ gives
$\delta_k \equiv \delta_k(\rho_0^{-1}, D_k)$, $\sigma_1 \equiv \sigma_1([v_1, AV_kD_k])$, $\kappa_k \equiv \kappa([v_1, AV_kD_k])$, and

$$\chi_{L2} \equiv \frac{\sigma_1}{\sqrt{2}} \cdot \frac{\{\rho_0^2 + \|D_k^{-1}y_k\|^2\}^{\frac{1}{2}}}{\{\|b\|^2 + \|A\|^2\|x_k\|^2\}^{\frac{1}{2}}} \leq \kappa_k \frac{\|r_k\|}{\|b\| + \|A\| \cdot \|x_k\|} \leq \sigma_1 \frac{\{\rho_0^2 + (1-\delta_k^2)^{-1}\|D_k^{-1}y_k\|^2\}^{\frac{1}{2}}}{\{\|b\|^2 + \|A\|^2\|x_k\|^2\}^{\frac{1}{2}}} \equiv \chi_{U2}. \tag{6.3}$$

Finally, the scaling $\gamma = \|b\|^{-1}$, $D_k = \|A\|^{-1}I$ gives

$$\delta_k \equiv \delta_k\left(\frac{\|A\|}{\|b\|}, I\right), \quad \sigma_1 \equiv \sigma_1\left(\left[\frac{v_1\rho_0}{\|b\|}, \frac{AV_k}{\|A\|}\right]\right), \quad \kappa_k \equiv \kappa\left(\left[\frac{v_1\rho_0}{\|b\|}, \frac{AV_k}{\|A\|}\right]\right), \tag{6.4}$$

$$\chi_{L3} \equiv \frac{\sigma_1}{\sqrt{2}} \cdot \frac{\{\|b\|^2 + \|A\|^2\|y_k\|^2\}^{\frac{1}{2}}}{\{\|b\|^2 + \|A\|^2\|x_k\|^2\}^{\frac{1}{2}}} \leq \kappa_k \frac{\|r_k\|}{\|b\| + \|A\| \cdot \|x_k\|} \leq \sigma_1 \frac{\{\|b\|^2 + (1-\delta_k^2)^{-1}\|A\|^2\|y_k\|^2\}^{\frac{1}{2}}}{\{\|b\|^2 + \|A\|^2\|x_k\|^2\}^{\frac{1}{2}}} \equiv \chi_{U3}. \quad \square \tag{6.5}$$

Throughout this discussion of GMRES in exact arithmetic, we could have replaced $\|y_k\|_2$ by $\|x_k - x_0\|_2$, since $x_k = x_0 + V_ky_k$. However, we chose not to do this in order that the results be relevant to the finite precision case as well, where V_k may lose orthogonality. The exception which will follow will allow us to write the result (6.5) in a very simple form. Consider for the moment $x_0 = 0$. Then (ideally) $\|x_k\| = \|y_k\|$ and when $\delta_k(\|b\|^{-1}\|A\|, I) \ll 1$, (6.5) reduces to (with the definitions in (6.4))

$$\frac{\sigma_1}{\sqrt{2}} \;\le\; \kappa_k \,\frac{\|r_k\|}{\|b\| + \|A\| \cdot \|x_k\|} \;\lesssim\; \sigma_1\,. \tag{6.6}$$

For the scaling in (6.3), each of the $k+1$ columns of $[v_1, AV_kD_k]$ has a 2-norm of 1, so

$$1 \le \sigma_1 \equiv \|[v_1\rho_0\gamma, AV_kD_k]\| \le \sqrt{k+1}\,. \tag{6.7}$$

For the scaling in (6.4) and x_0 chosen from (2.8), the 2-norm of each column of $[v_1\rho_0\|b\|^{-1}, AV_k\|A\|^{-1}]$ is bounded above by 1, so the upper bound in (6.7) holds. If we assume $x_0 = 0$ as well, then $v_1\rho_0 = r_0 = b$, so the first column has the 2-norm of 1, and all of (6.7) holds. However, generalizing a suggestion by Ruiz [26], for any matrix partitioned into two submatrices

$$\begin{aligned}
\max\{\|W\|, \|Z\|\} \;\le\; \|[W, Z]\| &= \max_{\|w\|^2+\|z\|^2=1} \|Ww + Zz\| \\
&\le \max_{\|w\|^2+\|z\|^2=1} \{\|W\|\|w\| + \|Z\|\|z\|\} \\
&= \max_{\|w\|^2+\|z\|^2=1} (\|W\|, \|Z\|)(\|w\|, \|z\|)^T \\
&\le \{\|W\|^2 + \|Z\|^2\}^{\frac{1}{2}}\,.
\end{aligned} \tag{6.8}$$

Applying these bounds to $\|[v_1\rho_0\gamma, AV_kD_k]\|$ with scaling in (6.4) and $x_0 = 0$ gives

$$1 \;\le\; \sigma_1 \equiv \|[v_1\rho_0\gamma, AV_kD_k]\| \;\le\; \sqrt{2}\,. \tag{6.9}$$

Thus for the scaling in (6.4) with $x_0 = 0$, both (6.6) and (6.9) hold, which gives the simplest form of our main result—the correct version of (1.2).

Proposition 6.2. *Under the conditions and assumptions of Theorem* 3.1*, using the notation of Theorem* 5.1 *and assuming that* $\delta_k \ll 1$*, we have with* $\gamma = \|b\|^{-1}$*,* $D_k = \|A\|^{-1}$*, and* $x_0 = 0$*:*

$$\frac{1}{\sqrt{2}} \le \kappa_k \,\frac{\|r_k\|}{\|b\| + \|A\| \cdot \|x_k\|} \;\lesssim\; \sqrt{2}, \tag{6.10}$$

which can be written as

$$\kappa_k \,\frac{\|r_k\|}{\|b\| + \|A\| \cdot \|x_k\|} \;=\; O(1)\,. \qquad \square \tag{6.11}$$

The last results hold even for nonzero x_0 whenever $\|y_k\| = O(\|x_k\|)$. (Numerical experiments suggest that for well chosen x_0 (see (2.8)), this assumption is very realistic.) Because of the simple form of (6.10) we call the scaling in (6.4) scaling for elegance. In the experiments in section 7 we will compare the bounds χ_{L1}, χ_{L2}, and χ_{L3}, and χ_{U1}, χ_{U2}, and χ_{U3}, together with the effects of the particular scalings on the tightness of (5.3).

In an iterative solution of equations with nonsingular A we expect $\|r_k\| \to 0$ so $\delta_k \to 0$ (see (3.8)), and $\delta_k \ll 1$ is necessary eventually. For a general (finite dimensional) problem this seems trivial, but there are extreme possibilities: δ_k may, for example, be close to unity (or $\delta_k = 1$ in some special cases) for $k = 1, 2, \ldots, n-1$ and $\delta_n = 0$ (see section 4). However, in many practical problems there exists a k_0 much smaller than n such that $\delta_k \ll 1$ for $k = k_0, k_0+1, \ldots,$ and in Corollary 3.2

$$0 \le \eta = \eta(k) \approx 0 \tag{6.12}$$

holds for $k > k_0$. In other problems $\delta_k \ll 1$ for a number of steps, but then suddenly δ_k appears very close to unity. In these cases the smoothed upper bound (3.9) might be considered—our experiments suggest it is usually close to $\|r_k\|$ for all iteration steps k. Typical examples are shown in section 7.

Under the assumptions of Theorem 3.1 $\delta_k = \delta_k(\gamma, D_k)$ is bounded away from unity for all positive γ, D_k [21, Theorem 3.1] and, unless the projection of r_0 onto the left singular vector subspace corresponding to $\sigma_{\min}(AV_kD_k)$ of the matrix AV_kD_k is very small compared to $\|r_k\|$, we can expect that the bounds for $\|r_k\|$ given by Theorem 3.1 are sufficiently tight. Still, the choices of D_k having small elements on the diagonal seems very unfortunate because they (potentially) increase the value of δ_k. Fortunately, as shown in section 7, in practical computations small diagonal elements in D_k have a much less dramatic effect on δ_k, and on the tightness of the bounds (3.6) for $\|r_k\|$, than the weaker upper bound in (3.8) would suggest. Moreover, we will show that in our experiments the scaling $\gamma = \|b\|^{-1}$, $D_k = \|A\|^{-1}I$ (which provided the result (6.10)) indeed relaxed the tightness of the bounds (3.6) for $\|r_k\|$, but the resulting relaxed bounds always remained very acceptable.

As mentioned above, under the assumption (3.5) δ_k is bounded away from unity, but it can still get very close to unity for some k. We have observed numerically (see also section 7) that with no scaling ($\gamma = 1$, $D_k = I$), δ_k gets close to unity quite rarely. For the other scalings considered in this paper (which are important for the formulation of our results), some δ_k may be much closer to unity, and that may also happen more often. Still, as we will now show for the example of the scaling for elegance $\gamma = \|b\|^{-1}$, $D_k = \|A\|^{-1}I$, the situation $\delta_k \approx 1$ cannot occur after GMRES has converged to a reasonable accuracy, and therefore it does not represent a serious obstacle for our theory. From (3.8) we have

$$\delta_k \le \gamma\|r_k\|/\sigma_k(AV_kD_k), \tag{6.13}$$

which with the scaling $\gamma = \|b\|^{-1}$, $D_k = \|A\|^{-1}I$, and with $\|r_0\| \le \|b\|$ (perhaps via (2.8)) and $V_k^TV_k = I$, gives a bound in terms of the *relative residual* $\|r_k\|/\|r_0\|$:

$$\delta_k \le \frac{\|r_k\| \cdot \|A\|}{\|b\| \cdot \sigma_k(AV_k)} \le \frac{\|r_k\|}{\|r_0\|}\,\kappa(A). \tag{6.14}$$

Similarly (using the same scaling) with $\|x_k\| = \|y_k\|$ we can obtain a bound in terms of the *relative backward error*

$$\delta_k \le \frac{\|r_k\|}{\|b\| + \|A\| \cdot \|x_k\|}\,\sqrt{2}\,\kappa(A) = \beta(x_k)\sqrt{2}\,\kappa(A). \tag{6.15}$$

This follows using (3.7) and (5.8), since

$$\delta_k \le \frac{\|r_k\|}{\{\|b\|^2 + \|A\|^2\|x_k\|^2\}^{1/2}\sigma_k(AV_k)/\|A\|} \le \beta(x_k)\sqrt{2}\,\kappa(A).$$

Thus, when the relative residual norm drops significantly below $\kappa(A)^{-1}$, or the relative backward error drops significantly below $\{\sqrt{2}\,\kappa(A)\}^{-1}$, $\delta_k = \delta_k(\|b\|^{-1}\|A\|, I) \ll 1$ and (6.11) will hold.

For any given D_k there is a particular value of the scaling parameter γ such that $\delta(\gamma, D_k)$ is related to $\|r_k\|$ even in a more tight way than described above. For a fixed D_k define $\gamma_0^{(k)} = \gamma_0^{(k)}(D_k) = \sigma_k(AV_kD_k)/\rho_0$. With the scaling $D_k, \gamma_0^{(k)}$ the first column of the matrix $[v_1\rho_0\gamma_0^{(k)}, AV_kD_k]$ is equal to $\sigma_k(AV_kD_k)v_1$ and has the norm equal to $\sigma_k(AV_kD_k)$. Moreover, for this D_k, $\delta_k(\gamma, D_k) < 1$ for all $\gamma < \gamma_0^{(k)}$, and

$$\|r_k\| = \rho_0 \quad \text{if and only if} \quad \delta_k(\gamma_0^{(k)}, D_k) = 1\,, \tag{6.16}$$

$$\delta_k(\gamma_0^{(k)}, D_k) \le \|r_k\|/\rho_0 \le \sqrt{2}\,\delta_k(\gamma_0^{(k)}, D_k); \tag{6.17}$$

see [15, (3.11) and (3.12)]. Though with this particular scaling the relationship between $\delta_k(\gamma_0^{(k)}, D_k)$ and $\|r_k\|$ is extremely simple, it will not lead to a simple form of the main result (1.2). Also, possibly small value $\gamma_0^{(k)}$ may inconveniently relax the bound (5.3) and significantly complicate the analysis left to [17]. Therefore we have not used this scaling in our paper.

The approach here might also be useful for Krylov subspace methods which minimize other norms, such as minimum error methods, as we now show. Let $V_k = [v_1, \ldots, v_k]$ be generated in some way, and $r_0 = b - Ax_0$, $\rho_0 = \|r_0\|$, $v_1 = r_0/\rho_0$, $x_k = x_0 + V_ky_k$, $r_k = b - Ax_k = r_0 - AV_ky_k$, with A nonsingular. Consider, for example, a method that minimizes $\|A^{-1}r_k\| = \|x - x_k\|$ at each step (so y_k will differ from that in GMRES). Then taking $[c, B] = A^{-1}[v_1\rho_0, AV_k] = [(x - x_0), V_k]$ and $\gamma = \rho_0^{-1}$, Theorem 4.1 of [21] gives (with $\delta_k = \sigma_{k+1}([(x - x_0)\rho_0^{-1}, V_k])/\sigma_k(V_k)$) the bounds

$$\begin{aligned}\sigma_{k+1}([(x - x_0)\rho_0^{-1}, V_k])\,\{\rho_0^2 + \|y_k\|^2\}^{\frac{1}{2}} \quad &\le \quad \|x - x_k\| \\ \le \sigma_{k+1}([(x - x_0)\rho_0^{-1}, V_k])\,\{\rho_0^2 + (1 - \delta_k^2)^{-1}&\|y_k\|^2\}^{\frac{1}{2}},\end{aligned} \tag{6.18}$$

so at least this theory holds for more general minimum norm methods than just GMRES. Of course, if $V_k^TV_k = I$, then $\sigma_k(V_k) = 1$. We have not studied how this might be used.

It appears that the approach can also be applied to methods which minimize some norm with respect to other Krylov subspaces, such as LSQR [20, 19] for solution of equations with unsymmetric A, or LS solutions with rectangular A. It may also be useful for methods which are not based on Krylov subspaces.

7. Experimental results. We will illustrate our theoretical results with numerical experiments. We initiated these to look for possible limitations in our theory. We wished to check the validity of our assumptions in practical computations. We also wished to find out to what extent the results developed here for exact precision GMRES would hold for quantities computed in the presence of rounding errors.

In our theory $\delta_k = \sigma_{k+1}([v_1\rho_0\gamma, AV_kD_k])/\sigma_k(AV_kD_k)$ plays an important role. Section 4 showed it is possible to have $\delta_k = 1$ for all but the last step, and in that example the residual stagnated at $\|r_0\|$ until the final step. On the other hand, $\delta_k \approx 1$ cannot in general (for scalings different from D_k, $\gamma_0^{(k)}(D_k)$, see (6.16), (6.17)) be linked with the (approximate) stagnation of GMRES; the GMRES residual norm may almost stagnate while $\delta_k \ll 1$, and it can decrease rapidly while $\delta_k \approx 1$. If $\delta_k \approx 1$, then there can be a large gap between the upper and lower bounds in (3.6). This does not negate the argument that orthogonality is effectively maintained until convergence in finite precision MGS GMRES ($\delta_k \ll 1$ is necessary eventually), but it does make us question the tightness of the bounds in (3.6).

Fortunately, experiments suggest that (3.9) is always a sufficiently good (and mostly very good) upper bound. We also found that with no scaling ($\gamma = 1$, $D_k = I$) δ_k is often reasonably below unity during the entire computation. As k increases δ_k can decrease, then increase, but it must eventually become small, for from (3.7), (3.8), (6.14), and (6.15) we see the upper bound on δ_k must decrease as $\|r_k\|$ or $\|r_k\|/(\|b\| + \|A\| \cdot \|x_k\|)$ becomes sufficiently small. However, when $\delta_k \ll 1$ from the start to the end,

$$\|r_k\| \;\approx\; \sigma_{k+1}([v_1\rho_0\gamma, AV_kD_k])\,\{\gamma^{-2} + \|D_k^{-1}y_k\|^2\}^{\frac{1}{2}}$$

throughout the computation, and the lower and upper bounds are very close. Thus we can have this unexpectedly very close relationship between $\|r_k\|$ and the smallest singular value of $[v_1\rho_0\gamma, AV_kD_k]$. An interesting experience was that even when $\delta_k \approx 1$, leading to the upper bound being significantly larger than the lower bound in (3.6), it was not always the upper bound which was weak. We frequently observed that the upper bound was tight while the lower bound was a noticeable underestimate for $\|r_k\|$. Moreover, the dependence of δ_k and the tightness of the bounds (3.6) on the scaling parameters γ, D_k was quite weak. We will illustrate these observations by presenting results of numerical experiments showing different types of behavior of δ_k. These observations could also be further studied theoretically using the results of Proposition 6.1 or some other approach, but we do not wish to go into it here.

In all experiments $b = e \equiv (1, \ldots, 1)^T$. Except for the experiment shown in Figure 7.10 (where $x_0 = \text{randn}(n, 1)$ from MATLAB 5.3), x_0 is always determined from (2.8) with $x_p = \text{randn}(n, 1)$ from MATLAB 5.3. These choices of x_0 and x_p are worth a comment. We wish to illustrate our theoretical results on some nontrivial examples. The randomly generated initial vectors x_0 (or x_p in (2.8)) were chosen in our illustrations to avoid any correlation between the initial approximation and the solution. This is because we sought to illustrate cases where there were no hidden relationship that affected the computations. In practical computations, however, for the very same reason, a randomly chosen initial approximation should be avoided. Sometimes a random initial approximation x_0 is reported to give faster convergence than the other popular choice $x_0 = 0$. As we explain later, we believe that statements like that represent a serious misunderstanding caused by a superficial view of convergence. As far as genuine convergence characteristics are concerned, we argue that such statements are of no relevance.

Experiments were performed on a Silicon Graphics Origin 200 Workstation using MATLAB 5.3, $\epsilon = 1.11 \times 10^{-16}$. In all experiments matrices from the Rutherford–Boeing collection were used. Results for the matrix FS1836 with $n = 183$, $\|A\| \approx 1.2{*}10^9$, $\kappa(A) \approx 1.5{*}10^{11}$ (see Figures 7.1–7.4) illustrate improvement of the tightness of the bounds (3.6) as the residual norm drops. For the matrix WEST0132 with $n = 132$, $\|A\| \approx 3.2 * 10^5$, $\kappa(A) \approx 6.4 * 10^{11}$ (see Figures 7.5–7.8), the tightness of the bounds (3.6) oscillates during the whole computation. Results for the matrix STEAM1 with $n = 240$, $\|A\| \approx 2.2 * 10^7$, $\kappa(A) \approx 3.1 * 10^7$ (see Figures 7.9 and 7.10) represent the case when the bounds (3.6) are very tight from the start to the end.

We have given two figures for STEAM1 and four figures for the other matrices, so we now indicate what is in each figure. Figures 7.1, 7.5, 7.9, and 7.10 make the same use of lines. The dots show the norm of the *directly* computed residual divided

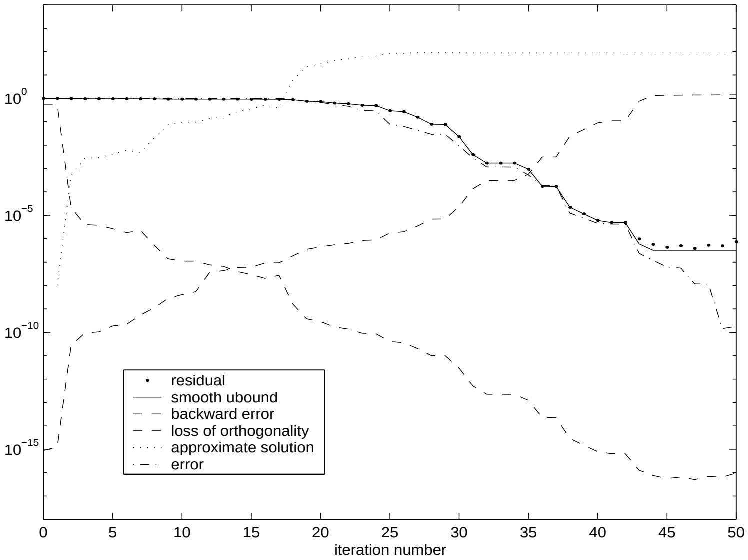


FIG. 7.1. *Norm of the directly computed relative residual (dots), the smooth upper bound (solid line), the loss of orthogonality among the Arnoldi vectors measured in the Frobenius norm (dashed line, monotonically increasing) and the normwise relative backward error (dashed line, mostly decreasing), norm of the approximate solution (dotted line), and the relative error (dashed-dotted line) for MGS GMRES applied to FS1836.*

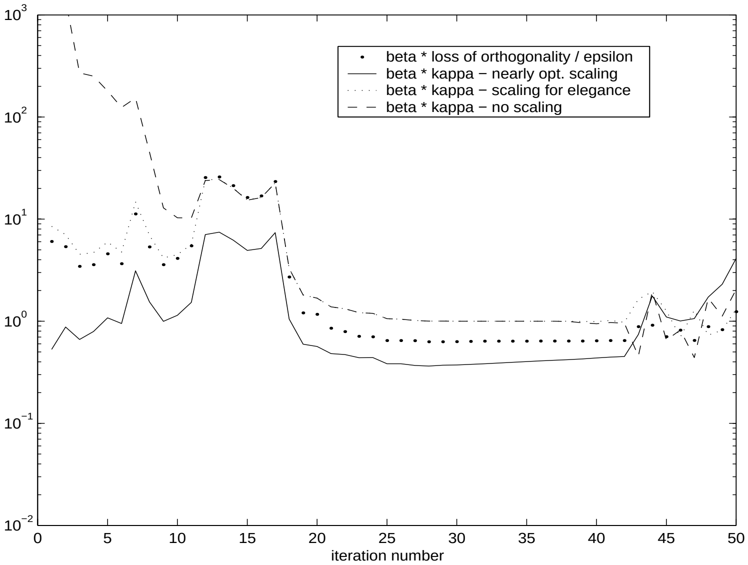


FIG. 7.2. *The product of the normwise relative backward error and the loss of orthogonality among the Arnoldi vectors measured in the Frobenius norm divided by the machine precision unit ϵ (dots), and the product of the normwise relative backward error $\beta(x_k)$ and the condition number of the matrix $[v_1\rho_0\gamma, AV_kD_k]$ for different scalings: the nearly optimal column scaling $\gamma = \rho_0^{-1}$, $D_k = \mathrm{diag}(\|Av_j\|^{-1})$ (solid line), the norm scaling (scaling for elegance) $\gamma = \|b\|^{-1}$, $D_k = \|A\|^{-1}I$ (dotted line), and no scaling $\gamma = 1$, $D_k = I$ (dashed line) for MGS GMRES applied to FS1836.*

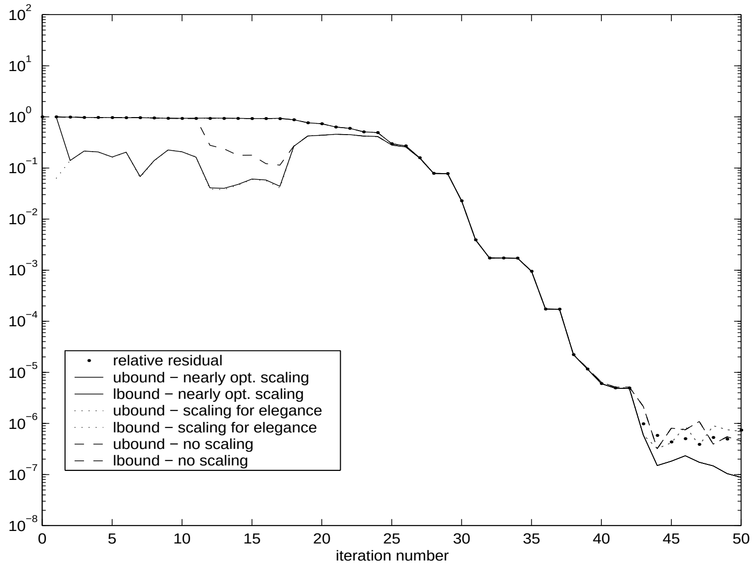


FIG. 7.3. *Norm of the directly computed relative residual (dots), and its lower and upper bounds μ_L and μ_U for different scalings: the nearly optimal column scaling (solid lines), the scaling for elegance (dotted lines), and no scaling (dashed lines) for MGS GMRES applied to FS1836. Until the orthogonality is completely lost, the upper bounds are indistinguishable from the actual quantities.*

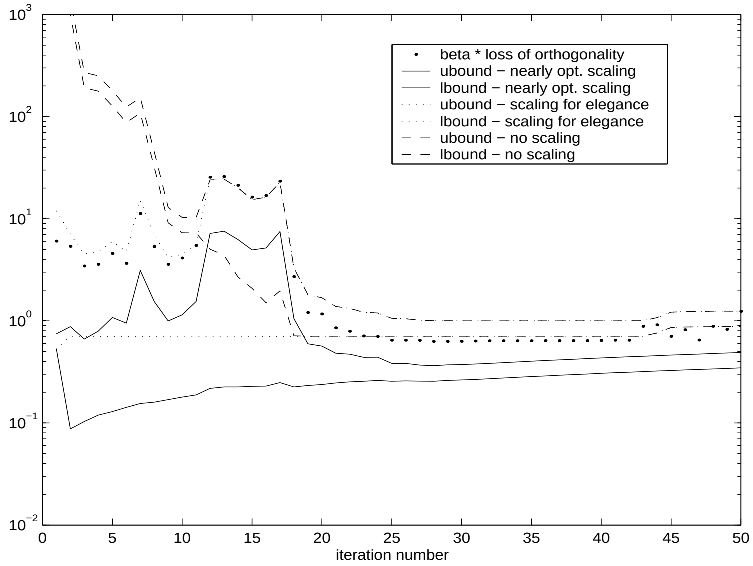


FIG. 7.4. *Product of the backward error and the loss of orthogonality among the Arnoldi vectors measured in the Frobenius norm divided by the machine precision unit ϵ (dots), and the values χ_L and χ_U for different scalings: the nearly optimal column scaling (solid lines), the scaling for elegance (dotted lines), and no scaling (dashed lines) for MGS GMRES applied to FS1836.*

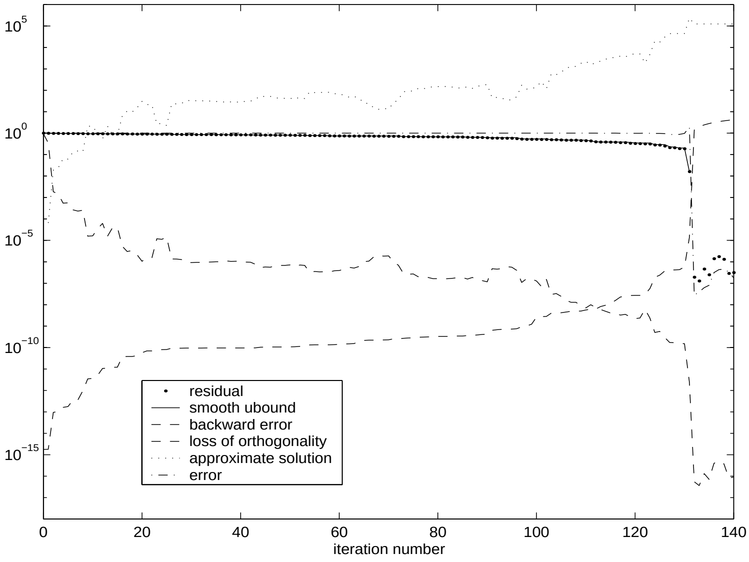


FIG. 7.5. *Norm of the directly computed relative residual (dots), the smooth upper bound (solid line), the loss of orthogonality among the Arnoldi vectors measured in the Frobenius norm (dashed line, monotonically increasing) and the normwise relative backward error (dashed line, mostly decreasing), norm of the approximate solution (dotted line), and the relative error (dashed-dotted line) for MGS GMRES applied to WEST0132.*

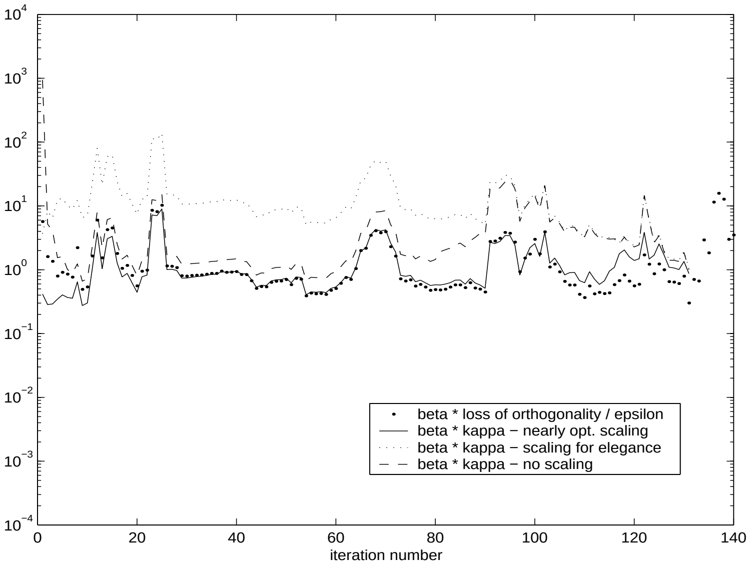


FIG. 7.6. *Product of the normwise relative backward error and the loss of orthogonality among the Arnoldi vectors measured in the Frobenius norm divided by the machine precision unit (dots), and product of the backward error and the condition number of the matrix* $[v_1\rho_0\gamma, AV_kD_k]$ *for different scalings: the nearly optimal column scaling (solid line), the scaling for elegance (dotted line), and no scaling (dashed line) for MGS GMRES applied to WEST0132.*

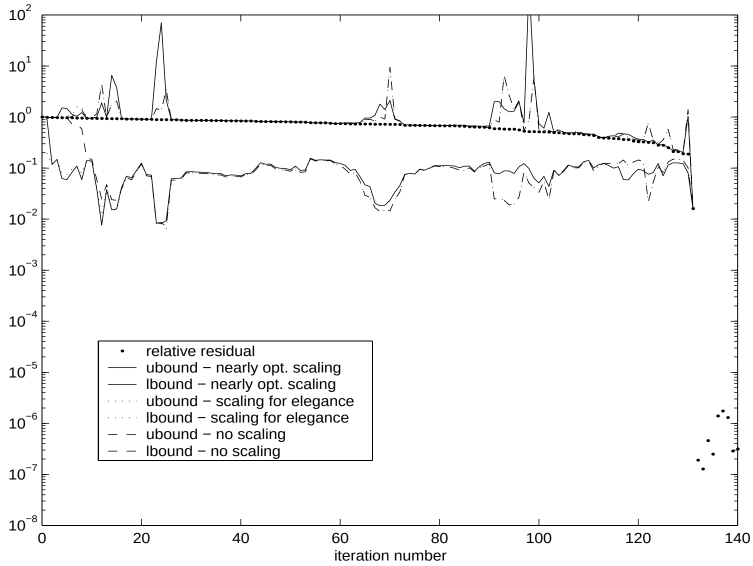


Fig. 7.7. *Norm of the directly computed relative residual(dots), and its lower and upper bounds* μ_L *and* μ_U *for different scalings: the nearly optimal column scaling (solid lines), the scaling for elegance (dotted lines), and no scaling (dashed lines) for MGS GMRES applied to WEST0132.*

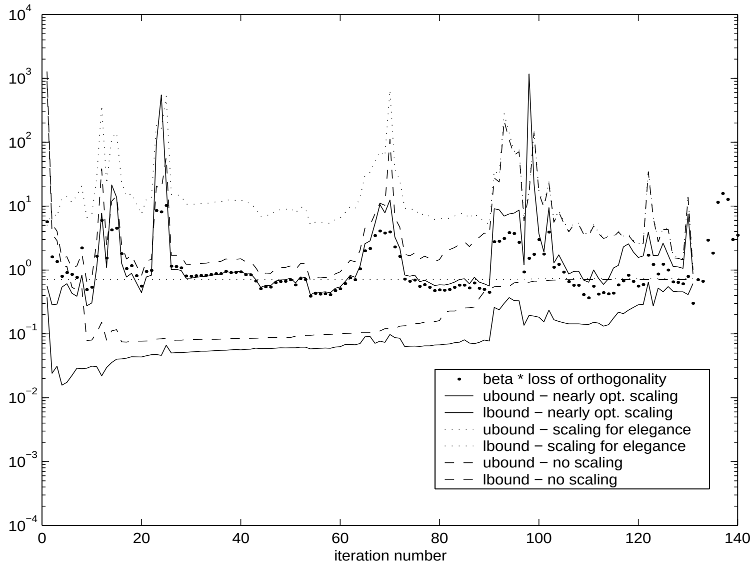


Fig. 7.8. *Product of the backward error and the loss of orthogonality among the Arnoldi vectors measured in the Frobenius norm divided by the machine precision unit* ϵ *(dots), and the values* χ_L *and* χ_U *for different scalings: the nearly optimal column scaling (solid lines), the scaling for elegance (dotted lines), and no scaling (dashed lines) for MGS GMRES applied to WEST0132.*

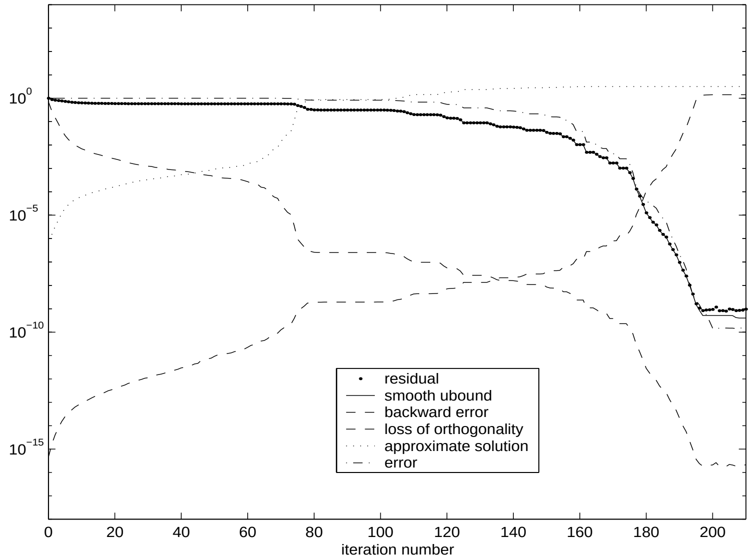


FIG. 7.9. *Norm of the directly computed relative residual (dots), the smooth upper bound (solid line), the loss of orthogonality among the Arnoldi vectors measured in the Frobenius norm (dashed line, monotonically increasing) and the normwise relative backward error (dashed line, mostly decreasing), norm of the approximate solution (dotted line), and the relative error (dashed-dotted line) for MGS GMRES applied to STEAM240.*

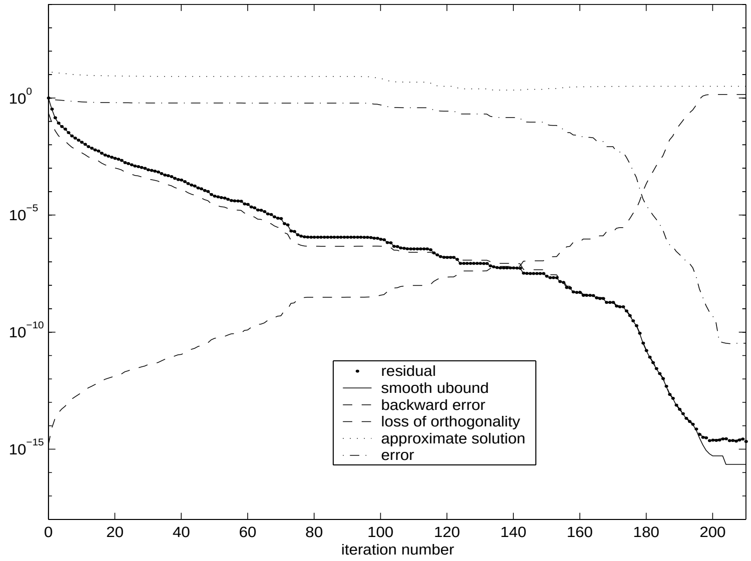


FIG. 7.10. *Norm of the directly computed relative residual (dots), the loss of orthogonality among the Arnoldi vectors measured in the Frobenius norm (dashed line, monotonically increasing) and the normwise relative backward error (dashed line, mostly decreasing), norm of the approximate solution (dotted line), and the relative error (dashed-dotted line) for MGS GMRES applied to STEAM240 with randomly chosen initial approximation x_0.*

by $\|r_0\|$, that is, $\|b - Ax_k\|/\|r_0\|$, which we call the relative residual. (We do not give the *iteratively* computed residual norm (2.7); until near convergence, it was always graphically indistinguishable from the norm of the directly computed residual.) The solid line gives the smoothed upper bound (3.9) divided by $\|r_0\|$. The dashed-dotted line gives the normalized norm of the error $\|x-x_k\|/\|x-x_0\|$; the dotted line gives the norm of the approximate solution $\|x_k\|$. The dashed lines give the loss of orthogonality among the Arnoldi vectors measured in the Frobenius norm $\|I - V_k^T V_k\|_F$ (essentially increasing), as well as the normwise relative backward error $\|r_k\|/(\|b\| + \|A\| \cdot \|x_k\|)$, which is mostly decreasing. Note the spectacular symmetry of the loss of orthogonality and the backward error in every case.

For each matrix, the remaining figures present and compare convergence characteristics, upper and lower bounds, and several quantities illustrating our theory for different scalings of the matrix $[v_1\rho_0\gamma, AV_kD_k]$. In each of Figures 7.2, 7.3, 7.4 (for FS1836) and 7.6, 7.7, 7.8 (for WEST0132) dashed lines represent results with no scaling $\gamma = 1$, $D_k = I$, solid lines the nearly optimal column scaling $\gamma = \rho_0^{-1}$, $D_k = \mathrm{diag}(\|Av_j\|^{-1})$, and dotted lines the scaling for elegance $\gamma = \|b\|^{-1}$, $D_k = \|A\|^{-1}I$.

Figures 7.2 and 7.6 are devoted to the tightness of the bound (5.3) for the loss of orthogonality among the Arnoldi vectors. The dots show the product of the normwise relative backward error and the loss of orthogonality divided by the machine precision unit $\{\|r_k\|/(\|b\| + \|A\| \cdot \|x_k\|)\} \cdot \|I - V_k^T V_k\|_F \,/\, \epsilon$, the dashed, solid, and dotted lines the product $\{\|r_k\|/(\|b\| + \|A\| \cdot \|x_k\|)\} \cdot \kappa([v_1\rho_0\gamma, AV_kD_k])$ for different scalings. The figures show that (5.5) is well justified for the nearly optimal column scaling. Replacing the actual loss of orthogonality $\|I - V_k^T V_k\|_F$ in our considerations by $\{\kappa([v_1\rho_0\gamma, AV_kD_k])\,\epsilon\}$ does not cause a significant difference (except perhaps at the beginning of the process with no scaling) even for the other scalings. Close to convergence (5.5) holds for all the scalings considered in our paper.

Figures 7.3 and 7.7 are devoted to normalized residual bounds, that is, bounds on $\|b - Ax_k\|/\|r_0\|$, which are denoted by points. The pairs of dashed, solid, and dotted lines give the upper and lower bounds μ_U and μ_L from (3.6) for different scalings. We can see that the effect of scaling on the bounds in (3.6) is quite insignificant.

Finally, Figures 7.4 and 7.8 compare the product of the normwise relative backward error and the loss of orthogonality divided by the machine precision unit, that is, $\{\|r_k\|/(\|b\| + \|A\| \cdot \|x_k\|)\} \cdot \|I - V_k^T V_k\|_F \,/\, \epsilon$ (denoted by dots), with the upper and lower bounds χ_U and χ_L from (6.2) (dashed lines), (6.3) (solid lines), and (6.5) (dotted lines). These figures reflect the possible lack of tightness of the bound for the loss of orthogonality among the Arnoldi vectors shown separately on Figures 7.2 and 7.6, as well as the lack of tightness of the bounds in (6.2)–(6.5). They demonstrate that though the results developed in this paper assume exact arithmetic, and though the form of the bounds in (6.2)–(6.5) seems a bit complicated, the simplest form of our main result (6.11) holds as convergence is approached for all our scalings and for the quantities actually computed using finite precision arithmetic.

The experiments for the figures discussed up to now (and for Figure 7.9) used $b = e$ and x_0 determined from (2.8) with $x_p = \mathrm{randn}(n, 1)$ from MATLAB 5.3. The remaining Figure 7.10 was computed for $x_0 = \mathrm{randn}(n, 1)$ from MATLAB 5.3 without using (2.8). Both Figures 7.9 and 7.10 were computed for the matrix STEAM1, and they show the same quantities as Figures 7.1 and 7.5. If we concentrate on the relative residual norm only, then it looks as if Figure 7.10 shows much better convergence (faster, and to much better accuracy) than Figure 7.9. Such a view on convergence, though understandable, is completely wrong. We cannot give a full quantitative explanation within this paper; however, we will present an intuitive but clear argument on which such an explanation will eventually be based. By using $x_0 = \text{randn}(n, 1)$ and then computing the initial residual as $r_0 = b - Ax_0 = e - Ax_0$ (as on Figure 7.10) we correlate the initial residual strongly with the dominating parts of the operator A. (Note that all the matrices used here have some dominating components.) In all cases the norm of the resulting initial residual is large, $\|r_0\| \gg \|b\|$. At the early stage of computation this artificially created dominating information is eliminated, which creates an illusion of fast convergence. However, no real fast convergence is taking place, as you can see on the error convergence curve, and the "good final accuracy" is due to the fact that the initial residual is large. For $b = e$ and x_0 determined from (2.8) with $x_p = \text{randn}(n, 1)$ from MATLAB 5.3 we get $\|x_0\| \ll 1$ and $x_0 \approx 0$. Then r_0 contains practically no information about the dominating parts of A, the problem is difficult to solve, and the convergence is (for many steps) slow. Still, this choice (which produces results very close to those with the choice $x_0 = 0$) gives the right information about the behavior of GMRES when applied to the problem $Ax = b$, $b = e$. The illusion of fast convergence and better final accuracy for a random x_0 has evolved among some users of numerical software perhaps as a side effect of using the norm of the relative residual for displaying convergence. Our point is that the illusive role of a random x_0 can easily be revealed by using the absolute values of the residual norm for displaying convergence and by comparing the convergence curve for a random x_0 to that for the initial approximation set to zero ($x_0 = 0$). Finally, please note the correspondence of the error and the backward error when comparing Figures 7.10 and 7.9.

Now we comment on particular characteristics of each problem. For the matrix FS1836 in Figures 7.1–7.4 the value of δ_k rises until it is close to unity, stays there for a few iteration steps, then follows the descent of the residual norm. For all scalings the upper bounds μ_U (for $\|r_k\|$) are very tight until convergence, the lower bounds μ_L (for $\|r_k\|$) are weak when $\delta_k \approx 1$, but no scaling ($\gamma = 1, D_k = I$) gives a significantly tighter lower bound than the other two at the early stages of the computation (Figure 7.3). On the other hand, at the early stages of the computation the condition number of the matrix $[v_1\rho_0, AV_k]$ is for no scaling much larger than for the other scalings, which explains the difference between the dashed and the other lines on Figures 7.2 and 7.4 for k from 1 to 10. After convergence is approached, all scalings produce about the same results.

For the matrix WEST0132 (in Figures 7.5–7.8) the value of δ_k is close to unity (with some oscillations) for most iteration steps. The upper and lower bounds μ_U and μ_L differ significantly until the sharp drop of the residual. Scalings are not important. Note that despite the oscillations (we have chosen this matrix on purpose because it seems to produce challenging results; many other examples not presented here give much smoother behavior) all the lines on Figures 7.6 and 7.8 converge together as the sharp drop of the residual is approached.

For the matrix STEAM1 we omit figures analogous to Figures 7.2–7.4 for FS1836 and Figures 7.6–7.8 for WEST0132. The omitted figures would show a good agreement of the computed results with our theory; they do not offer any other information, and therefore we see no reason for extending the length of the paper by including them.

Summarizing, our experiments suggest that the equivalents of Theorems 3.1 and 5.1, of the Propositions 6.1 and 6.2 (where κ_k will be replaced by $\|I - V_{k+1}^T V_{k+1}\|_F$ and $O(1)$ by $O(\epsilon)$), hold for the *numerically computed quantities*. However, the statements must be slightly modified to account for the effect of rounding errors, especially for the influence of the loss of orthogonality on the size of the directly computed residuals $\|b - Ax_k\|$. A rigorous proof will require further work and is intended in [17].

8. Conclusion. In Krylov subspace methods, approximate solutions to matrix problems are usually constructed by using orthogonality relations and projections. Orthogonality and projections create a mathematical elegance and beauty in this context. In the presence of rounding errors orthogonality and projection properties are gradually (and sometimes very quickly) lost. Fortunately, as was first shown for A symmetric and the Lanczos method (see, for example, [16], [10], [12]), not all the mathematical elegance need be lost with them.

This paper is devoted to GMRES, and our fundamental hypothesis is as follows. When the Arnoldi vectors are computed via the finite precision MGS process, the loss of orthogonality is related in a straightforward way to the convergence of GMRES. In particular, orthogonality among the Arnoldi vectors is effectively maintained until the normwise relative backward error converges close to the machine precision level. If we assume that the bound for the loss of orthogonality among the Arnoldi vectors is tight and (5.5) holds, then our hypothesis could be strengthened to the following: *the product of the loss of orthogonality among the Arnoldi vectors (measured in the Frobenius norm) and the normwise relative backward error is for any iteration step a small multiple of the machine precision unit.* This last statement would then imply that total loss of orthogonality among the Arnoldi vectors computed via finite precision MGS orthogonalization would mean convergence of the normwise relative backward error to machine precision level, and, consequently, it would prove backward stability of MGS GMRES. Our work can also be seen as another step on the way (probably started by Sheffield (see [6], especially the abstract and section 2, also [7])) towards the full justification of the MGS orthogonalization in competition with orthogonalization by Householder reflections for certain classes of problems.

Note that in the present paper we have not proven the finite precision versions of the statements formulated above. Our paper assumes exact arithmetic in its theoretical part and carries out groundwork for the detailed rounding error analysis of the MGS GMRES which we plan to publish in [17].

Acknowledgments. The authors would like to thank Miro Rozložník for his helpful comments. They also wish to thank Jőrg Liesen and Daniel Ruiz for many valuable suggestions which improved the content and presentation of this paper.

REFERENCES

[1] M. Arioli, *Stopping criterion for the conjugate gradient algorithm in a finite element method framework*, Numer. Math., submitted.

[2] M. Arioli, I. Duff, and D. Ruiz, *Stopping criteria for iterative solvers*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 138–144.

[3] M. Arioli and C. Fassino, *Roundoff error analysis of algorithms based on Krylov subspace methods*, BIT, 36 (1996), pp. 189–206.

[4] W. Arnoldi, *The principle of minimized iterations in the solution of the matrix eigenvalue problem*, Quart. Appl. Math., 9 (1951), pp. 17–29.

[5] Å. Björck, *Solving linear least squares problems by Gram-Schmidt orthogonalization*, BIT, 7 (1967), pp. 1–21.

[6] Å. Björck and C. C. Paige, *Loss and recapture of orthogonality in the modified Gram–Schmidt algorithm*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 176–190.

[7] J. Drkošová, A. Greenbaum, M. Rozložník, and Z. Strakoš, *Numerical stability of the GMRES method*, BIT, 35 (1995), pp. 308–330.

[8] R. W. Freund and N. M. Nachtigal, *QMR: A quasi-minimal residual method for non-Hermitian linear systems*, Numer. Math., 60 (1991), pp. 315–339.

[9] R. W. Freund and N. M. Nachtigal, *An implementation of the QMR method based on coupled two-term recurrences*, SIAM J. Sci. Comput., 15 (1994), pp. 313–337.

[10] A. Greenbaum, *Behavior of slightly perturbed Lanczos and conjugate gradient recurrences*, Linear Algebra Appl., 113 (1989), pp. 7–63.

[11] A. Greenbaum, M. Rozložník, and Z. Strakoš, *Numerical behavior of the modified Gram-Schmidt GMRES implementation*, BIT, 37 (1997), pp. 706–719.

[12] A. Greenbaum and Z. Strakos, *Predicting the behavior of finite precision Lanczos and conjugate gradient computations*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 121–137.

[13] C. Hegedüs, *private communication*, 1998.

[14] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia, PA, 1996.

[15] J. Liesen, M. Rozložník, and Z. Strakoš, *Least squares residuals and minimal residual methods*, SIAM J. Sci. Comput., 23 (2002), pp. 1503–1525.

[16] C. C. Paige, *Accuracy and effectiveness of the Lanczos algorithm for the symmetric eigenproblem*, Linear Algebra Appl., 34 (1980), pp. 235–258.

[17] C. C. Paige, M. Rozložník, and Z. Strakoš, *Rounding error analysis of the modified Gram-Schmidt GMRES*, in preparation.

[18] C. C. Paige and M. A. Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617–629.

[19] C. C. Paige and M. A. Saunders, *Algorithm* 583 *LSQR: Sparse linear equations and least squares problems*, ACM Trans. Math. Software, 8 (1982), pp. 195–209.

[20] C. C. Paige and M. A. Saunders, *LSQR: An algorithm for sparse linear equations and sparse least squares*, ACM Trans. Math. Software, 8 (1982), pp. 43–71.

[21] C. C. Paige and Z. Strakoš, *Bounds for the least squares distance using scaled total least squares*, Numer. Math., to appear.

[22] C. C. Paige and Z. Strakoš, *Scaled total least squares fundamentals*, Numer. Math., to appear.

[23] M. Rigal and J. Gaches, *On the compatibility of a given solution with the data of a given system*, J. Assoc. Comput. Mach., 14 (1967), pp. 543–548.

[24] M. Rozložník, *Numerical Stability of the GMRES Method*, Ph.D. Thesis, Institute of Computer Science, Academy of Sciences, Prague, 1997.

[25] M. Rozložník and Z. Strakoš, *Variants of the residual minimizing Krylov space methods*, in Proceedings of the XIth Summer School "Software and Algorithms of Numerical Mathematics", I. Marek, ed., Železná Ruda, University of West Bohemia, Plzen, Czech Republic, 1996, pp. 208–225.

[26] D. Ruiz, *private communication*, 2001.

[27] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[28] Z. Strakoš, *Theory of Convergence and Effects of Finite Precision Arithmetic in Krylov Subspace Methods*, D.Sc. Thesis, Institute of Computer Science, Academy of Sciences, Prague, 2001.

[29] A. van der Sluis, *Condition numbers and equilibration matrices*, Numer. Math., 14 (1969), pp. 14–23.

[30] H. A. van der Vorst, *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 631–644.

[31] H. F. Walker, *Implementation of the GMRES method using Householder transformations*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 152–163.

[32] H. F. Walker, *Implementation of the GMRES method*, J. Comput. Phys., 53 (1989), pp. 311–320.

[33] D. Watkins, *Fundamentals of Matrix Computations*, John Wiley, New York, 1991.

SIAM J. SCI. COMPUT.
Vol. 23, No. 6, pp. 1924–1942

ADAPTIVE REDUCED-ORDER CONTROLLERS FOR A THERMAL FLOW SYSTEM USING PROPER ORTHOGONAL DECOMPOSITION[*]

S. S. RAVINDRAN[†]

Abstract. An adaptive reduced-order controller design is presented for flow control using proper orthogonal decomposition (POD). In reduced-order controller design, the idea is to start with an ensemble of data obtained from numerical simulation of the underlying partial differential equations (PDEs). POD is then used to obtain a reduced set of basis functions which is then used to derive a reduced-order model of the PDEs via Galerkin projection. This reduced-order model allows us to derive a reduced-order controller. However, it is not clear, a priori, what is the best way to obtain an ensemble of data that would give basis functions that represent the influence of the control action on the system. In this paper we explore an adaptive procedure for reduced-order controller design that improves the reduced-order model by successively updating the ensemble of data during the optimization iterations. We illustrate this method on a control problem in thermal flow system modeled by a thermally coupled Navier–Stokes equations. Numerical results are presented for a vorticity regulation problem in fluid flows using boundary temperature as control mechanism. Through our numerical experiments we demonstrate the feasibility and applicability of the adaptive reduced-order controllers.

Key words. reduced-order adaptive control, proper orthogonal decomposition, temperature control for fluids

AMS subject classifications. 65C20, 65M60, 93A15, 93C20, 76D05, 80A20

PII. S1064827500374716

1. Introduction.

The reduced-order controller design for physical process modeled by partial differential equations (PDEs) has attracted much attention in recent years due to its ability to reduce computational cost. In [21, 20], we designed a reduced-order controller for a flow system using a procedure called proper orthogonal decomposition (POD).

POD is a model reduction technique for complex nonlinear problems. It was first proposed by Karhunen [13] and Loeve [15] independently and sometimes called Karhunen–Loeve (K–L) expansion. Subsequently it has been applied in various applications. In [16] the method was first called POD and there it was used to study turbulent flows. In [23] other important progress was made and the method of snapshots was incorporated into the POD framework, which will be described in what follows. Other applications in turbulent flow simulations are given in [3, 4, 14, 17, 18, 24].

The essential idea is to generate an optimal basis for the representation of an ensemble of data obtained from numerical computations. One of the features of POD is that it yields a basis that is optimal in the sense that the information contained in an N-ordered POD basis expansion is greater than any other N-ordered basis expansion. Thus, when the PDEs are projected onto this basis using a Galerkin projection, one obtains a reduced-order model.

Reduced-order models allow us to derive reduced-order controllers. However, we need to choose the ensemble of data that represent the system behavior for various

[*] Received by the editors July 6, 2000; accepted for publication (in revised form) September 27, 2001; published electronically February 20, 2002.

http://www.siam.org/journals/sisc/23-6/37471.html

[†] Department of Mathematical Sciences, University of Alabama in Huntsville, Huntsville, AL 35899 (ravindra@ultra.math.uah.edu, http://www.math.uah.edu/ravindra).

trajectories of the control during the iterative optimization procedure. To meet this requirement, we propose in this paper an adaptive procedure that successively updates the ensemble of data and the reduced-order model during the optimization process using the sequential quadratic programming (SQP) method. The SQP method is preferred over the unconstrained optimization methods due to its superior convergence properties, among others. Its application to solve optimal control of Navier–Stokes flow has been demonstrated in [9, 10] and [19].

In this adaptive procedure, we begin with an initial estimate $(\mathbf{Y}_k, \boldsymbol{\zeta}_k)$ and the corresponding initial estimate for the control c_k for the SQP method and generate the corresponding ensemble of snapshots from the Navier–Stokes model. With these snapshots, we find a POD subspace and the corresponding reduced-order model. This reduced-order model is now used in the SQP algorithm to find the new iterate $(\mathbf{Y}_{k+1}, \boldsymbol{\zeta}_{k+1})$. This process of updating the reduced-order model is continued in each of the SQP iterations until convergence is achieved. The approach presented here bears similarities to the approach presented in [2], wherein using the POD model as a "surrogate" within the trust region framework is discussed.

For our physical application, we consider control of flow separation/recirculation in thermal flows. Thermal flow is both challenging and interesting due to the coupling of the fluid flow and the energy transport. It plays an important role in thermal insulation, cooling of fluids in channels surrounding nuclear reactor core, the circulation of liquid metals in solidifying ingot, and the manufacture of crystals. Also, thermal convection is often used as a controlling mechanism in many naturally occurring processes. In light of this, several treatments of control problems in thermal systems can be found in the literature. For instance, [8] studied optimal control of temperature peaks along the bounding surfaces of containers of fluid flows. In [5], a feedback control problem in thermal fluid was studied, and in [12] optimal control of flow separation in channel flow using temperature control was discussed. In [26], an optimal control problem was solved for the convection-diffusion equations with the goal of achieving uniform maximum flux to substrate. Our goal here is to apply POD to an optimal control problem in a backward-facing step channel flow. Backward-facing step flow serves as a prototype for unsteady separated flow. For high Reynolds numbers, flow separates near the corner of the step and recirculation appears downstream of the step. Such recirculation regions will strongly influence heat and mass transfer [1]. We will formulate and numerically solve a recirculation control problem in this configuration with the control action achieved through boundary temperature on a part of the boundary.

An outline of the POD and the reduced-order model derivation are presented in section 2. In section 3, we describe the system and governing equations with relevant boundary conditions; this section also describes the discretization schemes and solution techniques used to solve these governing equations. Section 4 states the model control problem and describes the proposed adaptive reduced-order control approach in the context of solving this model problem. In section 5, we present computational results showing the effectiveness of the present adaptive approach. Finally, we conclude the paper in section 6.

2. POD and model reduction.

In this section, we outline the POD for model reduction of a nonlinear uncontrolled dynamical system of the form

$$\dot{\mathbf{y}} = \mathcal{E}(\mathbf{y}), \qquad \mathbf{y} \in \mathbb{R}n. \tag{2.1}$$

The idea is to replace the given dynamics by an associated dynamics on an M-dimensional subspace $\mathbf{V}^{\mathrm{POD}} \subset \mathbb{R}n$ of the state space. This procedure uses two steps for model reduction: that of finding a low-dimensional subspace and that of applying Galerkin projection. We begin with an ensemble of data obtained from the numerical simulation, consisting of samples $\{\mathbf{Y}_1, \dots, \mathbf{Y}_N\}$ of $\mathbf{y}(t)$, and try to find an M-dimensional subspace that best approximates the underlying n-dimensional state space via a projection.

2.1. The POD procedure. POD provides us a method for achieving this task. The idea is to find a vector $\mathbf{\Phi}$ of length n that has a structure typical of the members of the ensemble data, $\mathbf{Y}_i, i = 1, 2, \dots, N$, where $\mathbf{Y}_i = (Y_{i1}, Y_{i2}, \dots, Y_{in})$. We can achieve this by maximizing

$$\max_{\mathbf{\Phi}} \frac{E\{(\mathbf{Y}^T\mathbf{\Phi})(\mathbf{Y}^T\mathbf{\Phi})\}}{\mathbf{\Phi}^T\mathbf{\Phi}}, \tag{2.2}$$

where $E\{\cdot\}$ is a suitably defined averaging operation. It is clear from the properties of the vector that we can rewrite (2.2) as

$$\max_{\mathbf{\Phi}} \frac{(\mathbf{R\Phi})^T\mathbf{\Phi}}{\mathbf{\Phi}^T\mathbf{\Phi}},$$

where $\mathbf{R} = E\{\mathbf{YY}^T\}$. Then the first-order necessary optimality condition for $\mathbf{\Phi}$ to provide a maximum in (2.2) is

$$\mathbf{R\Phi} = \lambda\mathbf{\Phi}, \tag{2.3}$$

which is an eigenvalue problem with n eigenvalues and eigenvectors. Since $\mathbf{R}$ is a symmetric matrix, we can extract an orthonormal basis $\mathbf{\Phi}_n$ for $\mathbb{R}n$ from the eigenvectors. Thus $\mathbf{\Phi}_n$ form a basis for the realizations in our ensemble, i.e.,

$$\mathbf{Y} = \sum_{i=1}^{n} (\mathbf{Y}^T\mathbf{\Phi}_i)\mathbf{\Phi}_i = \sum_{i=1}^{n} \alpha_i \mathbf{\Phi}_i.$$

Moreover, we have that α_i are uncorrelated with one another on average,

$$E\{\alpha_i\alpha_j\} = (\mathbf{R}\Phi_i)^T\Phi_j = \lambda_i\mathbf{\Phi}_i^T\Phi_j = \lambda_i\delta_{ij}$$

and

$$E\{\mathbf{Y}^T\mathbf{Y}\} = \sum_{i=1}^{n} E\{|\mathbf{Y\Phi}_i|^2\} = \sum_{i=1}^{n} \lambda_i,$$

so that the eigenvalues are the mean-square values of α_i, i.e., the information in the various modes, and the sum of the eigenvalues is the total information.

For practical calculations, the number of grid points n can be rather large leading to a very large eigenvalue problem. In order to save time in the computation of eigenfunction, a useful method was proposed in [23], where it was called the *method of snapshots.* In the method of snapshots, instead of solving for the eigensystem of an $n \times n$ matrix we need only deal with an $N \times N$ matrix, where N is the number of ensembles. In order to describe the steps, first assume that the averaging operation is of the form

$$E[\mathbf{YY}^T] = \frac{1}{N} \sum_{i=1}^{n} \mathbf{Y}_i \mathbf{Y}_i^T;$$

then (2.3) becomes

$$\frac{1}{N}\sum_{i=1}^{n}\mathbf{Y}_i\mathbf{Y}_i^T\mathbf{\Phi} = \lambda\mathbf{\Phi}. \tag{2.4}$$

We next assume that the eigenvector $\mathbf{\Phi}$ can be written in terms of the members of the ensemble as

$$\mathbf{\Phi} = \sum_{i=1}^{N} w_i\mathbf{Y}^{(i)}. \tag{2.5}$$

Substituting (2.4) into (2.3), we have

$$\mathbf{CW} = \lambda\mathbf{W},$$

where

$$\mathbf{C}_{ij} = \frac{1}{N}\mathbf{Y}_i^T\mathbf{Y}_j \quad \text{and} \quad \mathbf{W} = \begin{bmatrix} w_1 \\ w_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ w_N \end{bmatrix}.$$

It follows from the fact that $\mathbf{C}$ is a nonnegative Hermitian matrix that it has a complete set of orthogonal eigenvectors $\mathbf{W}_1, \mathbf{W}_2, \dots, \mathbf{W}_N$ along with a set of eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N \geq 0$. We also normalize these by requiring

$$(\mathbf{W}_l, \mathbf{W}_l) = \sum_{i=1}^{N} w_i^l w_i^{l*} = \frac{1}{N\lambda_l}.$$

It is now easy to check

$$(\Phi_l, \Phi_m) = \begin{cases} 1, & l = m, \\ 0, & l \neq m. \end{cases}$$

This completes the construction of the orthonormal set $\{\Phi_1, \Phi_2, \dots, \Phi_N\}$.

2.1.1. Optimality properties. The optimality properties of POD, including those discussed in section 2.1.1, are summarized below. These results can be found in statistical framework terms in [23] and elsewhere.

Proposition 1. *Let $\{\mathbf{\Phi}_1, \mathbf{\Phi}_2, \dots, \mathbf{\Phi}_N\}$ be the POD basis elements and let $\{\lambda_1, \dots, \lambda_N\}$ be the corresponding set of eigenvalues. Let $\mathbf{y}^N$ be the projection of $\mathbf{y}$ onto span$\{\mathbf{\Phi_1}, \mathbf{\Phi_2}, \dots, \mathbf{\Phi_N}\}$ and let*

$$\mathbf{y}^N = \sum_{i=1}^{N}\beta_i(t)\mathbf{\Phi}_i(\mathbf{x}).$$

Let $\{\mathbf{\Psi}_i\}_{i=1}^N$ be an arbitrary orthonormal set such that

$$\mathbf{y}^N = \sum_{i=1}^{N}\alpha_i(t)\mathbf{\Psi}_i.$$

Then the following two properties hold:

(i) $E\{\beta_i(t)\beta_j(t)\} = \delta_{ij}\lambda_i$;

(ii) *For every* $N_k \leq N$, $\displaystyle\sum_{i=1}^{N_k} E\{\beta_i(t)\beta_i(t)\} = \sum_{i=1}^{N_k}\lambda_i \geq \sum_{i=1}^{N_k} E\{\alpha_i(t)\alpha_i(t)\}$.

In essence this proposition states that among all the linear combinations, the one which corresponds to POD is the best in the sense that it will capture the most information possible in the average sense. Moreover, the coefficients β_i are uncorrelated, and thus the claim that the POD expansion is efficient for modeling $\mathbf{y}(\mathbf{x}, t)$. Moreover, the accumulated error of this optimal truncated representation is given by

$$\varepsilon = \sum_{i=1}^{N}\left\|\mathbf{y}^i - \sum_{j=1}^{M}(\mathbf{y}^i, \mathbf{\Phi}^j)\mathbf{\Phi}^j\right\|^2 = \sum_{i=1}^{N}\left\|\sum_{j=M+1}^{N}(\mathbf{y}^i, \mathbf{\Phi}^j)\mathbf{\Phi}^j\right\|^2,$$

where $\mathbf{y}^i = \sum_{j=1}^{N}(\mathbf{u}^i, \mathbf{\Phi}^j)\mathbf{\Phi}^j, i = 1, \ldots, N$, and can be computed using singular value analysis; see [4, 23].

Since the eigenvalues can be used to find how close the M-dimensional subspace approximant is to the data set, one can seek M such that the fraction of the total energy

$$\sum_{i=1}^{M}\lambda_i \Bigg/ \sum_{i=1}^{N}\lambda_i$$

is close to one and yet $M \ll N$. Then the POD reduced basis subspace is defined as $\mathbf{V}^{POD} = \text{span}\{\Phi_1, \Phi_2, \ldots, \Phi_M\}$.

2.2. The Galerkin projection and the reduced-order model. Given a collection of N snapshots $\mathbf{Y}_k$, $k = 1, \ldots, N$, as possible states of the system (2.1), we define the modified snapshots set

$$\mathbf{y}_k = \mathbf{Y}_k - \overline{\mathbf{Y}}, \qquad k = 1, \ldots, N,$$

by subtracting the mean $\overline{\mathbf{Y}} = \frac{1}{N}\sum_{k=1}^{N}\mathbf{Y}_k$ and apply POD to this set to find a reduced-order subspace $\mathbf{V}^{POD} = \text{span}\{\Phi_1, \Phi_2, \ldots, \Phi_M\}$.

In the Galerkin projection, we write $\mathbf{y}(t)$ as

$$\mathbf{y}(t) = P^*\boldsymbol{\alpha}(t) + \mathbf{r}(t),$$

where $\boldsymbol{\alpha}(t) \in \mathbb{R}M$ and P is a suitable projection. Inserting this expression into (2.1), we get

$$P^*\dot{\boldsymbol{\alpha}}(t) + \dot{\mathbf{r}}(t) = \mathcal{E}(P^*\boldsymbol{\alpha}(t) + \mathbf{r}(t)).$$

The Galerkin method then enforces the residual to be orthogonal to the M-dimensional subspace such that $P\mathbf{r}(t) = 0$. This leads to

$$\dot{\boldsymbol{\alpha}}(t) = P\mathcal{E}(P^*\boldsymbol{\alpha}(t) + \mathbf{r}(t)).$$

If we now choose the projection P such that $r(t)$ is small in some sense, then the reduced-order model is given by

$$\dot{\boldsymbol{\alpha}}(t) = P\mathcal{E}(P^*\boldsymbol{\alpha}(t)).$$

3. The thermal channel flow system. In this section, we describe the thermal system and the governing equations with relevant boundary conditions. We also describe the discretization schemes and solution techniques to solve these equations.

3.1. The governing equations. The class of thermally convective flow we consider is modeled by a thermally coupled Navier–Stokes equations that uses Boussinesq approximations based on certain assumptions about the thermodynamics and the thermal effects on the flow. The first one is that variations in density are negligible except for the body force term $\rho\mathbf{g}$ in the momentum equations, where ρ is the density and the vector $\mathbf{g}$ is the constant acceleration of gravity. We next assume that the density ρ in the term $\rho\mathbf{g}$ can be given by $\rho = \rho_0[1 - \beta(T - T_0)]$, where T_0 and ρ_0 are reference temperature and density, respectively, T is the absolute temperature, and β is the thermal expansion coefficient. Furthermore, we assume that in the energy equation, the dissipation of mechanical energy is negligible and the viscosity μ, the heat conductivity κ, the thermal expansion coefficient β, and the specific heat at constant pressure c_p are constant. Then under these assumptions the flow is governed by the equations

$$\begin{aligned}
&\mathbf{u}_t - \mu\Delta\mathbf{u} + \rho_0(\mathbf{u}\cdot\nabla)\mathbf{u} + \operatorname{grad} p = \mathbf{g}\rho_0[1 - \beta(T - T_0)]\,,\\
&T_t - \kappa\,\Delta T + \rho_0 c_p \mathbf{u}\cdot\nabla T = 0\,,\\
&\nabla\cdot\mathbf{u} = 0
\end{aligned}$$

in the domain $\Omega \times [0, T]$, where Ω is a bounded open set and the heat source is assumed to be zero. If we assume there is a length scale ℓ, a velocity scale $V_{\max}$, and a temperature scale $T_1 - T_0$ in the flow, then one can define nondimensional Prandtl number $Pr = \mu c_p/\kappa$, Grashof number $Gr = \beta\ell^3\rho_0^2|\mathbf{g}|(T_1 - T_0)/\mu^2$, and Reynolds number $Re = \rho_0 V_{\max}\ell/\mu$. Next, if we nondimensionalize according to $\mathbf{x} \leftarrow \mathbf{x}/\ell$, $\mathbf{u} \leftarrow \mathbf{u}/V_{\max}$, $T \leftarrow (T - T_0)/(T_1 - T_0)$, and $p \leftarrow (p - \mathbf{g}\cdot\mathbf{x})/(\rho_0 {V_{\max}}^2)$, we obtain the following nondimensional form of Boussinesq equations (uncontrolled system (S_u)):

$$\begin{aligned}
&\mathbf{u}_t - \frac{1}{Re}\Delta\mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \operatorname{grad} p + \frac{Gr}{Re^2}T\mathbf{g} = \mathbf{0}\,,\\
&T_t - \frac{1}{RePr}\,\Delta T + \mathbf{u}\cdot\nabla T = 0\,,\\
&\nabla\cdot\mathbf{u} = 0
\end{aligned}\tag{3.1}$$

in the domain $\Omega \times [0, T]$, where $\mathbf{g}$ is now a unit vector in the direction of gravitational acceleration.

For all of our numerical simulations in this article, we choose the two-dimensional flow in a backward-facing step channel as our flow setting. A schematic of the geometry is given in Figure 1. The height of the inflow boundary is 0.5 and that of the outflow boundary is 1. The length of the narrower section of the channel is 1 and that of the wider section of the channel is 7 (the total horizontal length is 8). At the inflow boundary a parabolic velocity profile is prescribed, i.e., $u(x = 0,\ 1/2 \le y \le 1) = 8(y - 1/2)(1 - y)$, $v(x = 0,\ 1/2 \le y \le 1) = 0$ (the coordinate system is chosen such that $y = 1$ on the top boundary), which produces a maximum inflow velocity of $u_{max} = 1/2$. On the solid walls the no-slip condition ($\mathbf{u} = 0$) is imposed. The pseudo stress-free condition,

$$-p + \frac{1}{Re}\frac{\partial u}{\partial x} = 0 \quad\text{and}\quad \frac{\partial v}{\partial x} = 0,$$

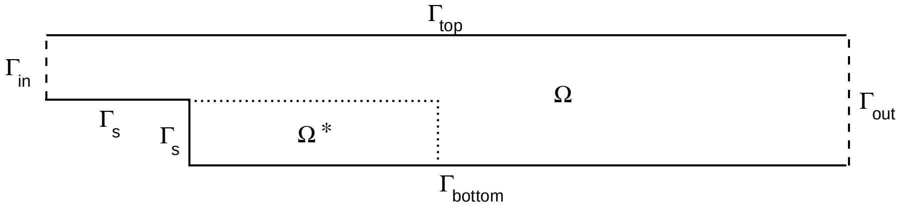


Fig. 1. *Computational domain for the backward-facing step channel problem.*

at the outflow boundary is not "physical" but is used to represent the flow in an unbounded region; see [22]. The boundary condition for the temperature is taken to be $T = 2$ except at the outflow boundary where we assume $\frac{\partial T}{\partial n} = 0$. We define the Reynolds number as $Re = \frac{V_{max} \cdot H}{\nu}$, where V_{max}= maximum inlet velocity, H = channel height, ν= kinematic viscosity of the fluid. Throughout this simulation we choose $V_{max} = \frac{1}{2}$ and $H = 1$ with a corresponding $Re = \frac{1}{2\nu}$.

3.2. Discretizations and solution techniques of the governing equations. In this section, we introduce discretizations and solution techniques of the governing equations. For spatial discretization, we use a mixed finite element method, and for the time discretization we use the strongly A-stable backward Euler method. In setting up the finite element model of the governing equations, one starts with a weak form. We define the weak form as

$$\begin{aligned} &(\mathbf{u}_t + \mathbf{u} \cdot \nabla \mathbf{u}, \mathbf{v}) + \frac{1}{Re}(\nabla \mathbf{u}, \nabla \mathbf{v}) - (p, \nabla \cdot \mathbf{v}) + \frac{Gr}{Re^2}(T\mathbf{g}, \mathbf{v}) = 0\,, \\ &(T_t + \mathbf{u} \cdot \nabla T, \psi) + \frac{1}{RePr}(\nabla T, \nabla \psi) = 0\,, \\ &(\nabla \cdot \mathbf{u}, q) = 0\,, \qquad \mathbf{u}(0) = \mathbf{u}_0\,, \qquad T(0) = T_0 \end{aligned} \tag{3.2}$$

for all test functions $(\mathbf{v}, q, \psi) \in \mathbf{V} \times L^2(\Omega) \times V_1$, where

$$\mathbf{V} = \{\mathbf{v} \in \mathbf{H}^1(\Omega) : \quad \mathbf{v}|_{\Gamma \backslash \Gamma_{\text{out}}} = 0\},$$

$$V_1 = \{\psi \in H^1(\Omega) : \quad \psi|_{\Gamma \backslash \Gamma_{\text{out}}} = 0\},$$

and $(\cdot, \cdot)$ denotes the $L^2(\Omega)$ or $\mathbf{L}^2(\Omega)$ inner product. The state variables $(\mathbf{u}, p, T)$ for the problem are taken to be

$$\mathbf{u} \in L^2(0, T; \mathbf{H}^1(\Omega)), \qquad \mathbf{u}|_{\Gamma_{\text{in}}} = \mathbf{u}_{\text{in}}, \qquad \text{and } \ \mathbf{u}|_{\Gamma \backslash \Gamma_{\text{out}} \backslash \Gamma_{\text{in}}} = 0,$$

$$T \in L^2(0, T; H^1(\Omega)), \qquad T|_{\Gamma \backslash \Gamma_{\text{out}}} = T_b, \qquad \text{and } \ T_b \in L^2(0, T; H^{1/2}(\Gamma \setminus \Gamma_{\text{out}})),$$

$$p \in L^2(\Omega), \qquad \mathbf{u}_0, T_0 \in L^2(\Omega), \qquad \text{and } \ \mathbf{u}_{\text{in}} \in L^2(0, T; H^{1/2}(\Gamma_{\text{in}})).$$

On the finite mesh covering the domain with local element width h, one defines polynomial trial functions for velocity and pressure. Let $\mathcal{K}^h$ be a standard finite element triangulation of Ω, where h is the maximal length of all the triangulation edges in $\mathcal{K}_h$. Let $\mathcal{P}^k$ to be the space of all polynomials of degree less than or equal to k and let

$$\mathbf{V}^h = \{\mathbf{v}^h | \ \mathbf{v}^h \in C^0(\bar{\Omega}) \times C^0(\bar{\Omega}), \ \ \mathbf{v}^h|_K \in \mathcal{P}^2 \times \mathcal{P}^2 \quad \forall K \in \mathcal{K}^h\},$$

$$V^h = \{v^h | \ v^h \in C^0(\bar{\Omega}), \ \ v^h|_K \in \mathcal{P}^2 \quad \forall K \in \mathcal{K}^h\},$$

$$P^h = \{q^h | \ q^h \in C^0(\bar{\Omega}), \ \ q^h|_K \in \mathcal{P}^1 \quad \forall K \in \mathcal{K}^h\}.$$

These spaces $\mathbf{V}_h$ and P^h should lead to numerically stable approximations as $h \to 0$, i.e., they should satisfy the Babuska–Brezzi condition with a mesh-independent constant γ (see [6]),

$$\min_{p_h \in P_h} \max_{\mathbf{v}_h \in \mathbf{V}_h} \frac{(p_h, \nabla \cdot \mathbf{v}_h)}{\|\mathbf{v}_h\|_1 \, \|p_h\|_0} \geq \gamma \geq 0 \,.$$

Many stable pairs are available in the literature. Our choice is the triangular element, which uses piecewise quadratic polynomial shape functions for the velocities and a piecewise linear polynomial for the pressure. The temperature is also defined using piecewise quadratic polynomial shape functions defined on the same triangle. A triangular grid is generated as follows. The domain is first divided into squares, and then each square is subdivided into two triangles by cutting from bottom right to top left. The velocity, temperature, and pressure are defined on the same triangulation, and on each triangle the degrees of freedom for quadratic elements are the function values at the vertices and midpoints of each edge; the degrees of freedom for linear elements are the function values at the vertices.

We now restrict the weak form (3.2) to $\mathbf{V}_h \times V_h \times P_h$ to obtain the finite element model: *Seek* $(\mathbf{u}_h, p_h, T_h)$ *such that* $\mathbf{u}_h|_{\Gamma_{in}} = \mathbf{u}_{in}, \ \ \mathbf{u}_h|_{\Gamma \backslash \Gamma_{out} \backslash \Gamma_{in}} = 0$, $T_h|_{\Gamma \backslash \Gamma_{out}} = 0$, *and*

$$\begin{aligned} &\left(\frac{\partial \mathbf{u}_h}{\partial t} + \mathbf{u}_h \cdot \nabla \mathbf{u}_h + \frac{Gr}{Re^2} T_h \mathbf{g}, \mathbf{v}_h \right) + \frac{1}{Re} (\nabla \mathbf{u}_h, \nabla \mathbf{v}_h) - (p_h, \nabla \cdot \mathbf{v}_h) = 0 \,, \\ &\left(\frac{\partial T_h}{\partial t} + \mathbf{u}_h \cdot \nabla T_h, \psi_h \right) + \frac{1}{RePr} (\nabla T_h, \nabla \psi_h) = 0 \,, \\ &(\nabla \cdot \mathbf{u}_h, q_h) = 0 \,, \qquad \mathbf{u}_h^0 = \mathbf{u}_0, \qquad T_h^0 = T_0 \end{aligned} \tag{3.3}$$

for all $(\mathbf{v}_h, T_h, q_h) \in \mathbf{V}^h \times V^h \times P^h$. In matrix notation, with $\mathbf{U}$, T, and P denoting the nodal values of the velocity, temperature, and pressure, respectively, the system (3.3) is equivalent to

$$\begin{aligned} &M\dot{\mathbf{U}} + A\mathbf{U} + N(\mathbf{U})\mathbf{U} + B^T \mathbf{P} + DT = 0, \\ &N\dot{T} + E(\mathbf{U})T + FT = 0, \\ &B\mathbf{U} = 0 \,. \end{aligned} \tag{3.4}$$

We discretize the time derivative in (3.4) by a one-step θ-scheme, namely, the A-stable backward Euler method with $\theta = 1/2$.

Given $\mathbf{U}^n = \mathbf{U}(n\Delta t)$, $T^n = T(n\Delta t)$, *and the time step* $\Delta t = t_{n+1} - t_n$, *solve for* $\mathbf{U}^{n+1}$, T^{n+1}, *and* P^{n+1} *from*

$$\begin{aligned} &[M + \Delta t \, \theta(A + N(\mathbf{U}^{n+1})]\mathbf{U}^{n+1} + \Delta t \, B^T P^{n+1} + \Delta t \, \theta D T^{n+1} = \mathbf{f}^n \,, \\ &[N + \Delta t \, \theta(F + E(\mathbf{U}^{n+1})]T^{n+1} = \mathbf{h}^n \,, \\ &B\mathbf{U}^{n+1} = 0 \,, \end{aligned} \tag{3.5}$$

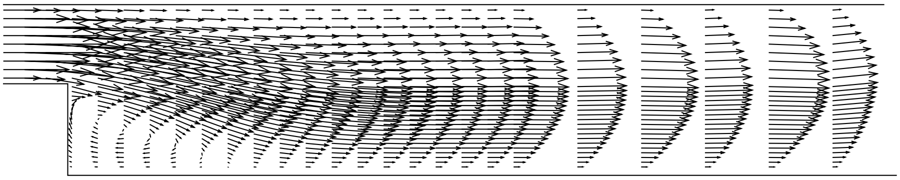


FIG. 2. *Baseline/uncontrolled flow; vector fields of velocity at $t = 100$.*

with the right-hand sides

$$\mathbf{f}^n = [M + \Delta t(\theta - 1)(A + N(\mathbf{U}^n))]\mathbf{U}^n + \Delta t(\theta - 1)T^n\,,$$
$$\mathbf{h}^n = [N + \Delta t(\theta - 1)(F + E(\mathbf{U}^n))]T^n.$$

In each time step we have to solve a nonlinear algebraic system. Our approach to solving this discrete nonlinear problem is to linearize the convective term by a constant extrapolation in time; for example, we approximate $N(\mathbf{U}^{n+1})\mathbf{U}^{n+1}$ by $N(\mathbf{U}^n)\mathbf{U}^{n+1}$. For the solution of the discrete linear problem, we use banded Gaussian elimination with pivoting.

The computational grid was nonuniform in both the streamwise and cross-flow coordinate directions. A fine grid was used in regions where sharp variations in velocities were expected. All the computations were done with a 45 × 45 grid and a time step size $\Delta t = 1/200$ for the Reynolds number 200, the Prandtl number 0.72, and the Grashof number $Gr = 40{,}000$. The flow separates at the corner of the step and a recirculation forms. After the reattachment of the lower wall eddy, the flow slowly recovers toward a fully developed Poiseuille flow. The resulting steady flow field is given in Figure 2.

To verify the grid independence of the solution, the same problem was solved halving the grid size, i.e., using a 90 × 90 grid and a time step size $\Delta t' = 1/400$. The results from both grids agreed well and predicted the reattachment point on the downstream lower wall. Typical computations require about 180 CPU seconds per time step on a Sun Ultra 60, while the time integration goes for 2000 time steps.

4. Reduced-order adaptive optimal control approach.

4.1. The control problem for the channel. Minimization of the vorticity level in a flow domain is of interest in control/delay of transition of flow past bluff bodies. Thus in this section we formulate a related optimal control problem in channel flow. The flow configuration considered is a backward-facing step channel. As the Reynolds number is increased, the flow separates near the corner of the step. The objective of the optimal control is to reduce the size of the recirculation and hence of the length of reattachment. The control action is affected through boundary temperature on Γ_c. In terms of the boundary condition, it takes the following form along the boundary Γ_c:

$$T = c(t) \quad \text{on } \Gamma_c \times [0, T],$$

where $c(t) : [0,T] \to \mathbb{R}$. The *controlled system* (S_c) we consider is

$$\begin{aligned}
&\mathbf{u}_t - \frac{1}{Re}\Delta\mathbf{u} + \mathbf{u}\cdot\nabla\mathbf{u} + \nabla p + \frac{Gr}{Re^2}T\mathbf{g} = 0\,,\\
&T_t - \frac{1}{RePr}\,\Delta T + \mathbf{u}\cdot\nabla T = 0\,,\\
&\nabla\cdot\mathbf{u} = 0, \qquad \mathbf{u}(\mathbf{x},0) = \mathbf{u}_0(\mathbf{x}), \qquad T(0) = T_0
\end{aligned} \tag{4.1}$$

in the domain $\Omega \times [0,T]$ and the boundary conditions

$$\begin{array}{lll}
\Gamma_{\text{in}}: & \mathbf{u} = (8(0.5-y)(1-y),0); & T = 2;\\
\Gamma_{\text{out}}: & -p + \dfrac{1}{Re}\dfrac{\partial u}{\partial x} = 0 \quad \text{and} \quad \dfrac{\partial v}{\partial x} = 0; & \dfrac{\partial T}{\partial n} = 0;\\
\Gamma_{\text{top}}: & \mathbf{u} = (0,0); & T = 2;\\
\Gamma_c = \Gamma_s \cup \Gamma_{\text{bottom}}: & \mathbf{u} = (0,0); & T = c(t).
\end{array}$$

The choice of cost functional or objective functional to meet the control objective of reducing the recirculation is not trivial. Here we will consider a functional of the form

$$\mathcal{G}(\mathbf{u}) = \int_\Omega |\nabla\times\mathbf{u}|^2\,d\mathbf{x}$$

which corresponds to minimization of vorticity levels in the flow. The optimal control problem is defined as follows: *find $c(t)$ such that the cost functional $\mathcal{J}(\mathbf{u},U) = \frac{1}{2}\int_0^T \left[\mathcal{G}(\mathbf{u}) + \beta|U|^2\right] dt$ is minimized subject to the constraints that the flow fields satisfy the controlled system (S_c).*

Note here that by including a term involving U one minimizes the rate of change of controlling temperature. The parameter $\beta > 0$ adjusts the relative weight of the two terms in the functional. Solutions of this optimal control problem require numerical approximation. For a typical control problem in flow control, approximations are large scale, involving hundreds of thousands of variables; see, for example, [12, 9, 10, 7, 25]. Below we discuss reduced-order controller design using POD.

4.2. Construction of POD basis functions. When designing optimal control without POD-based reduced-order models, one needs to solve the full Navier–Stokes model and its adjoint equations during each iteration of the optimization algorithm. Here we present an approach that requires the solution of the full Navier–Stokes model only when we need to update the model; thus the optimization process is much faster, resulting in substantial savings in computational work. In order to construct a reduced-order model, one needs an ensemble. As explained earlier, we use snapshots in time of the flow fields. As described in [21, 20], it is necessary to remove the inhomogeneities on the boundary. A convenient way to do this is to introduce a reference flow field $(\mathbf{u}_r, p_r, T_r)$. Let $(\mathbf{u}_{r_1}, p_{r_1}, T_{r_1})$ be a flow field satisfying the steady state Navier–Stokes model

$$\begin{aligned}
&-\frac{1}{Re}\Delta\mathbf{u} + \mathbf{u}\cdot\nabla\mathbf{u} + \nabla p + \frac{Gr}{Re^2}T\mathbf{g} = 0,\\
&-\frac{1}{RePr}\,\Delta T + \mathbf{u}\cdot\nabla T = 0\,,\\
&\nabla\cdot\mathbf{u} = 0
\end{aligned} \tag{4.2}$$

in the domain Ω and the boundary conditions

$$\begin{array}{lll}
\Gamma_{\text{in}}: & \mathbf{u}=(8(0.5-y)(1-y),0); & T=2; \\
\Gamma_{\text{out}}: & -p+\dfrac{1}{Re}\dfrac{\partial u}{\partial x}=0 \quad \text{and} \quad \dfrac{\partial v}{\partial x}=0; & \dfrac{\partial T}{\partial n}=0; \\
\Gamma_{\text{top}}: & \mathbf{u}=(0,0); & T=2; \\
\Gamma_c=\Gamma_s\cup\Gamma_{\text{bottom}}: & \mathbf{u}=(0,0); & T=c_r,
\end{array}$$

where $c_r(=1)$ is a constant temperature field imposed on the control boundary Γ_c, and let $(\mathbf{u}_{r_0},p_{r_0},T_{r_0})$ be another set of flow field satisfying the above steady state Navier–Stokes model with the boundary value $c_r=0$ on the control boundary Γ_c. We next set $(\mathbf{u}_r,p_r,T_r)=(\mathbf{u}_{r_1},p_{r_1},T_{r_1})-(\mathbf{u}_{r_0},p_{r_0},T_{r_0})$. Then each flow field in the modified snapshot set

$$\left\{(\mathbf{u}(\mathbf{x},t^k),p(\mathbf{x},t^k),T(\mathbf{x},t^k))-\frac{c(t^k)}{c_r}(\mathbf{u}_r(\mathbf{x}),p_r(\mathbf{x}),T_r(\mathbf{x}))\right\}, \quad k=1,\dots,N,$$

satisfies homogeneous boundary conditions on Γ_c. Finally, we let $(\mathbf{u}_m,p_m,T_m)$ be the mean flow field of these modified fields and define a new snapshot set

$$\left\{(\mathbf{u}(\mathbf{x},t^k),p(\mathbf{x},t^k),T(\mathbf{x},t^k))-\frac{c(t^k)}{c_r}(\mathbf{u}_r,p_r,T_r)-(\mathbf{u}_m,p_m,T_m)\right\}, \quad k=1,\dots,N,$$

which satisfies homogeneous boundary conditions on all boundaries. To these N snapshots satisfying homogeneous boundary conditions, we apply the POD to obtain basis functions in descending order according to the information content of the system.

4.3. The reduced-order model. We employ the Galerkin projection on the Navier–Stokes model with the above POD basis functions to derive the reduced-order model, which will later be used in the optimization algorithm to solve the optimal control problem described earlier. The flow field is decomposed as follows:

$$(\mathbf{u},p,T)=(\mathbf{u}_m,p_m,T_m)+\frac{c(t)}{c_r}(\mathbf{u}_r,p_r,T_r)+\sum_{i=1}^{M}\alpha_i(t)\mathbf{\Phi}_i, \tag{4.3}$$

where $\mathbf{\Phi}_i$ is the ith POD basis function, $\alpha_i(t)$ is the corresponding coefficient, and M is the total number of POD basis functions used in the Galerkin projection. Using the Galerkin projection, we get

$$\int_\Omega \mathbf{R}\cdot\mathbf{\Phi}_i d\Omega=0, \quad i=1,\dots,M, \tag{4.4}$$

where $\mathbf{R}=(R_1,R_2,R_3)$ and

$$\begin{aligned}
R_1&=\mathbf{u}_t-\frac{1}{Re}\Delta\mathbf{u}+\mathbf{u}\cdot\nabla\mathbf{u}+\nabla p+\frac{Gr}{Re^2}T\mathbf{g},\\
R_2&=T_t-\frac{1}{RePr}\Delta T+\mathbf{u}\cdot\nabla T,\\
R_3&=\nabla\cdot\mathbf{u}.
\end{aligned}$$

As described in [21, 20] for a similar problem, when the expansion (4.3) is inserted into (4.4) and in the cost functional $\mathcal{J}$, we get the reduced-order control problem

$$\tag{4.5} \begin{aligned} &\text{Minimize } \mathcal{J}(\mathbf{X}, U) = \int_0^T \left[\ell(\mathbf{X}) + \frac{\beta}{2} U^2\right] dt \\ &\text{subject to} \quad \dot{\mathbf{X}} = f(\mathbf{X}) + BU\,, \\ &\phantom{\text{subject to}} \quad \mathbf{X}(0) = \mathbf{X}_0, \end{aligned}$$

where $\mathbf{X} = (1, \boldsymbol{\alpha}, c)^T$, $f(\mathbf{X}) = -\mathbf{A}\mathbf{X} - \mathbf{N}(\mathbf{X})$, $\ell(\mathbf{X}) = \frac{1}{2}\mathbf{X}^T \mathcal{Q}\mathbf{X}$, $\mathcal{Q}_{i,j} = (\nabla \times \boldsymbol{\Phi}_i, \nabla \times \boldsymbol{\Phi}_j)$, $i, j = 0, \ldots, M+1$, and U is the rate of change of the control $c(t)$.

Although it is not essential for the adaptive control procedure to be described later, we discretize the problem (4.5) by using the Crank–Nicholson for the time derivative and trapezoidal rule for the time integral and rewrite it as

$$\tag{4.6} \begin{aligned} &\text{Minimize } \mathcal{J}(\mathbf{Y}) \\ &\text{subject to } F(\mathbf{Y}) = 0\,, \end{aligned}$$

where $\mathbf{Y} = (\mathbf{X}, U)$,

$$\mathcal{J}(\mathbf{Y}) = \sum_{k=1}^{N} \left[\frac{1}{2}\left(\ell(\mathbf{X}^{k-1}) + \ell(\mathbf{X}^k)\right) + h(U^k)\right] \Delta t,$$

$$F(\mathbf{Y}) = \begin{bmatrix} \dfrac{\mathbf{X}^1 - \mathbf{X}^0}{\Delta t} - \dfrac{1}{2}\left(f(\mathbf{X}^1) + f(\mathbf{X}^0)\right) + BU^1 \\ \vdots \\ \vdots \\ \vdots \\ \dfrac{\mathbf{X}^N - \mathbf{X}^{N-1}}{\Delta t} - \dfrac{1}{2}\left(f(\mathbf{X}^N) + f(\mathbf{X}^{N-1})\right) + BU^N \end{bmatrix},$$

and $\Delta t = T/N$. The optimal control problem (4.6) is a constrained optimization problem. An efficient method for solving optimization problems is the SQP method, which can be viewed as an application of Newton's method to the first order necessary conditions of optimality in the present case. The SQP method is preferred over the unconstrained optimization methods due to its superior convergence properties, among others. Its application to solve optimal control of Navier–Stokes flow has been demonstrated in [9, 10] and [19]. In [21, 11], we applied it to solve a reduced-order optimal control problem in Navier–Stokes flow. Here we present an adaptive procedure where one successively updates the state information within the SQP framework.

In SQP methods, a sequence of iterates is generated by solving a quadratic programming (QP) subproblem, converging to a solution of the constrained nonlinear optimization problem. Each QP subproblem minimizes a quadratic model of the form

$$\tag{4.7} \begin{aligned} &\text{Minimize} \quad \tfrac{1}{2}(\nabla_{YY}\mathcal{L}(\mathbf{Y}_k, \boldsymbol{\zeta}_k)(\mathbf{Y} - \mathbf{Y}_k), \mathbf{Y} - \mathbf{Y}_k) + (\nabla_Y \mathcal{L}(\mathbf{Y}_k, \boldsymbol{\zeta}_k), \mathbf{Y} - \mathbf{Y}_k) \\ &\text{subject to} \quad F_Y(\mathbf{Y}_k)(\mathbf{Y} - \mathbf{Y}_k) + F(\mathbf{Y}_k) = 0\,, \end{aligned}$$

where $\mathcal{L}(\mathbf{Y},\boldsymbol{\zeta}) = \mathcal{J}(\mathbf{Y}) + (F(\mathbf{Y}),\boldsymbol{\zeta})$ is the Lagrangian function. Once the QP solution $(\widehat{\mathbf{Y}}_k, \widehat{\boldsymbol{\zeta}}_k)$ has been determined, the major iteration proceeds by determining new variables

$$
\begin{aligned}
\mathbf{Y}_{k+1} &= \mathbf{Y}_k + \eta_k(\widehat{Y}_k - \mathbf{Y}_k),\\
\boldsymbol{\zeta}_{k+1} &= \boldsymbol{\zeta}_k + \eta_k(\widehat{\boldsymbol{\zeta}}_k - \boldsymbol{\zeta}_k)\,.
\end{aligned}
\tag{4.8}
$$

One of the properties of the SQP method is that the SQP iterations do not usually satisfy the nonlinear constraints except as the solution is approached.

In our application the constraint $F(\mathbf{Y}) = 0$ is provided by the reduced-order model obtained through the POD procedure. Therefore the reduced-order model must predict the system behavior accurately for various trajectories of the boundary temperature $c(t)$ that include not only the optimal trajectory but also other trajectories appearing during the optimization iterations. In order to meet this requirement we propose an adaptive procedure in the next section.

4.4. Reduced-order adaptive control. Choosing the ensemble of data that represent the system behavior for various trajectories during the iterative optimization procedure is critical in POD-based control design. To meet this requirement, we propose an adaptive procedure that successively updates the reduced-order model to be used in the sequential QP optimization algorithm described earlier. We begin with an initial estimate $(\mathbf{Y}_k, \boldsymbol{\zeta}_k)$ and the corresponding initial estimate for the control c_k for the SQP method and generate the corresponding ensemble of snapshots from the Navier–Stokes model. With these snapshots, we find a POD subspace and the corresponding reduced-order model. This reduced-order model is now used in the SQP algorithm to find the new iterate $(\mathbf{Y}_{k+1}, \boldsymbol{\zeta}_{k+1})$. This process of updating the reduced-order model is continued in each of the SQP iterations until convergence is achieved. The computation using the adaptive reduced-order procedure takes the following form.

Algorithm I. Reduced-order adaptive procedure.

Select the tolerance $\epsilon > 0$ and perform the following steps:

Step 1. Set $k = 1$, $(\mathbf{Y}_{k-1}, \boldsymbol{\zeta}_{k-1}) = (\mathbf{Y}_0, \boldsymbol{\zeta}_0)$, and initial control estimate $c_{k-1} = c_0$.
Step 2. Compute the snapshots and derive the reduced-order model for the control variable c_{k-1}.
Step 3. Update the constraint and cost functional in the QP subproblem and solve for $(\widehat{\mathbf{Y}}_{k-1}, \widehat{\boldsymbol{\zeta}}_{k-1})$.
Step 4. Determine the new variables $(\mathbf{Y}_{k-1}, \boldsymbol{\zeta}_{k-1})$ and the control variable c_k.
Step 5. If $\left|c_k - c_{k-1}\right| > \epsilon$, add 1 to k and go to Step 2.
Step 6. The procedure is completed.

In the next section, we show the effectiveness and feasibility of this algorithm in the present application.

5. Computational results. For numerical computations here, the flow configuration and all the parameters are the same as in the uncontrolled simulations presented in section 3, except that the control action is in the form of temperature on the boundary $\Gamma_c = \Gamma_s \cup \Gamma_{\text{bottom}}$. The control objective is to reduce the recirculation behind the step and thus the reattachment length. The cost functional is taken to be the vorticity functional defined earlier.

We present numerical results for the adaptive POD approach in solving the optimal control problem at $Re = 200$. The snapshots for Algorithm I were updated

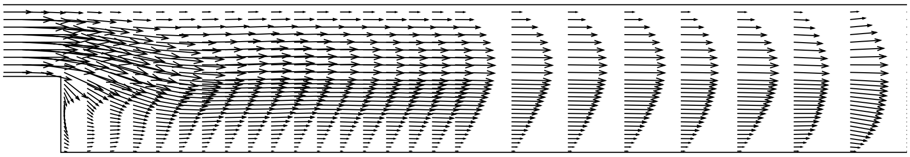


FIG. 3. *Controlled flow; vector fields of velocity at* $t = 100$.

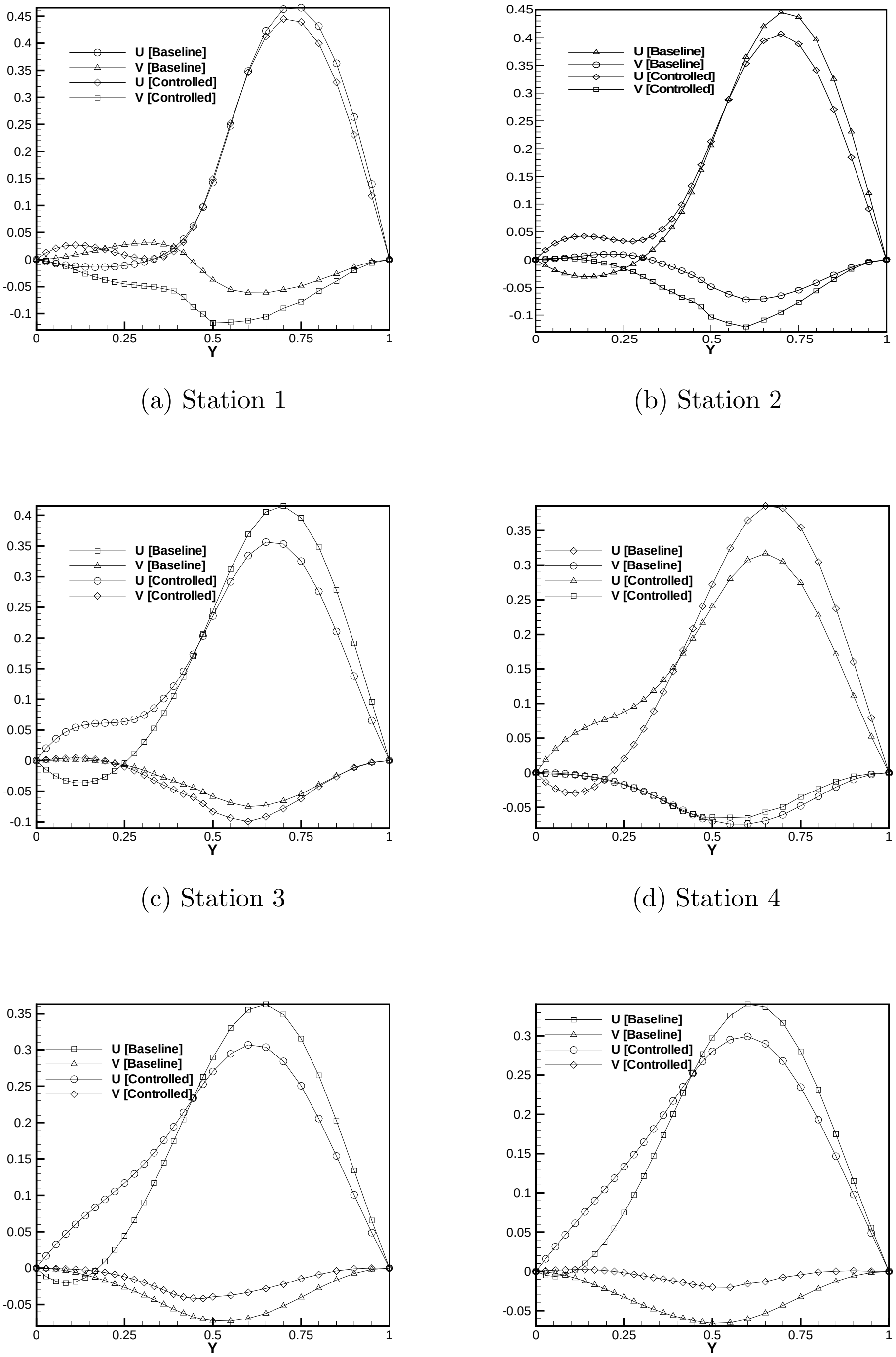
in every iteration by applying the computed optimal control $c(t)$ on Γ_c and simulating the system (S_c) in the interval $[0,T]$ with $T = 100$. A hundred snapshots were recorded at constant time intervals Δt^* $(\Delta t^* = 20\Delta t)$ for each iteration, and the POD was used to compute the dominant modes. The eigenvalues of the correlation matrix decayed rapidly, and we found only 9 modes were enough to capture 99% of the information content. Initially state and control were given a prescribed value and the optimal control problem was solved using the algorithm described in section 4. All computations were performed on a Sun Ultra 60 machine.

The adaptive algorithm terminates after about 5 iterations with significant reduction in the cost functional. In implementing Algorithm I, we carried out Step 2 in two different ways. In one implementation, we expanded the ensemble by adding new snapshots in each iteration. This implementation required more and more POD modes in each iteration to capture 99% of the information in the snapshots and thus increased the computational cost. However, the second implementation, which involved replacing the old snapshots with the new in step 2 of Algorithm I, required only 9 modes throughout the iterations and captured the required percentage of the information in the ensemble.

We carried out several simulations to study the performance of the controller. Here we present only a sample of the results. The controlled flow fields with 9, 12, and 14 modes showed similar results, and hence only results with 9 modes are presented. The flow fields presented in Figure 4 are u and v components of the flow field $\mathbf{u}$ at different stations in the channel for the controlled and uncontrolled cases with 9 POD modes. Figure 4 illustrates the suppression of flow reversal present in the uncontrolled flow. We plot the horizontal and vertical components of the baseline and controlled velocities versus the distance normal to the wall. The top two curves are for the baseline and controlled horizontal component of the velocity. We see that the optimal control has affected a very substantial reduction in the back-flow. The bottom two curves in the figures are analogous to the top curves but for the vertical component of velocity. The flow fields presented in Figures 2 and 3 are the vector fields of the baseline and controlled velocities. As indicated by the controlled flow fields, separation has been effectively eliminated by the optimal temperature. Substantial reduction in the recirculation bubble is also seen. The reattachment length has been reduced by more than 99% compared to the uncontrolled case.

Figures 5–10 show controls that were obtained in successive adaptive iterations in Algorithm I. These figures show convergence of the adaptive strategy. It is a plot of the temperature on the lower wall of the channel versus time. Note that convergence is achieved after about 6 iterations.

The penalty parameter β plays a critical role in the controller design. For $\beta \leq 1$ control oscillates at the beginning and the end of the time interval. The amplitude of



(a) Station 1

(b) Station 2

(c) Station 3

(d) Station 4

(e) Station 5

(f) Station 6

Fig. 4. *A comparison of cross-channel profiles of horizontal and vertical velocity components of the baseline flow with that of the controlled flow.*

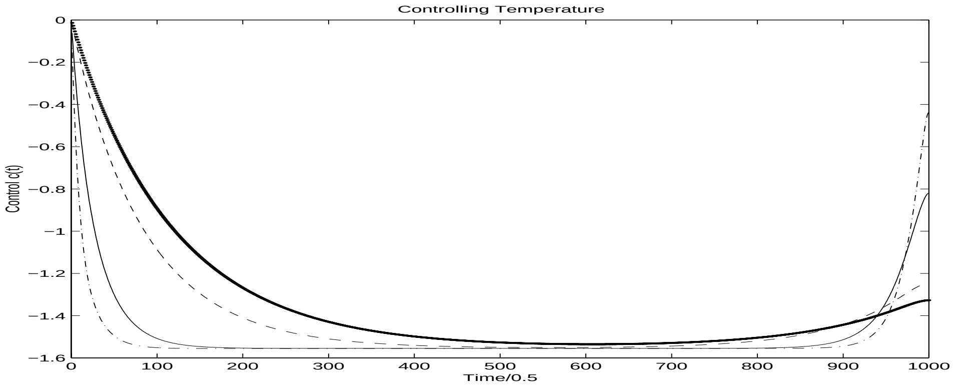


FIG. 5. *Optimal control (POD) at iteration* 1 *as a function of nondimensional time for different values of beta: $\beta = 10$ (dashed line), $\beta = 1$ (line), $\beta = 20$ (dotted line), $\beta = 0.2$ (dash-dot line).*

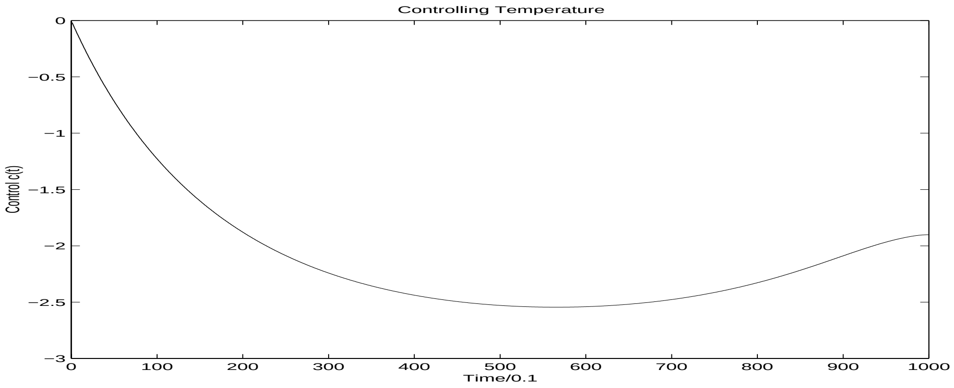


FIG. 6. *Optimal control (POD) at iteration* 2 *as a function of nondimensional time.*

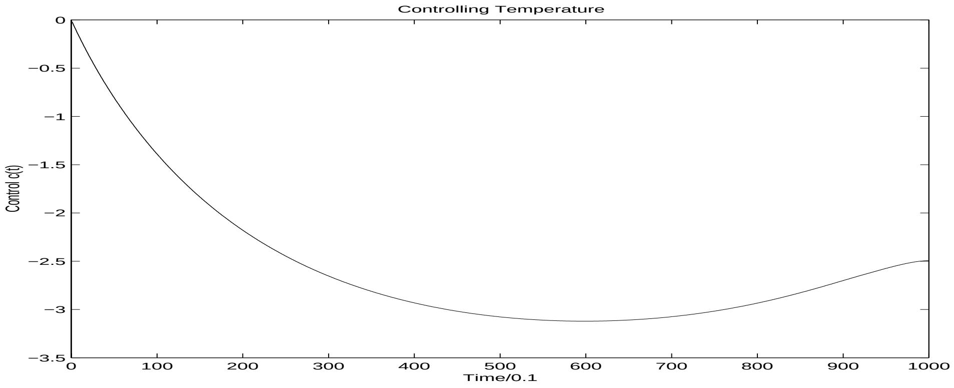


FIG. 7. *Optimal control (POD) at iteration* 3 *as a function of nondimensional time.*

the oscillation increases as more weight is placed on achieving the vorticity reduction by decreasing the parameter value β in the cost functional $\mathcal{J}$. In other words, oscillations increase as the rate of change of control U is increased; see Figure 5. Whereas for $\beta > 20$ control undershoots. With the value $\beta = 10$, a smooth control was obtained.

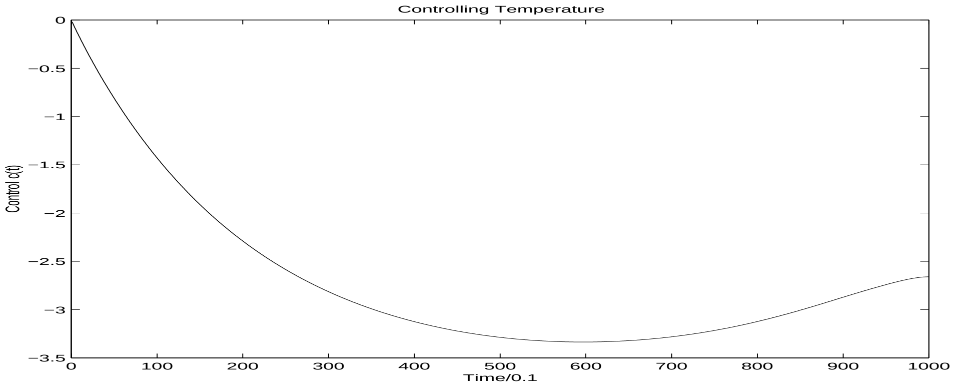


FIG. 8. *Optimal control (POD) at iteration* 4 *as a function of nondimensional time.*

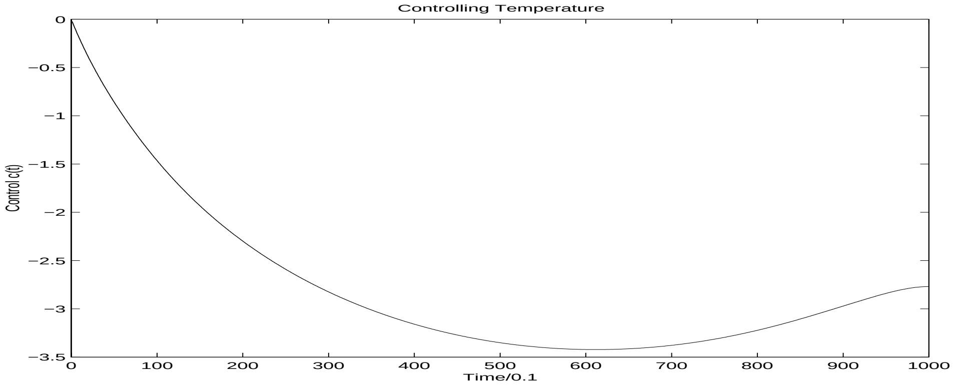


FIG. 9. *Optimal control (POD) at iteration* 5 *as a function of nondimensional time.*

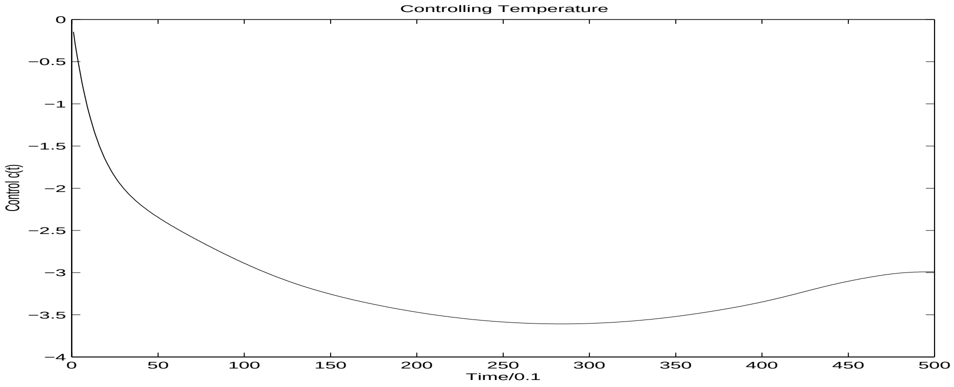


FIG. 10. *Optimal control (POD) at iteration* 6 *as a function of nondimensional time.*

In general, the computational cost for solving the optimal control problem is essentially determined by the length of the time interval and the convergence rate of the optimization process. However, one can easily see that the CPU time requirement for the reduced-order control computation is much less than that for the case of full-order control computation as the number of degrees of freedom is about 1/746 of the latter. In the control computation using the Sun Ultra 60 workstation, one iteration of the reduced-order SQP requires 3 minutes. The number of iterations needed to reach the converged optimal control is 6. During each of these iterations, a reduced-order dynamical model has to be constructed by solving the Navier–Stokes model with different control updates to generate the snapshots. This step requires 72 hours. The remaining steps in the POD take about 3 minutes. The total CPU time requirement for the adaptive reduced-order control computation is about 73 hours. This is a substantial savings in computation time since the control computation with full-order Navier–Stokes model using SQP method would take 480 CPU hours and 18 iterations to converge. This savings in computational time is mainly due to the fact that the adjoint equation of the Navier–Stokes model, which is computationally as costly as the Navier–Stokes equation itself, is computed using the reduced-order model in the adaptive reduced-order control computation. The reduction in computation time with the use of reduced-order model instead of full-order Navier–Stokes model will be much more significant in the three-dimensional case as the difference in the degrees of freedom between reduced-order model and the original full-order Navier–Stokes model will become much larger.

The location of the actuator (temperature control) was selected as the bottom wall and the vertical part of the step. This choice is motivated by the fact that if one wants maximum influence in the flow, then the control has to be applied in that vicinity. However, the optimal control can be very different depending on the location. This was evident in our computational experiments. In one experiment we used inflow temperature as control and the optimal control was heating, and in another we placed control at the top wall and the optimal control was again heating.

6. Conclusion. The optimal control techniques for flow control problems are complex and very computationally demanding. The POD method provides a systematic way to obtain reduced-order models and allows one to design controllers at drastically reduced computational cost. However, it is not clear, a priori, what is the best way to obtain an ensemble of data for POD that would lead to POD basis functions that represent the influence of the control during the optimization algorithm. Therefore the reduced-order model should be improved to represent such influences. We presented an adaptive procedure that successively updates the ensemble and the reduced-order model during the iterative optimization process using SQP. The present approach was illustrated by implementing it on the boundary optimal control problems of a thermally coupled Navier–Stokes model. Numerical experiments indicate the effectiveness of this approach. Compared to the full order optimal control approach, the reduced-order approach is computationally less costly and yet gives competitive controls. From the results presented above, we conclude that reduced-order adaptive control is feasible and provides an efficient method for solving problems of flow control, which is practical enough to be implemented in industrial processes.

REFERENCES

[1] B. F. Armaly, F. Durst, and J. C. F. Pereira, *Experimental and theoretical investigation of backward-facing step flow*, J. Fluid Mech., 127 (1983), pp. 473–496.

[2] E. Arian, M. Fahl, and E. Sachs, *Trust-Region Proper Orthogonal Decomposition for Flow Control*, ICASE Report No. 2000-25, Hampton, VA, 2000, p. 23.

[3] N. Aubry, P. Holmes, J. L. Lumley, and E. Stone, *The dynamics of coherent structures in the wall region of a turbulent boundary layer*, J. Fluid Mech., 192 (1988), pp. 115–173.

[4] G. Berkooz, P. Holmes, and J. L. Lumley, *The proper orthogonal decomposition in the analysis of turbulent flows*, Annu. Rev. Fluid Mech., 25 (1993), pp. 539–575.

[5] J. A. Burns, B. B. King, and D. Rubio, *Feedback control of thermal fluid using state estimation*, Int. J. Comput. Fluid Dyn., 11 (1998), pp. 93–112.

[6] M. D. Gunzburger, *Finite Element Methods for Viscous Incompressible Flows*, Academic Press, London, 1989.

[7] M. D. Gunzburger, ed., *Flow Control*, IMA Vol. Math. Appl. 68, Springer-Verlag, New York, 1995.

[8] M. D. Gunzburger and H. C. Lee, *Analysis, approximation, and computation of a coupled solid/fluid temperature control problem*, Comput. Methods Appl. Mech. Engrg., 118 (1994), pp. 133–152.

[9] L. S. Hou and S. S. Ravindran, *Computations of boundary optimal control problems for an electrically conducting fluid*, J. Comput. Phys., 128 (1996), pp. 319–330.

[10] L. S. Hou and S. S. Ravindran, *Numerical approximation of optimal flow control problems by a penalty method: Error estimates and numerical results,* SIAM J. Sci. Comput., 20 (1999), pp. 1753–1777.

[11] K. Ito and S. S. Ravindran, *A reduced-order method for simulation and control of fluid flows*, J. Comput. Phys., 143 (1998), pp. 403–425.

[12] K. Ito and S. S. Ravindran, *Optimal control of thermally convected fluid flows*, SIAM J. Sci. Comput., 19 (1998), pp. 1847–1869.

[13] K. Karhunen, *Zur spektral theorie stochasticher prozesse*, Ann. Acad. Sci. Fennicae Ser. A. I. Math.-Phys., 37 (1946).

[14] M. Kirby, J. P. Boris, and L. Sirovich, *A proper orthogonal decomposition of a simulated supersonic shear layer*, Internat. J. Numer. Methods Fluids, 10 (1990), pp. 411–428.

[15] M. Loeve, *Fonctions aleatoire de second ordre*, Revue, 84 (1946), pp. 195–206.

[16] J. L. Lumley, *The structure of inhomogeneous turbulence*, in Atmospheric Turbulence and Radio Wave Propagation, A.M. Yaglom and V.I. Tatarski, eds., Nauka, Moscow, 1967, pp. 166–178.

[17] P. Moin and R. D. Moser, *Characteristic-eddy decomposition of turbulence in a channel*, J. Fluid Mech., 200 (1989), pp. 417–509.

[18] M. Rajaee, S. K. F. Karlson, and L. Sirovich, *Low dimensional description of free shear flow coherent structures and their dynamical behavior*, J. Fluid Mech., 258 (1994), pp. 1401–1402.

[19] S. S. Ravindran, *Numerical approximation of optimal flow control problems by SQP method*, in Optimal Control of Viscous Flow, S. S. Sritharan, ed., SIAM, Philadelphia, 1998, pp. 181–198.

[20] S. S. Ravindran, *Proper Orthogonal Decomposition in Optimal Control of Fluids*, NASA Technical Memorandum, NASA TM 1999-209113, Hampton, VA, 1999.

[21] S. S. Ravindran, *A reduced-order approach to optimal control of fluids using proper orthogonal decomposition*, Internat. J. Numer. Methods Fluids, 34 (2000), pp. 425-448.

[22] R. L. Sani and P. M. Gresho, *Resume and remarks on the open boundary condition minisymposium*, Internat. J. Numer. Methods Fluids, 18 (1994), pp. 983–1008.

[23] L. Sirovich, *Turbulence and the dynamics of coherent structures: Part* I–III, Quart. Appl. Math., 45 (1987), pp. 561–590.

[24] L. Sirovich, *Analysis of turbulent flows by means of the empirical eigenfunctions*, Fluid Dynam. Res., 8 (1991), pp. 85–100.

[25] S. S. Sritharan, Ed., *Optimal Control of Viscous Flow*, SIAM, Philadelphia, 1998.

[26] H. V. Ly and H. T. Tran, *Proper orthogonal decomposition for flow calculation and optimal control in a horizontal CVD reactor*, Quart. Appl. Math., to appear.

SIAM J. Sci. Comput.
Vol. 23, No. 6, pp. 1943–1958

IMPLICIT INTEGRATION OF THE TIME-DEPENDENT GINZBURG–LANDAU EQUATIONS OF SUPERCONDUCTIVITY*

D. O. GUNTER[†], H. G. KAPER[†], and G. K. LEAF[†]

Abstract. This article is concerned with the integration of the time-dependent Ginzburg–Landau (TDGL) equations of superconductivity. Four algorithms, ranging from fully explicit to nonlinearly implicit, are presented and evaluated for stability, accuracy, and compute time. The benchmark problem for the evaluation is the equilibration of a vortex configuration in a superconductor that is embedded in a thin insulator and subject to an applied magnetic field.

Key words. time-dependent Ginzburg–Landau equations, superconductivity, vortex solution, implicit time integration

AMS subject classifications. Primary, 65M12, 82D55; Secondary, 35K55

PII. S1064827500375473

1. Introduction.

At the macroscopic level, the state of a superconductor can be described in terms of a complex-valued order parameter and a real vector potential. These variables, which determine the superconducting and electromagnetic properties of the system at equilibrium, are found as solutions of the Ginzburg–Landau (GL) equations of superconductivity. Because they correspond to critical points of the GL energy functional [1, 2], they can, at least in principle, be determined by minimizing a functional. In practice, one introduces a time-like variable and computes equilibrium states by integrating the time-dependent Ginzburg–Landau (TDGL) equations. The TDGL equations, first formulated by Schmid [3] and subsequently derived from microscopic principles by Gor'kov and Éliashberg [4], are nontrivial generalizations of the (time-independent) GL equations, as the time rate of change must be introduced in such a manner that gauge invariance is preserved at all times. The TDGL equations have been analyzed by several authors; see, for example, the articles [5, 6] and the references cited therein.

We are interested, in particular, in vortex solutions of the GL equations. These are singular solutions, where the phase of the order parameter changes by 2π along any closed contour surrounding a vortex point. Vortices are of critical importance in technological applications of superconductivity.

Computing vortex solutions of the GL equations by integrating the TDGL equations to equilibrium has the advantage that the solutions thus found are stable. At the same time, one obtains information about the transient behavior of the system. Integrating the TDGL equations to equilibrium is, however, a time-consuming process requiring considerable computing resources. In simulations of vortex dynamics in superconductors, which were performed on an IBM SP with tens of processors

*Received by the editors July 17, 2000; accepted for publication (in revised form) October 18, 2001; published electronically February 27, 2002. The submitted manuscript has been created by the University of Chicago as Operator of Argonne National Laboratory ("Argonne") under contract W-31-109-ENG-38 with the U.S. Department of Energy. The U.S. Government retains for itself, and others acting on its behalf, a paid-up, nonexclusive, irrevocable worldwide license in said article to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, by or on behalf of the Government. Copyright is owned by SIAM to the extent not limited by these rights.

http://www.siam.org/journals/sisc/23-6/37547.html

[†]Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439 (gunter@mcs.anl.gov, kaper@mcs.anl.gov, leaf@mcs.anl.gov).

in parallel, using a simple one-step Euler integration procedure, we routinely experienced equilibration times on the order of 100 hours [7, 8, 9]. Incremental changes would gradually drive the system to lower energy levels. These very long equilibration times arise, of course, because we are dealing with large physical systems undergoing a phase transition. The energy landscape for such systems is a broad, gently undulating plain with many shallow local minima. It is therefore important to develop efficient integration techniques that remain stable and accurate as the time step increases.

In this article we present four integration techniques for problems on rectangular domains in two dimensions. These two-dimensional domains should be viewed as cross sections of three-dimensional systems that are infinite and homogeneous in the direction of the magnetic field, which is orthogonal to the plane of the cross section. The algorithms are scalable in a multiprocessing environment and generalize to three dimensions. We evaluate the performance of each algorithm on the same benchmark problem, namely, the equilibration of a vortex configuration in a system consisting of a superconducting core embedded in a blanket of insulating material (air) and undergoing a transition from the Meissner state to the vortex state under the influence of an externally applied magnetic field. We determine the maximum allowable time step for stability, the number of time steps needed to reach the equilibrium configuration, and the CPU cost per time step.

The benchmark problem keeps the focus of the present article on the numerical issues. The problem is standard, and the equilibrium vortex configuration is certainly well known in the physics community. The capabilities of the methods extend well beyond the benchmark problem, and the methods have been used successfully to solve more challenging problems; we refer to [9] for an extensive report on several large-scale numerical simulations of vortex dynamics in superconducting media with ordered and disordered defects.

Different algorithms correspond to different dynamics through state space, so the eventual equilibrium vortex configuration may differ from one algorithm to another. Hence, once we have the equilibrium configurations, we need some measure to assess their accuracy. For this purpose we use three parameters: the number of vortices, the mean intervortex distance (bond length), and the mean bond angle taken over nearest-neighbor pairs of bonds. When each of these parameters differs less than a specified tolerance, we say that the corresponding vortex configurations are the same.

Our investigations show that one can increase the time step by almost two orders of magnitude, without losing stability, by going from the fully explicit to a nonlinearly implicit algorithm. The nonlinearly implicit algorithm has a higher cost per time step, but the wall clock time needed to compute the equilibrium solution (the most important measure for practical purposes) is still significantly less. The wall clock time can be reduced further by using a multitimestepping procedure.

In section 2, we present the GL model of superconductivity, first in its formulation as a system of partial differential equations, then as a system of ordinary differential equations after the spatial derivatives have been approximated by finite differences. In section 3, we give four algorithms to integrate the system of ordinary differential equations: an explicit, a semi-implicit, a linearly implicit, and a nonlinearly implicit algorithm. In section 4, we present and evaluate the results of the investigation. In section 5, we further evaluate the nonlinearly implicit algorithm from the point of view of parallelism and multitimestepping. The conclusions are summarized in section 6.

2. GL model. The TDGL equations of superconductivity [2, 3, 4] are two coupled partial differential equations for the complex-valued *order parameter* $\psi = |\psi| \mathrm{e}^{i\phi}$

and the real vector-valued *vector potential* $\mathbf{A}$,

$$\frac{\hbar^2}{2m_sD}\left(\frac{\partial}{\partial t}+\frac{ie_s}{\hbar}\Phi\right)\psi=-\frac{1}{2m_s}\left(\frac{\hbar}{i}\nabla-\frac{e_s}{c}\mathbf{A}\right)^2\psi+a\psi-b|\psi|^2\psi, \tag{2.1}$$

$$\nu\left(\frac{1}{c}\frac{\partial\mathbf{A}}{\partial t}+\nabla\Phi\right)=-\frac{c}{4\pi}\nabla\times\nabla\times\mathbf{A}+\mathbf{J}_s. \tag{2.2}$$

Here, $\mathbf{J}_s$ is the *supercurrent density*, which is a nonlinear function of ψ and $\mathbf{A}$,

$$\mathbf{J}_s=\frac{e_s\hbar}{2im_s}(\psi^*\nabla\psi-\psi\nabla\psi^*)-\frac{e_s^2}{m_sc}|\psi|^2\mathbf{A}=\frac{e_s}{m_s}|\psi|^2\left(\hbar\nabla\phi-\frac{e_s}{c}\mathbf{A}\right). \tag{2.3}$$

The real scalar-valued *electric potential* Φ is a diagnostic variable. The constants in the equations are $\hbar$, Planck's constant divided by 2π; a and b, two positive constants; c, the speed of light; m_s and e_s, the effective mass and charge, respectively, of the superconducting charge carriers (Cooper pairs); ν, the electrical conductivity; and D, the diffusion coefficient. As usual, i is the imaginary unit, and * denotes complex conjugation.

The quantity $|\psi|^2$ represents the local density of Cooper pairs. The local time rate of change $\partial_t\mathbf{A}$ of $\mathbf{A}$ determines the *electric field*, $\mathbf{E}=(1/c)\partial_t\mathbf{A}+\nabla\Phi$, while its spatial variation determines the (induced) *magnetic field*, $\mathbf{B}=\nabla\times\mathbf{A}$.

The TDGL equations describe the gradient flow for the GL energy functional. This functional is zero in the normal state, when $\psi=0$ and the externally applied magnetic field penetrates the superconductor everywhere, $\nabla\times\mathbf{A}=\mathbf{H}$. In the superconducting state, it is given by the expression

$$E=\int\left[\frac{1}{2m_s}\left|\left(\frac{\hbar}{i}\nabla-\frac{e_s}{c}\mathbf{A}\right)\psi\right|^2+\left(-a|\psi|^2+\frac{b}{2}|\psi|^4\right)+|\nabla\times\mathbf{A}-\mathbf{H}|^2\right]\mathrm{d}x. \tag{2.4}$$

The three terms represent the kinetic energy, the condensation energy, and the field energy, respectively. A thermodynamic equilibrium configuration corresponds to a critical point of E.

The energy functional (2.4) assumes that there are no defects in the superconductor. Material defects can be naturally present or artificially induced and can be in the form of point, planar, or columnar defects (quenched disorder). A material defect results in a local reduction of the depth of the well of the condensation energy. A simple way to include material defects in the GL model is by assuming that the parameter a depends on position and has a smaller value wherever a defect is present; see [9].

2.1. Dimensionless form. Let $\psi_\infty^2=a/b$, and let λ, ξ, and H_c denote the London penetration depth, the coherence length, and the thermodynamic critical field, respectively:

$$\lambda=\left(\frac{m_sc^2}{4\pi\psi_\infty^2e_s^2}\right)^{1/2},\quad \xi=\left(\frac{\hbar^2}{2m_sa}\right)^{1/2},\quad H_c=(4\pi a\psi_\infty^2)^{1/2}. \tag{2.5}$$

In this study, we render the TDGL equations dimensionless by measuring lengths in units of ξ, time in units of the relaxation time ξ^2/D, fields in units of $H_c\sqrt{2}$, and energy densities in units of $(1/4\pi)H_c^2$. The nondimensional TDGL equations are

$$\left(\frac{\partial}{\partial t}+i\Phi\right)\psi=\left(\nabla-\frac{i}{\kappa}\mathbf{A}\right)^2\psi+\tau\psi-|\psi|^2\psi, \tag{2.6}$$

$$\sigma\left(\frac{\partial \mathbf{A}}{\partial t}+\kappa\nabla\Phi\right)=-\nabla\times\nabla\times\mathbf{A}+\mathbf{J}_s, \tag{2.7}$$

where

$$\mathbf{J}_s=\frac{1}{2i\kappa}(\psi^*\nabla\psi-\psi\nabla\psi^*)-\frac{1}{\kappa^2}|\psi|^2\mathbf{A}=\frac{1}{\kappa}|\psi|^2\left(\nabla\phi-\frac{1}{\kappa}\mathbf{A}\right). \tag{2.8}$$

Here, $\kappa=\lambda/\xi$ is the GL parameter and σ is a dimensionless resistivity, $\sigma=(4\pi D/c^2)\nu$. The coefficient τ has been inserted to account for defects; $\tau(x)<1$ if x is in a defective region; otherwise $\tau(x)=1$. The nondimensional TDGL equations are associated with the dimensionless energy functional

$$E=\int\left[\left|\left(\nabla-\frac{i}{\kappa}\mathbf{A}\right)\psi\right|^2+\left(-\tau|\psi|^2+\tfrac{1}{2}|\psi|^4\right)+|\nabla\times\mathbf{A}-\mathbf{H}|^2\right]\mathrm{d}x. \tag{2.9}$$

2.2. Gauge choice. The (nondimensional) TDGL equations are invariant under a gauge transformation,

$$\mathcal{G}_\chi:(\psi,\mathbf{A},\Phi)\mapsto(\psi\mathrm{e}^{i\chi},\mathbf{A}+\kappa\nabla\chi,\Phi-\partial_t\chi). \tag{2.10}$$

Here, χ can be any real scalar-valued function of position and time. We maintain the zero-electric potential gauge, $\Phi=0$, at all times, using the *link variable* $\mathbf{U}$,

$$\mathbf{U}=\exp\left(-\frac{i}{\kappa}\int\mathbf{A}\right). \tag{2.11}$$

This definition is componentwise: $U_x=\exp(-i\kappa^{-1}\int^x A_x(x',y,z)\,\mathrm{d}x')$, ... The gauged TDGL equations can now be written in the form

$$\frac{\partial\psi}{\partial t}=\sum_{\mu=x,y,z}U_\mu^*\frac{\partial^2}{\partial\mu^2}(U_\mu\psi)+\tau\psi-|\psi|^2\psi, \tag{2.12}$$

$$\sigma\frac{\partial\mathbf{A}}{\partial t}=-\nabla\times\nabla\times\mathbf{A}+\mathbf{J}_s, \tag{2.13}$$

where

$$J_{s,\mu}=\frac{1}{\kappa}\,\mathrm{Im}\,\left[(U_\mu\psi)^*\frac{\partial}{\partial\mu}(U_\mu\psi)\right],\qquad \mu=x,y,z. \tag{2.14}$$

2.3. Two-dimensional problems. From here on we restrict the discussion to problems on a two-dimensional rectangular domain (coordinates x and y), assuming boundedness in the x direction and periodicity in the y direction. The domain represents a superconducting core surrounded by a blanket of insulating material (air) or a normal metal. The order parameter vanishes outside the superconductor, and no superconducting charge carriers leave the superconductor. The whole system is driven by a time-independent externally applied magnetic field $\mathbf{H}$ that is parallel to the z axis, $\mathbf{H}=(0,0,H)$. The vector potential and the supercurrent have two nonzero components, $\mathbf{A}=(A_x,A_y,0)$ and $\mathbf{J}_s=(J_x,J_y,0)$, while the magnetic field has only one nonzero component, $\mathbf{B}=(0,0,B)$, where $B=\partial_xA_y-\partial_yA_x$.

2.4. Spatial discretization. The physical configuration to be modeled (superconductor embedded in blanket material) is periodic in y and bounded in x. In the x direction, we distinguish three subdomains: an interior subdomain occupied by the superconducting material and two subdomains, one on either side, occupied by the blanket material. We take the two blanket layers to be equally thick but do not assume that the problem is symmetric around the midplane. (Possible sources of asymmetry are material defects in the system, surface currents, and different field strengths on the two outer surfaces.)

We impose a regular grid with mesh widths h_x and h_y,

$$\Omega_{i,j} = (x_i, x_{i+1}) \times (y_j, y_{j+1}), \quad x_i = x_0 + ih_x, \quad y_j = y_0 + jh_y, \tag{2.15}$$

assuming the following correspondences:

Left outer surface:	$x = x_0 + \frac{1}{2}h_x,$	$i = 0,$
Left interface:	$x = x_{n_{sx}-1} + \frac{1}{2}h_x,$	$i = n_{sx} - 1,$
Right interface:	$x = x_{n_{ex}} + \frac{1}{2}h_x,$	$i = n_{ex},$
Right outer surface:	$x = x_{n_x} + \frac{1}{2}h_x,$	$i = n_x.$

One period in the y direction is covered by the points $j = 1, \ldots, n_y$. We use the symbols Sc and Bl to denote the index sets for the superconducting and blanket region, respectively:

$$\mathrm{Sc} = \{(i,j) : (i,j) \in [n_{sx}, n_{ex}] \times [1, n_y]\}, \tag{2.16}$$

$$\mathrm{Bl} = \{(i,j) : (i,j) \in [1, n_{sx} - 1] \cup [n_{ex} + 1, n_x] \times [1, n_y]\}. \tag{2.17}$$

The order parameter ψ is evaluated at the grid vertices,

$$\psi_{i,j} = \psi(x_i, y_j), \qquad (i,j) \in \mathrm{Sc}, \tag{2.18}$$

the components A_x and A_y of the vector potential at the midpoints of the respective edges,

$$A_{x;i,j} = A_x(x_i + \tfrac{1}{2}h_x, y_j), \quad A_{y;i,j} = A_y(x_i, y_j + \tfrac{1}{2}h_y), \quad (i,j) \in \mathrm{Sc} \cup \mathrm{Bl}, \tag{2.19}$$

and the induced magnetic field B at the center of a grid cell,

$$\begin{aligned} B_{i,j} &= B(x_i + \tfrac{1}{2}h_x, y_j + \tfrac{1}{2}h_y) \\ &= \frac{A_{y;i+1,j} - A_{y;i,j}}{h_x} - \frac{A_{x;i,j+1} - A_{x;i,j}}{h_y}, \ (i,j) \in \mathrm{Sc} \cup \mathrm{Bl}; \end{aligned} \tag{2.20}$$

see Figure 2.1. The values of the link variables and the supercurrent are computed from the expressions

$$U_{x;i,j} = \mathrm{e}^{-i\kappa^{-1}h_x A_{x;i,j}}, \quad U_{y;i,j} = \mathrm{e}^{-i\kappa^{-1}h_y A_{y;i,j}}, \tag{2.21}$$

$$J_{x;i,j} = \frac{1}{\kappa h_x}\mathrm{Im}\left[\psi^*_{i,j} U_{x;i,j} \psi_{i+1,j}\right], \quad J_{y;i,j} = \frac{1}{\kappa h_y}\mathrm{Im}\left[\psi^*_{i,j} U_{y;i,j} \psi_{i,j+1}\right]. \tag{2.22}$$

The discretized TDGL equations are

$$\frac{\mathrm{d}\psi_{i,j}}{\mathrm{d}t} = \left(L_{xx}(U_{x;\cdot,j})\psi_{\cdot,j}\right)_i + \left(L_{yy}(U_{y;i,\cdot})\psi_{i,\cdot}\right)_j + N\left(\psi_{i,j}\right), \ (i,j) \in \mathrm{Sc}, \tag{2.23}$$

$$\sigma\,\frac{\mathrm{d}A_{x;i,j}}{\mathrm{d}t} = \left(D_{yy}A_{x;i,\cdot}\right)_j - \left(D_{yx}A_{y;\cdot,\cdot}\right)_{i,j} + J_{x;i,j}, \quad (i,j) \in \mathrm{Sc} \cup \mathrm{Bl}, \tag{2.24}$$

$$\sigma\,\frac{\mathrm{d}A_{y;i,j}}{\mathrm{d}t} = \left(D_{xx}A_{y;\cdot,j}\right)_i - \left(D_{xy}A_{x;\cdot,\cdot}\right)_{i,j} + J_{y;i,j}, \quad (i,j) \in \mathrm{Sc} \cup \mathrm{Bl}, \tag{2.25}$$

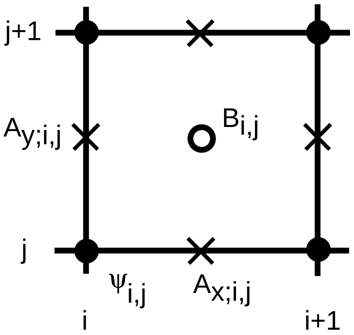


FIG. 2.1. *Computational cell with evaluation points for ψ, A_x, A_y, and B.*

where

$$(L_{xx}(U_{x;\cdot,j})\psi_{\cdot,j})_i = h_x^{-2}\left[U_{x;i,j}\psi_{i+1,j} - 2\psi_{i,j} + U^*_{x;i-1,j}\psi_{i-1,j}\right], \tag{2.26}$$

$$(L_{yy}(U_{y;i,\cdot})\psi_{i,\cdot})_j = h_y^{-2}\left[U_{y;i,j}\psi_{i,j+1} - 2\psi_{i,j} + U^*_{y;i,j-1}\psi_{i,j-1}\right], \tag{2.27}$$

$$N(\psi_{i,j}) = \tau_{i,j}\psi_{i,j} - |\psi_{i,j}|^2\psi_{i,j}, \tag{2.28}$$

$$(D_{yy}A_{x;i,\cdot})_j = h_y^{-2}\left[A_{x;i,j+1} - 2A_{x;i,j} + A_{x;i,j-1}\right], \tag{2.29}$$

$$(D_{xx}A_{y;\cdot,j})_i = h_x^{-2}\left[A_{y;i+1,j} - 2A_{y;i,j} + A_{y;i-1,j}\right], \tag{2.30}$$

$$(D_{yx}A_{y;\cdot,\cdot})_{i,j} = h_x^{-1}h_y^{-1}\left[(A_{y;i+1,j} - A_{y;i,j}) - (A_{y;i+1,j-1} - A_{y;i,j-1})\right], \tag{2.31}$$

$$(D_{xy}A_{x;\cdot,\cdot})_{i,j} = h_x^{-1}h_y^{-1}\left[(A_{x;i,j+1} - A_{x;i,j}) - (A_{x;i-1,j+1} - A_{x;i-1,j})\right]. \tag{2.32}$$

The interface conditions are

$$\psi_{n_{sx}-1,j} = U_{x;n_{sx}-1,j}\psi_{n_{sx},j}, \quad \psi_{n_{ex}+1,j} = U^*_{x;n_{ex},j}\psi_{n_{ex},j}, \quad j = 1,\dots,n_y. \tag{2.33}$$

At the outer boundary, B is given:

$$B_{0,j} = H_{L_j}, \quad B_{n_x,j} = H_{R_j}, \quad j = 1,\dots,n_y. \tag{2.34}$$

The resulting approximation is second-order accurate [10].

3. Time integration. We now address the integration of (2.23)–(2.25). The first equation, which controls the evolution of ψ, involves the second-order linear finite-difference operators L_{xx} and L_{yy}, whose coefficients depend on A_x and A_y, and the local nonlinear operator N, which involves neither A_x nor A_y. Each of the other two equations, which control the evolution of A_x and A_y respectively, involves likewise a second-order linear finite-difference operator, but with constant coefficients, and the nonlinear supercurrent operator, which involves ψ, A_x, and A_y. The following algorithms are distinguished by whether the various operators are treated explicitly or implicitly.

3.1. Fully explicit integration. Algorithm I uses a fully explicit forward Euler time-marching procedure for ψ, A_x, and A_y. Starting from an initial triple (ψ^0, A_x^0, A_y^0), we solve for $n = 0, 1, \dots,$

$$\frac{\psi^{n+1}_{i,j} - \psi^n_{i,j}}{\Delta t} = \left(L_{xx}(U^n_{x;\cdot,j})\psi^n_{\cdot,j}\right)_i + \left(L_{yy}(U^n_{y;i,\cdot})\psi^n_{i,\cdot}\right)_j + N\left(\psi^n_{i,j}\right), \ (i,j) \in \mathrm{Sc}, \tag{3.1}$$

$$(3.2)\ \sigma\,\frac{A^{n+1}_{x;i,j}-A^{n}_{x;i,j}}{\Delta t}=\left(D_{yy}A^{n}_{x;i,\cdot}\right)_j-\left(D_{yx}A^{n}_{y;\cdot,\cdot}\right)_{i,j}+J^{n}_{x;i,j},\quad (i,j)\in \mathrm{Sc}\cup\mathrm{Bl},$$

$$(3.3)\ \sigma\,\frac{A^{n+1}_{y;i,j}-A^{n}_{y;i,j}}{\Delta t}=\left(D_{xx}A^{n}_{y;\cdot,j}\right)_i-\left(D_{xy}A^{n}_{x;\cdot,\cdot}\right)_{i,j}+J^{n}_{y;i,j},\quad (i,j)\in \mathrm{Sc}\cup\mathrm{Bl},$$

where J^n is defined in terms of ψ^n, A^n_x, and A^n_y in the obvious way. The initial triple is usually chosen so the superconductor is in the Meissner state, with a seed present to trigger the transition to the vortex state.

Algorithm I has been described in [10]. It has been implemented in a distributed-memory multiprocessor environment (IBM SP2); the transformations necessary to achieve the parallelism have been described in [11]. The code uses the message passing interface (MPI) standard [12] as implemented in the MPICH software library [13] for domain decomposition, interprocessor communication, and file I/O. The code has been used extensively to study vortex dynamics in superconducting media [7, 8, 9]. The underlying algorithm provides highly accurate solutions but requires a significant number of time steps for equilibration. For stability reasons, the time step Δt cannot exceed 0.0025.

3.2. Semi-implicit integration. Algorithm II involves an implicit treatment of the second-order linear finite-difference operators D_{yy} and D_{xx} in the equations for A_x and A_y, respectively:

$$(3.4)\qquad \frac{\psi^{n+1}_{i,j}-\psi^{n}_{i,j}}{\Delta t}=\left(L_{xx}(U^{n}_{x;\cdot,j})\psi^{n}_{\cdot,j}\right)_i+\left(L_{yy}(U^{n}_{y;i,\cdot})\psi^{n}_{i,\cdot}\right)_j+N\left(\psi^{n}_{i,j}\right),\ (i,j)\in\mathrm{Sc},$$

$$(3.5)\ \sigma\,\frac{A^{n+1}_{x;i,j}-A^{n}_{x;i,j}}{\Delta t}=\left(D_{yy}A^{n+1}_{x;i,\cdot}\right)_j-\left(D_{yx}A^{n}_{y;\cdot,\cdot}\right)_{i,j}+J^{n}_{x;i,j},\quad (i,j)\in \mathrm{Sc}\cup\mathrm{Bl},$$

$$(3.6)\ \sigma\,\frac{A^{n+1}_{y;i,j}-A^{n}_{y;i,j}}{\Delta t}=\left(D_{xx}A^{n+1}_{y;\cdot,j}\right)_i-\left(D_{xy}A^{n}_{x;\cdot,\cdot}\right)_{i,j}+J^{n}_{y;i,j},\quad (i,j)\in \mathrm{Sc}\cup\mathrm{Bl}.$$

Equations (3.5) and (3.6) lead to two linear systems of equations,

$$(3.7)\qquad \left(I-\frac{\Delta t}{\sigma}D_{yy}\right)A^{n+1}_{x;i}=F_i(\psi^n,A^n_x,A^n_y),\qquad i=1,\dots,n_x-1,$$

$$(3.8)\qquad \left(I-\frac{\Delta t}{\sigma}D_{xx}\right)A^{n+1}_{y;j}=G_j(\psi^n,A^n_x,A^n_y),\qquad j=1,\dots,n_y,$$

for the vectors of unknowns $A_{x;i}=\{A_{x;i,j}:j=1,\dots,n_y\}$ and $A_{y;j}=\{A_{y;i,j}:i=1,\dots,n_x-1\}$. The matrix D_{yy} has dimension $n_y\times n_y$ and is periodic tridiagonal with elements $-h_y^{-2},2h_y^{-2},-h_y^{-2}$; the matrix D_{xx} has dimension $(n_x-1)\times(n_x-1)$ and is tridiagonal with elements $-h_x^{-2},2h_x^{-2},-h_x^{-2}$ (except along the edges because of the boundary conditions). Both matrices are independent of i and j. Furthermore, if the boundary conditions are time independent, they are constant throughout the time-stepping process. Hence, the coefficient matrices in (3.7) and (3.8) need to be factored only once; in fact, the factorization can be done in the preprocessing stage, and the factors can be stored.

In a parallel processing environment, the coefficient matrices extend over several processors, so (3.7) and (3.8) are broken up in blocks corresponding to the manner in which the computational mesh is distributed within the processor set. We first solve the equations within each processor (inner iterations) and then couple the solutions across processor boundaries (outer iterations). Hence, we deal with interprocessor coupling in an iterative fashion. Two to three inner iterations usually suffice to reach a desired tolerance for convergence. After each inner iteration, each processor shares boundary data with its neighbors through MPI calls.

3.3. Linearly implicit integration. Algorithm III combines the semi-implicit treatment of A_x and A_y with an implicit treatment of the order parameter:

$$\frac{\psi_{i,j}^{n+1} - \psi_{i,j}^{n}}{\Delta t} = \left(L_{xx}(U_{x;\cdot,j}^{n})\psi_{\cdot,j}^{n+1}\right)_i + \left(L_{yy}(U_{y;i,\cdot}^{n})\psi_{i,\cdot}^{n+1}\right)_j + N\left(\psi_{i,j}^{n}\right), \tag{3.9}$$

$$\sigma\,\frac{A_{x;i,j}^{n+1} - A_{x;i,j}^{n}}{\Delta t} = \left(D_{yy}A_{x;i,\cdot}^{n+1}\right)_j - \left(D_{yx}A_{y;\cdot,\cdot}^{n}\right)_{i,j} + J_{x;i,j}^{n}, \tag{3.10}$$

$$\sigma\,\frac{A_{y;i,j}^{n+1} - A_{y;i,j}^{n}}{\Delta t} = \left(D_{xx}A_{y;\cdot,j}^{n+1}\right)_i - \left(D_{xy}A_{x;\cdot,\cdot}^{n}\right)_{i,j} + J_{y;i,j}^{n}. \tag{3.11}$$

As before, the first equation holds for all $(i,j) \in \mathrm{Sc}$, while the second and third equations hold for all $(i,j) \in \mathrm{Sc} \cup \mathrm{Bl}$. The latter are solved as in the semi-implicit algorithm of the preceding section, while the former is solved by a method similar to the method of Douglas and Gunn [14] for the Laplacian.

We begin by transforming (3.9) into an equation for the correction matrix $\phi^{n+1} = \psi^{n+1} - \psi^{n}$. The equation has the general form

$$(I - \Delta t(L_{xx} + L_{yy}))\,\phi^{n+1} = F(\psi^n, A_x^n, A_y^n). \tag{3.12}$$

If Δt is sufficiently small, we may replace the operator in the left member by an approximate factorization,

$$(I - \Delta t(L_{xx} + L_{yy})) \approx (I - \Delta t L_{xx})\,(I - \Delta t L_{yy})\,, \tag{3.13}$$

and consider, instead of (3.12),

$$(I - \Delta t L_{xx})\,(I - \Delta t L_{yy})\,\phi^{n+1} = F(\psi^n, A_x^n, A_y^n). \tag{3.14}$$

This equation can be solved in two steps,

$$(I - \Delta t L_{xx})\,\varphi = F, \tag{3.15}$$

$$(I - \Delta t L_{yy})\,\phi^{n+1} = \varphi. \tag{3.16}$$

The conditions (2.33), which must be satisfied at the interface between the superconductor and the blanket material, require some care. If we impose the conditions at every time step, then

$$\begin{aligned}\phi_{n_{sx}-1,j}^{n+1} &= U_{x;n_{sx}-1,j}^{n+1}\phi_{n_{sx},j}^{n+1} + \left[U_{x;n_{sx}-1,j}^{n+1} - U_{x;n_{sx}-1,j}^{n}\right]\psi_{n_{sx},j}^{n},\\ \phi_{n_{ex}+1,j}^{n+1} &= \left(U_{x;n_{ex},j}^{n+1}\right)^{*}\phi_{n_{ex},j}^{n+1} + \left[\left(U_{x;n_{ex},j}^{n+1}\right)^{*} - \left(U_{x;n_{sx}-1,j}^{n}\right)^{*}\right]\psi_{n_{sx},j}^{n}\end{aligned}$$

for $j = 1, \dots, n_y$. These conditions couple the correction ϕ to the update of A_x. To eliminate this coupling, we solve (3.12) subject to the reduced interface conditions

$$\phi_{n_{sx}-1,j}^{n+1} = U_{x;n_{sx}-1,j}^{n+1}\phi_{n_{sx},j}^{n+1}, \qquad j = 1, \dots, n_y, \tag{3.17}$$

$$\phi_{n_{ex}+1,j}^{n+1} = \left(U_{x;n_{ex},j}^{n+1}\right)^{*}\phi_{n_{ex},j}^{n+1}, \qquad j = 1, \dots, n_y. \tag{3.18}$$

When (3.12) is replaced by (3.14), these conditions are inherited by the system (3.15).

3.4. Nonlinearly implicit integration. Algorithm IV uses a nonlinearly implicit integration procedure for the order parameter:

$$\frac{\psi_{i,j}^{n+1}-\psi_{i,j}^{n}}{\Delta t} = \left(L_{xx}(U_{x;\cdot,j}^{n})\psi_{\cdot,j}^{n+1}\right)_i + \left(L_{yy}(U_{y;i,\cdot}^{n})\psi_{i,\cdot}^{n+1}\right)_j + N\left(\psi_{i,j}^{n+1}\right), \tag{3.19}$$

$$\sigma\,\frac{A_{x;i,j}^{n+1}-A_{x;i,j}^{n}}{\Delta t} = \left(D_{yy}A_{x;i,\cdot}^{n+1}\right)_j - \left(D_{yx}A_{y;\cdot,\cdot}^{n}\right)_{i,j} + J_{x;i,j}^{n}, \tag{3.20}$$

$$\sigma\,\frac{A_{y;i,j}^{n+1}-A_{y;i,j}^{n}}{\Delta t} = \left(D_{xx}A_{y;\cdot,j}^{n+1}\right)_i - \left(D_{xy}A_{x;\cdot,\cdot}^{n}\right)_{i,j} + J_{y;i,j}^{n}. \tag{3.21}$$

The new element here is the term $N\left(\psi_{i,j}^{n+1}\right)$ in the first equation.

The second and third equations, with $(i,j)\in \mathrm{Sc}\cup\mathrm{Bl}$, are solved again as in the semi-implicit algorithm; the first, with $(i,j)\in\mathrm{Sc}$, is solved by a slight modification of the method used in the linearly implicit algorithm of the preceding section. The modification is brought about by the approximation

$$N\left(\psi^{n+1}\right) = \tau\psi^{n+1} - |\psi^{n+1}|^2\psi^{n+1} \approx \frac{1}{\Delta t}\left(S\left(\psi^{n}\right)-\psi^{n}\right), \tag{3.22}$$

where S is a nonlinear map:

$$S(\psi) = \frac{\tau^{1/2}\psi}{\left[|\psi|^2 + (\tau-|\psi|^2)\exp(-2\tau\Delta t)\right]^{1/2}}. \tag{3.23}$$

(This approximation is explained in the remark below.) Equation (3.19) is again of the form (3.12) but with a different right-hand side:

$$(I-\Delta t(L_{xx}+L_{yy}))\phi^{n+1} = G(\psi^n, A_x^n, A_y^n). \tag{3.24}$$

The difference is that, where F in (3.12) contains a term $(\Delta t)N\left(\psi^n\right)$, G in (3.24) contains the more complicated term $S\left(\psi^n\right)-\psi^n$.

Remark. The approximation (3.22) is suggested by semigroup theory. Symbolically,

$$N(\psi) = \lim_{\Delta t\to 0}\frac{S(\Delta t)\psi-\psi}{\Delta t}. \tag{3.25}$$

To find an expression for the "semigroup" S, we start from the continuous TDGL equations (2.6)–(2.8) (zero-electric potential gauge, $\Phi=0$), using the polar representation $\psi = |\psi|\mathrm{e}^{i\phi}$:

$$\partial_t|\psi| = \Delta|\psi| - |\psi||\nabla\phi-\kappa^{-1}\mathbf{A}|^2 + \tau|\psi| - |\psi|^3, \tag{3.26}$$

$$|\psi|\partial_t\phi = 2(\nabla|\psi|)\cdot(\nabla\phi-\kappa^{-1}\mathbf{A}) + |\psi|\nabla\cdot(\nabla\phi-\kappa^{-1}\mathbf{A}), \tag{3.27}$$

$$\sigma\partial_t\mathbf{A} = -\nabla\times\nabla\times\mathbf{A} + \kappa^{-1}|\psi|^2(\nabla\phi-\kappa^{-1}\mathbf{A}). \tag{3.28}$$

At this point, we are interested in the effect of the nonlinear term $|\psi|^3$ on the dynamics. To highlight this effect, we concentrate on the time evolution of the scalar $u=|\psi|$ and the vector $v=\nabla\phi-\kappa^{-1}\mathbf{A}$. (In physical terms, u^2 is the density of superconducting charge carriers, while u^2v is κ times the supercurrent density.) Ignoring their spatial variations, we have a dynamical system,

$$u' = -u|v|^2 + \tau u - u^3, \tag{3.29}$$

$$v' = -\varepsilon u^2 v, \tag{3.30}$$

where $'$ denotes differentiation with respect to t, and $\varepsilon = (\kappa^2\sigma)^{-1}$. This system yields a pair of ordinary differential equations for the scalars $x = u^2$ and $y = |v|^2$:

$$x' = 2x(\tau - x - y), \tag{3.31}$$

$$y' = -2\varepsilon xy. \tag{3.32}$$

If κ is large, ε is small, and the dynamics are readily analyzed. To leading order, y is constant; $y = 0$ is the only meaningful choice. (Recall that $xy^{1/2}$ is κ times the magnitude of the supercurrent density.) Then the dynamics of x are given by

$$x' = 2x(\tau - x). \tag{3.33}$$

We integrate this equation from $t = t_n$ to t:

$$x(t) = \frac{\tau x(t_n)}{x(t_n) + (\tau - x(t_n)) \exp(-2\tau(t - t_n))}. \tag{3.34}$$

In particular,

$$x(t_{n+1}) = \frac{\tau x(t_n)}{x(t_n) + (\tau - x(t_n)) \exp(-2\tau\Delta t)}, \tag{3.35}$$

where $\Delta t = t_{n+1} - t_n$. Since $x(t_n) = |\psi^n|^2$ and $x(t_{n+1}) = |\psi^{n+1}|^2$, it follows that

$$|\psi^{n+1}| = \frac{\tau^{1/2}|\psi^n|}{[|\psi^n|^2 + (\tau - |\psi^n|^2) \exp(-2\tau\Delta t)]^{1/2}}. \tag{3.36}$$

The phase ϕ of ψ is constant in time. If we multiply both sides by $\mathrm{e}^{i\phi}$, we obtain the expression (3.23) for the "semigroup" S.

4. Evaluation.

We now present the results of several experiments, where the algorithms described in the preceding section were applied to a benchmark problem.

4.1. Benchmark problem.

The benchmark problem adopted for this investigation is the equilibration of a vortex configuration in a homogeneous superconductor without defects ($\kappa = 16$, $\sigma = 1$, $\tau = 1$) embedded in a thin insulator (air), where the entire system is periodic in the direction of the free surfaces (y).

The superconductor measures 128ξ in the transverse (x) direction. The thickness of the insulating layer on either side is taken to be 2ξ, so the width of the entire system is 132ξ. The period in the y direction is taken to be 192ξ, so the entire system measures $132\xi \times 192\xi$.

The computational grid is uniform, with a mesh width $h_x = h_y = \frac{1}{2}\xi$. The periodic boundary conditions in the y direction are handled through ghost points, so the computational grid has 264×386 vertices. The index sets for the superconductor and blanket (see (2.16) and (2.17)) are

$$\mathrm{Sc} = \{(i, j) : i = 5, \ldots, 260,\ j = 1, \ldots, 386\}, \tag{4.1}$$

$$\mathrm{Bl} = \{(i, j) : i = 1, \ldots, 4, 261, \ldots, 264,\ \ j = 1, \ldots, 386\}. \tag{4.2}$$

The applied field is uniform:

$$H_L = H_R = H = 0.5. \tag{4.3}$$

(Units of H are $H_c\sqrt{2}$, so $H \approx 0.707\ldots H_c$.) As there is no transport current in the system, the solution of the TDGL equations tends to an equilibrium state.

4.2. Benchmark solution. First, preliminary runs were made to determine, for each algorithm, the optimal number of processors in a multiprocessing environment. Figure 4.1 shows the wall clock time for 50 time steps against the number of processors on the IBM SP2. Each algorithm shows a saturation around 16 processors, beyond which any improvement becomes marginal. All problems were subsequently run on 16 processors.

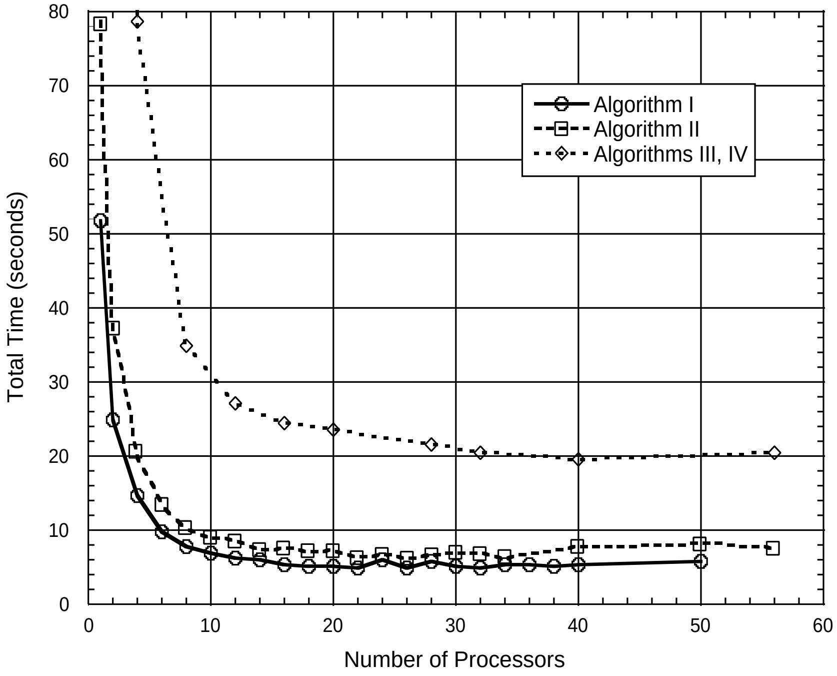


FIG. 4.1. *Elapsed time for* 50 *time steps as a function of the number of processors.*

Next, we used the fully explicit Algorithm I to establish a benchmark equilibrium configuration. We integrated (3.1)–(3.3) with a time step $\Delta t = 0.0025$ (units of ξ^2/D), the maximal value for which the algorithm remained stable, and followed the evolution of the vortex configuration by monitoring the number of vortices and their positions. Equilibrium was reached after 10,000,000 time steps, when the number of vortices remained constant and the vortex positions varied less than 1.0×10^{-6} (units of ξ). The equilibrium vortex configuration had 116 vortices arranged in a hexagonal pattern; see Figure 4.2. The wall clock time for the entire computation was approximately 3,000 minutes. The elapsed time per time step (0.018 seconds) is a measure for the computational cost of Algorithm I.

4.3. Evaluation of Algorithms II–IV. With the benchmark solution in place, we evaluated each of the remaining Algorithms (II–IV) for stability, accuracy, and computational cost.

We found the stability limit in the obvious way, gradually increasing the time step and integrating to equilibrium until arithmetic divergences caused the algorithm to fail. Equilibrium was defined by the same criteria as for the benchmark solution: no change in the number of vortices and a variation in the vortex positions of less than 1.0×10^{-6}.

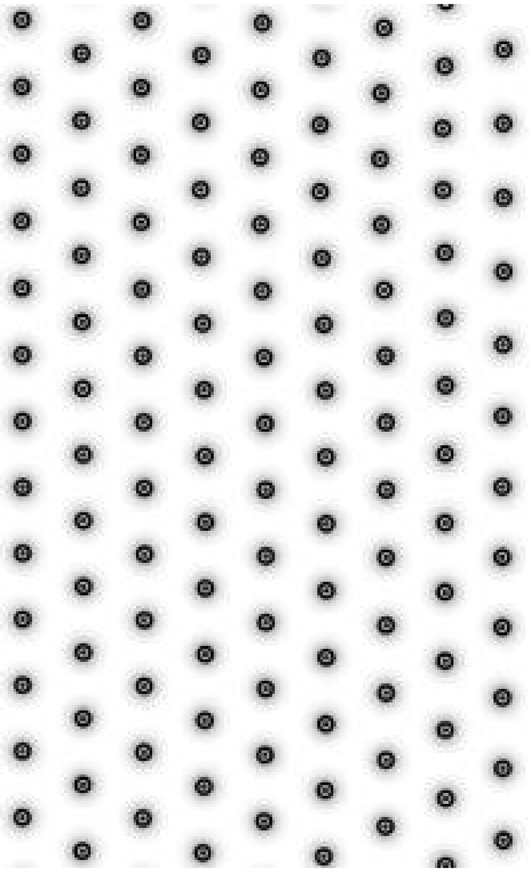


FIG. 4.2. *Equilibrium vortex configuration for the benchmark problem.*

Because each algorithm defines its own path through phase space, one can expect to find neither identical equilibrium configurations nor equilibrium configurations that are exactly the same as the benchmark. The equilibrium vortex configurations for the four algorithms were indeed different, albeit slightly. To measure the differences quantitatively, we computed the following three parameters: (i) the number of vortices in the superconducting region, (ii) the mean bond length joining neighboring pairs of vortices, and (iii) the mean bond angle subtended by neighboring bonds throughout the vortex lattice. In all cases, the number of vortices was the same (116); the mean bond length varied less than $1.0 \times 10^{-3}\xi$, and the mean bond angle varied by less than 1.0×10^{-3} radians. Within these tolerances, the equilibrium vortex configurations were the same.

The results are given in Table 4.1; Δt is the time step at the stability limit (units of ξ^2/D), N the number of time steps needed to reach equilibrium, and C the cost of the algorithm (seconds per time step). From these data we obtain the wall clock time needed to compute the equilibrium configuration, $T = NC/60$ (minutes).

Note that the existence of a stability limit for Algorithm IV is a consequence of the implementation in a multiprocessing environment. Since we restrict interprocessor communication to the end of each time step, the implicit character of the algorithm is lost. On a single processor, Algorithm IV is implicit, and the stability limit is infinite.

TABLE 4.1
Performance data for Algorithms I–IV.

Algorithm	Δt	N	C	T
I	0.0025	10,000,000	0.018	3,000
II	0.0500	500,000	0.103	858
III	0.1000	250,000	0.232	967
IV	0.1900	100,000	0.233	388

5. Further evaluation of Algorithm IV.

We evaluated the nonlinearly implicit Algorithm IV in more detail by considering its speedup in a multiprocessing environment and its performance under a multitimestepping procedure.

5.1. Parallelism.

First, we investigated the speedup of Algorithm IV in a multiprocessing environment, using the benchmark problem and two other problems, obtained from the benchmark problem by twice doubling the size of the system in each direction. The mesh width was kept constant at $\frac{1}{2}\xi$, so the resulting computational grid had 264 × 386 vertices for the benchmark problem, 528 × 772 vertices for the intermediate problem, and 1056 × 1544 vertices for the largest problem. Speedup was defined as the ratio of the wall clock time (exclusive of I/O) to reach equilibrium on p processors divided by the time to reach equilibrium on a single processor for the benchmark and intermediate problem, or twice the time to reach equilibrium on two processors for the largest problem. (The largest problem did not fit on a single processor.) The results are given in Figure 5.1. The curve for the benchmark problem was obtained as an average over many runs; the data for the intermediate and largest problem were obtained from single runs, so they are less smooth. The speedup is clearly linear when the number of processors is small; it becomes sublinear at about 12 processors for the smallest problem, 14 processors for the intermediate problem, and 18 processors for the largest problem.

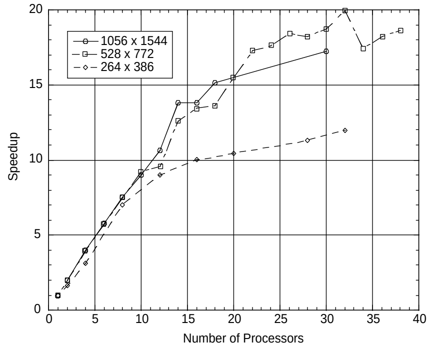


FIG. 5.1. *Performance of Algorithm* IV *in a multiprocessing environment.*

5.2. Multitimestepping.

The final set of experiments shows that the performance of Algorithm IV is enhanced by a multitimestepping procedure, where $\mathbf{A}$ is updated less frequently than ψ:

$$\frac{\psi_{i,j}^{n+1}-\psi_{i,j}^{n}}{\Delta t} = \left(L_{xx}(U_{x;\cdot,j}^{n})\psi_{\cdot,j}^{n+1}\right)_i + \left(L_{yy}(U_{y;i,\cdot}^{n})\psi_{i,\cdot}^{n+1}\right)_j + N\left(\psi_{i,j}^{n+1}\right), \tag{5.1}$$

$$\sigma\,\frac{A_{x;i,j}^{n+m}-A_{x;i,j}^{n}}{m\Delta t} = \left(D_{yy}A_{x;i,\cdot}^{n+m}\right)_j - \left(D_{yx}A_{y;\cdot,\cdot}^{n}\right)_{i,j} + J_{x;i,j}^{n}, \tag{5.2}$$

Table 5.1
Effect of update frequency on time and cost of Algorithm IV.

m	N	C	T
1	100,000	0.2330	388
10	200,000	0.0707	236
15	250,000	0.0646	270

$$\sigma \frac{A^{n+m}_{y;i,j} - A^{n}_{y;i,j}}{m\Delta t} = \left(D_{xx}A^{n+m}_{y;\cdot,j}\right)_i - \left(D_{xy}A^{n}_{x;\cdot,\cdot}\right)_{i,j} + J^{n}_{y;i,j}. \tag{5.3}$$

When $m = 1$, both ψ and $\mathbf{A}$ are updated at every time step, but when m is greater than 1, $\mathbf{A}$ is updated only every mth time step. In the limit as $m \to \infty$, this procedure yields the frozen-field approximation, which is a good approximation of the GL model near the upper critical field when the charge of the superconducting charge carriers is small [15].

We applied this modification of Algorithm IV with $m = 10, 15$ to the benchmark problem of section 4. The results are given in Table 5.1. All computations were done with $\Delta t = 0.19$ (units of ξ^2/D). The data for the computation with $m = 1$ are taken from Table 4.1. We observe that the cost of the algorithm (C, seconds per time step) decreases with increasing m, while the number of time steps to equilibrium (N) increases. If $m = 10$, the overall wall clock time ($T = NC/60$, minutes) is approximately one-third less than the wall clock time for $m = 1$. For larger values of m, the increase in the number of steps needed to reach equilibrium offsets any gain from the decrease in cost. These data suggest an optimal strategy, where $\mathbf{A}$ is updated every 10 time steps.

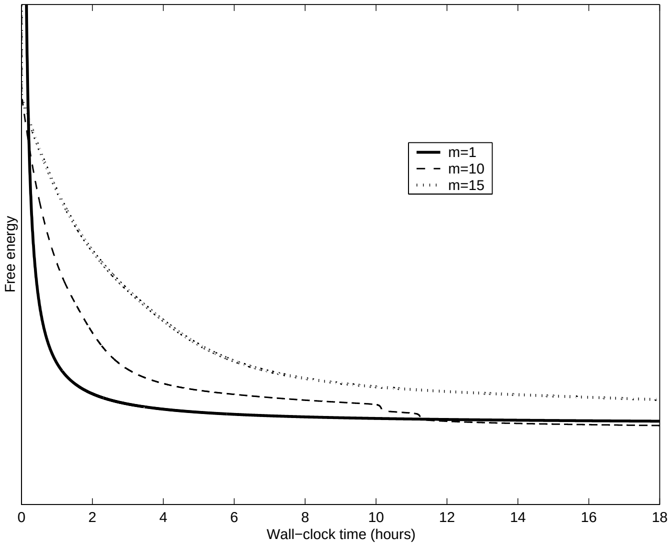


Fig. 5.2. *Effect of the updating frequency on the evolution of the energy functional.*

Figure 5.2 shows the effect of the updating frequency on the evolution of the free-energy functional (2.9). Note the dramatic increase of the wall-clock time for $m = 15$.Eventually, the curve for $m = 15$ merges with the other curves, but this happens well beyond the range of the figure.

6. Conclusions.

In summary, we present the following conclusions:

(i) One can increase the time step Δt nearly 80-fold, without losing stability, by going from the fully explicit Algorithm I to the nonlinearly implicit Algorithm IV.

(ii) As one goes to the nonlinearly implicit Algorithm IV, the complexity of the matrix calculations and, hence, the cost C of a single time step increase.

(iii) The increase in the cost C per time step is more than offset by the increase in the size of the time step Δt. In fact, the wall clock time needed to compute the same equilibrium state with the nonlinearly implicit Algorithm IV is one-sixth of the wall clock time for the fully explicit Algorithm I.

(iv) The (physical) time to reach equilibrium—that is, $N\Delta t$, the number of time steps needed to reach equilibrium times the step size—is (approximately) the same for all algorithms, namely, 25,000 (units of ξ^2/D).

(v) The nonlinearly implicit Algorithm IV displays linear speedup in a multiprocessing environment. The speedup curves show sublinear behavior when the number of processors is large.

(vi) The performance of the nonlinearly implicit Algorithm IV can be improved further by a multitimestepping procedure, where the vector potential $\mathbf{A}$ is updated less frequently than the order parameter ψ.

REFERENCES

[1] V. L. Ginzburg and L. D. Landau, *On the theory of superconductivity*, Zh. Eksp. Teor. Fiz. (USSR) 20 (1950), pp. 1064–1082. English translation in Men of Physics: L.D. Landau, Vol. 1, D. ter Haar, ed., Pergamon Press, Oxford, 1965, pp. 138–167 .

[2] M. Tinkham, *Introduction to Superconductivity* 2nd ed., McGraw-Hill, New York, 1996.

[3] A. Schmid, *A time dependent Ginzburg–Landau equation and its application to a problem of resistivity in the mixed state*, Phys. kondens. Materie, 5 (1966), pp. 302–317.

[4] L. P. Gor'kov and G. M. Éliashberg, *Generalizations of the Ginzburg–Landau equations for non-stationary problems in the case of alloys with paramagnetic impurities*, Zh. Eksp. Teor. Fiz., 54 (1968), pp. 612–626; Soviet Phys. JETP, 27 (1968), pp. 328–334.

[5] Q. Du, M. D. Gunzburger, and J. S. Peterson, *Analysis and approximation of the Ginzburg–Landau model of superconductivity*, SIAM Rev., 34 (1992), pp. 54–81.

[6] J. Fleckinger-Pellé, H. G. Kaper, and P. Takáč, *Dynamics of the Ginzburg-Landau Equations of Superconductivity*, Nonlinear Anal., 32 (1998), pp. 647–665.

[7] D. W. Braun, G. W. Crabtree, H. G. Kaper, A. E. Koshelev , G. K. Leaf, D. M. Levine, and V. M. Vinokur, *Structure of a moving vortex lattice*, Phys. Rev. Lett., 76 (1996), pp. 831–834.

[8] G. W. Crabtree, G. K. Leaf, H. G. Kaper, V. M. Vinokur, A. E. Koshelev, D. W. Braun, D. M. Levine, W. K. Kwok, and J. A. Fendrich, *Time-dependent Ginzburg-Landau simulations of vortex guidance by twin boundaries*, Physica C, 263 (1996), pp. 401–408.

[9] G. C. Crabtree, D. O. Gunter, H. G. Kaper, A. E. Koshelev, G. K. Leaf, and V. Vinokur, *Numerical simulations of driven vortex systems*, Phys. Rev. B, 61 (2000), pp. 1446–1455.

[10] W. D. Gropp, H. G. Kaper, G. K. Leaf, D. M. Levine, M. Palumbo, and V. M. Vinokur, *Numerical simulations of vortex dynamics in type*-II *superconductors*, J. Comput. Phys., 123 (1996), pp. 254–266.

[11] N. Galbreath, W. Gropp, D. Gunter, G. Leaf, and D. Levine, *Parallel solution of the three-dimensional, time-dependent Ginzburg-Landau equation*, in Proceedings of the Sixth SIAM Conference on Parallel Processing for Scientific Computing, R. F. Sincovec, D. E. Keyes, M. R. Leuze, L. R. Petzold, and D. A. Reed, eds., SIAM, Philadelphia, 1993, pp. 160–164.

[12] W. Gropp, E. Lusk, and A. Skjellum, *Using MPI*, 2nd ed., MIT Press, Cambridge, MA, 1999.

[13] W. Gropp, E. Lusk, N. Doss, and A. Skjellum, *A High-Performance, Portable Implementation of the MPI Message Passing Interface Standard*, Technical Report ANL/MCS-P567-0296, Argonne National Laboratory, Argonne, IL, 1996.

[14] J. Douglas and J. E. Gunn, *A general formulation of alternating direction methods—Part* I*: Parabolic and hyperbolic problems*, Numer. Math., 6 (1964), pp. 428–453.

[15] H. G. Kaper and H. Nordborg, *The frozen-field approximation and the Ginzburg–Landau equations of superconductivity*, J. Engrg. Math., 39 (2001), 221–240.

SIAM J. Sci. Comput.
Vol. 23, No. 6, pp. 1959–1980

A DOMAIN DECOMPOSITION METHOD FOR ADVECTION-DIFFUSION PROCESSES WITH APPLICATION TO BLOOD SOLUTES[*]

ALFIO QUARTERONI[†], ALESSANDRO VENEZIANI[‡], AND PAOLO ZUNINO[§]

Abstract. In the present paper we consider a heterogeneous model for the dynamics of a blood solute both in the vascular lumen and inside the arterial wall. In the lumen, we consider an advection-diffusion equation, where the convective field is provided by the velocity of blood, which is in turn obtained by solving the Navier–Stokes equations. Inside the arterial wall we consider a pure diffusive dynamics. Since the endothelial layer at the interface between the lumen and the wall acts as a permeable membrane, whose permeability depends on the shear rate exerted by the blood, the solute concentration is discontinuous across this membrane. A possible approach for the numerical study of this kind of problem is inspired by domain decomposition techniques. In particular, we introduce a splitting in the computation and alternate the solution of the advection-diffusion equation in the lumen with that of the diffusion equation in the wall. We set up an efficient iterative method, based on a suitable reformulation of the problem in terms of a Steklov–Poincaré interface equation. This formulation is a nonstandard one because of the concentration discontinuity at the lumen-wall interface and plays a key role in the proof of convergence of our method. In particular, we prove that the convergence rate performed by the proposed method is independent of the finite element space discretization and provides a criterion for the selection of an acceleration parameter.

Several numerical results, referred to as biomedical applications, support our theoretical conclusions and illustrate the efficiency of this algorithm.

Key words. heterogeneous models, domain decomposition techniques, Steklov–Poincaré operators, advection-diffusion equations, finite elements

AMS subject classifications. 76D05, 76Z05, 76R50, 76M10, 65M60, 65Y99

PII. S1064827500375722

1. Introduction.

We consider the numerical treatment of a model for the dynamics of blood solutes which was introduced in [14]. This model is based on an advection-diffusion equation describing the solute dynamics in the vascular lumen, the convective field being provided by the blood velocity. This equation is coupled with a pure diffusive model accounting for the solute dynamics inside the arterial wall, where convection is negligible. The two subdomains (namely the lumen and the wall) are physically separated by the endothelial layer, which acts as a selective permeable membrane. The interface equation matching the two subproblems in fact follows from the specific nature of this membrane. It is worthwhile noticing that, due to the presence of this membrane, the two concentrations fail to match continuously at the interface.

[*] Received by the editors July 25, 2000; accepted for publication (in revised form) November 1, 2001; published electronically February 27, 2002. This work was supported by MURST research contracts and the GNIM-INDAM project "Modelling of Mass Transport Processes in Heterogeneous Media."

http://www.siam.org/journals/sisc/23-6/37572.html

[†] Dipartimento di Matematica "F. Brioschi," Politecnico di Milano, Piazza L. da Vinci 32, I-20133 Milan, Italy and Département de Mathématiques, École Polytechnique Fédérale de Lausanne, CH-1015, Lausanne, Switzerland (Alfio.Quarteroni@epfl.ch).

[‡] Dipartimento di Matematica "F. Brioschi," Politecnico di Milano, Piazza L. da Vinci 32, I-20133 Milan, Italy (ales@mate.polimi.it).

[§] Département de Mathématiques, École Polytechnique Fédérale de Lausanne, CH-1015, Lausanne, Switzerland (Paolo.Zunino@epfl.ch).

The well posedness of this model, coupled with the Navier–Stokes equations for the description of the blood velocity and pressure fields, has been analyzed in [14]. The interest for this problem stems from the consideration that, in some cases, the cause of widespread pathologies of the vascular system has been related to specific features of the blood flow in a diseased district as well as to the influence of the flow pattern on the transfer processes of solutes between blood and walls (see, e.g., [1], [7], [15], [2], [3]). In [2], [3], for example, a more complex physiological model has been introduced; however, so far it has been tested only on simple two-dimensional (2D) geometries. On the other hand, more realistic three-dimensional (3D) calculations have been pursued in [7]; however, in this case the model does not include the arterial wall as a domain, per se; rather it surrogates the artery wall by prescribing a Dirichlet boundary condition for the concentration. In this paper, our aim is to set up efficient *numerical methods* for solving heterogeneous problems featuring a discontinuous solution. Although stemming from the specific application at hand, the interest of these methods goes beyond it as they can be applied to any system of advection-diffusion equations which models the transfer of mass between heterogeneous media through permeable membranes.

In the numerical analysis of these kinds of problems, a monolithic solver is typically adopted. Our approach is based on domain decomposition methods, solving alternatively the different problems in the different subdomains (*iterative substructuring approach*). In such a way, the discontinuity at the interface is accounted for naturally. The general theory underlying this approach has been developed in [13, Chapters 1 and 4]. However, the discontinuity of the solution in this specific application makes the convergence analysis of the method "nonstandard." In particular, we will focus on the choice of convenient preconditioned iterative techniques and the proof of convergence in the very general and abstract framework provided by the Steklov–Poincaré operator theory. In a forthcoming paper, we will consider the extension of these techniques to the more realistic models considered in [2], [3].

The paper is organized as follows. In section 2 we provide a brief introduction to the heterogeneous model and recall the well posedness results obtained in [14]. In section 3 we provide a reinterpretation of the problem in terms of a Steklov–Poincaré interface equation. This reformulation is not a mere extension of techniques adopted in other contexts, due to the presence of discontinuous solutions. In section 4, we introduce an iterative method for the solution of the problem by solving successively the solute dynamics in the lumen and in the wall. Starting from the Steklov–Poincaré reformulation of the interface problem, we show that this method can be regarded as a particular Richardson preconditioned method for the interface problem. Then, we carry out the convergence analysis of this method; we identify an optimal preconditioner associated to it and propose various generalizations based on the (flexible) preconditioned GMRES method.

The numerical results presented refer to biomedical applications and illustrate the efficiency of our schemes (section 5).

1.1. Some notation. Let $\Omega \subset \mathbb{R}^d$ $(d = 2, 3)$ be a physical bounded domain and $\mathbf{x} \in \Omega$. We denote by $L^2(\Omega)$ the Hilbert space of square integrable functions in Ω. The scalar product in $L^2(\Omega)$ is denoted by $(\cdot,\cdot)$ and the related norm by $\|\cdot\|_{L^2(\Omega)}$. The space of essentially bounded functions in Ω is denoted by $L^\infty(\Omega)$. The Sobolev space of functions, whose first (distributional) derivatives belong to $L^2(\Omega)$, is denoted by $H^1(\Omega)$ and its norm by $\|\cdot\|_{H^1(\Omega)}$.

If $\Sigma \subset \partial\Omega$ is open and nonempty, then the space of functions defined on Σ which are traces of functions belonging to $H^1(\Omega)$ is indicated by $H^{1/2}(\Sigma)$. We recall (see [5]) that the trace operator $\gamma : H^1(\Omega) \to H^{1/2}(\Sigma)$ is surjective and continuous and there exists an injective, linear, and continuous map $\mathcal{L} : H^{1/2}(\Sigma) \to H^1(\Omega)$ called *lifting* such that $\lambda = \gamma\mathcal{L}\lambda$ for all $\lambda \in H^{1/2}(\Sigma)$. In particular, denoting by ϕ a function in $H^1(\Omega)$ and $\gamma\phi$ its trace on Σ, the following *trace inequality* holds:

$$\exists\beta_t > 0 : \quad \|\gamma\phi\|_{H^{1/2}(\Sigma)} \leq \beta_t\|\phi\|_{H^1(\Omega)} \quad \forall\phi \in H^{1/2}(\Sigma). \tag{1.1}$$

We denote by $H^1_\Sigma(\Omega)$ the subspace of $H^1(\Omega)$ of the functions that have null traces on Σ. In particular, we adopt the usual notation $H^1_0(\Omega) = H^1_{\partial\Omega}(\Omega)$. In $H^1_\Sigma(\Omega)$ the following Poincaré inequality holds:

$$\exists\alpha > 0 : \quad \|\phi\|_{L^2(\Omega)} \leq \alpha\|\nabla\phi\|_{L^2(\Omega)} \quad \forall\phi \in H^1_\Sigma(\Omega). \tag{1.2}$$

If Γ denotes a $(d-1)$-dimensional manifold in $\overline{\Omega}$ with $\Gamma \cap \Sigma \neq \emptyset$, the trace of $u \in H^1_\Sigma(\Omega)$ on Γ belongs to a subspace of $H^{1/2}(\Gamma)$, usually denoted by $H^{1/2}_{00}(\Gamma)$. To simplify our notation we will set $\Lambda = H^{1/2}_{00}(\Gamma)$, and Λ' will denote its dual (see [5]). We remark that if ζ is any positive function in $L^2(\Sigma)$, the following definition makes sense:

$$(\rho, \lambda)_\zeta = \int_\Sigma \zeta\rho\lambda = \left(\sqrt{\zeta}\lambda, \sqrt{\zeta}\rho\right) \quad \forall\lambda, \rho \in H^{1/2}(\Sigma). \tag{1.3}$$

Indeed, by the Sobolev embedding theorem, λ and ρ belong to $L^4(\Sigma)$; hence $\lambda\rho$ belongs to $L^2(\Sigma)$. Denoting by $\mathcal{L}\lambda$ and $\mathcal{L}\rho$ any continuous lifting of λ and ρ from Σ to Ω, it follows that

$$\left|(\rho, \lambda)_\zeta\right| \leq \beta_e^2\|\zeta\|_{L^2(\Sigma)}\|\lambda\|_{H^{1/2}(\Sigma)}\|\rho\|_{H^{1/2}(\Sigma)} \leq \beta^2\|\mathcal{L}\lambda\|_{H^1(\Omega)}\|\mathcal{L}\rho\|_{H^1(\Omega)}, \tag{1.4}$$

with $\beta^2 = \beta_e^2\beta_t^2\|\zeta\|_{L^2(\Sigma)}$, where β_e is the embedding constant of $H^{1/2}(\Sigma)$ in $L^4(\Sigma)$ and β_t is the constant of the trace inequality (1.1). Finally, we set

$$\|\lambda\|^2_\zeta = (\lambda, \lambda)_\zeta = \|\sqrt{\zeta}\lambda\|^2_{L^2(\Sigma)}.$$

For space-time functions $v : \Omega \times (0,T) \to \mathbb{R}$, for all real q and $s = 0, 1$, we introduce the space

$$L^q(0,T; H^s(\Omega)) \equiv \left\{ v : (0,T) \to H^s |\ v(t) \text{ is measurable}, \int_0^T \|v(t)\|^q_{H^s(\Omega)}dt < \infty \right\}$$

endowed with the norm

$$\|v\|_{L^q(0,T;H^s(\Omega))} \equiv \left(\int_0^T \|v(t)\|^q_{H^s(\Omega)}dt\right)^{1/q}.$$

2. Problem formulation. Let us consider a specific vascular district $\Omega \subset \mathbb{R}^d$ $(d = 2, 3)$, composed by a lumen or a fluid subdomain Ω_f and a structure or wall subdomain Ω_w. Their interface Γ belongs to $\mathbb{R}^{d-1}$ (see Figure 2.1).

The artificial sections delimiting the district proximally and distally with respect to the heart will be denoted by Γ_{up} and Γ_{dw}, respectively. For $\mathbf{x} \in \Omega$ and $t > 0$ we denote by $\mathbf{u}(\mathbf{x}, t)$ the velocity of the blood and by $P(\mathbf{x}, t)$ its pressure. $C_f(\mathbf{x}, t)$ and $C_w(\mathbf{x}, t)$ denote the concentrations of the solute in the lumen Ω_f and in the wall Ω_w, respectively. We assume the blood to be an incompressible Newtonian fluid (which is a realistic assumption in large and medium vessels—see, e.g., [11]) within rigid walls. Then, the blood motion is described by the Navier–Stokes incompressible equations, obtained by the momentum and mass conservation principles. The initial-boundary values problem we are going to consider for the blood dynamics therefore reads

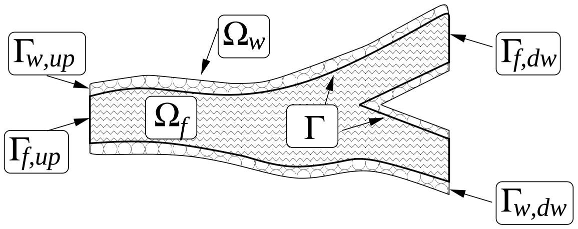


FIG. 2.1. *Computational domain representing a 2D section of a vascular district featuring the lumen Ω_f and the wall Ω_w.*

$$\begin{aligned}
&\rho\frac{\partial \mathbf{u}}{\partial t}+\rho\left(\mathbf{u}\cdot\nabla\right)\mathbf{u}-\nu\Delta\mathbf{u}+\nabla P=\mathbf{f}\quad \mathbf{x}\in\Omega_f,\qquad t>0,\\
&\nabla\cdot\mathbf{u}=0\quad \mathbf{x}\in\Omega_f,\qquad t>0,\\
&\mathbf{u}=\mathbf{b}\quad\text{on}\quad\Gamma_{f,up},\quad t>0,\quad \mathbf{u}=\mathbf{0}\quad\text{on}\quad\Gamma,\quad t>0,\\
&P\mathbf{n}-\rho\nu\nabla\mathbf{u}\cdot\mathbf{n}=P_{ext}\mathbf{n}\quad\text{on}\quad\Gamma_{f,dw},\qquad t>0,\\
&\mathbf{u}\left(\mathbf{x},t\right)=\mathbf{u}_0\quad\text{with}\quad\nabla\cdot\mathbf{u}_0=0,\quad\mathbf{x}\in\Omega_f,\quad t=0.
\end{aligned}\tag{2.1}$$

In (2.1) we suppose that the boundary $\partial\Omega_f$ of the computational domain of the fluid part of the problem is split into the interface with the wall Γ, the upstream or proximal part $\Gamma_{f,up}$ (where we prescribe the velocity field), and the downstream or distal one, $\Gamma_{f,dw}$ (where the normal fluid stress is assigned). We are also assuming that the density of the blood ρ_b as well as the viscosity ν are constant; the physiological range of these parameters is discussed in section 5.

The dynamics of solutes is described by an advection-diffusion process. In the lumen, the convective field is provided by the blood velocity, while in the wall, because of the very low velocity of the solvent, the advection is negligible (see [8]). The interface Γ can be regarded as a permeable membrane whose permeability ζ is a positive function of the shear stress σ exerted by the blood on the wall (see [15]). More precisely, the solute flux through Γ is proportional to the difference of concentration between lumen and wall. All these considerations lead to the following model for the solute concentrations for all $t>0$:

$$\left\{\begin{aligned}
&\frac{\partial C_f}{\partial t}-\mu_f\Delta C_f+\mathbf{u}\cdot\nabla C_f=s_f\quad\text{in}\quad\Omega_f,\\
&\frac{\partial C_w}{\partial t}-\mu_w\Delta C_w=s_w\quad\text{in}\quad\Omega_w,\\
&\mu_f\frac{\partial C_f}{\partial\mathbf{n}_f}+\zeta\left(C_f-C_w\right)=0\quad\text{on}\quad\Gamma,\\
&\mu_w\frac{\partial C_w}{\partial\mathbf{n}_w}+\zeta\left(C_w-C_f\right)=0\quad\text{on}\quad\Gamma,\\
&C_f=C_{f,up}\quad\text{on}\quad\Gamma_{f,up},\quad C_w=C_{w,up}\quad\text{on}\quad\Gamma_{w,up},\\
&\mu_f\frac{\partial C_f}{\partial\mathbf{n}_f}=0\quad\text{on}\quad\Gamma_{f,dw},\quad\mu_w\frac{\partial C_w}{\partial\mathbf{n}_w}=0\quad\text{on}\quad\Gamma_{w,dw}.
\end{aligned}\right.\tag{2.2}$$

To complete (2.2), we add the initial conditions

$$C_f(\mathbf{x},t)=C_{f0}(\mathbf{x}),\quad\mathbf{x}\in\Omega_f,\qquad C_w(\mathbf{x},t)=C_{w0}(\mathbf{x}),\quad\mathbf{x}\in\Omega_w,\quad t=0,$$

where the diffusivity coefficients μ_f and μ_w are positive and constant and $s_f(\mathbf{x},t)$ and $s_w(\mathbf{x},t)$ are possible source terms. In (2.2), we have denoted by $\Gamma_{w,up}$ ($\Gamma_{w,dw}$) the part of the wall corresponding to the proximal (distal) section of the fluid domain (see Figure 2.1).

In this model, observe that the solute is regarded as a *passive scalar*; that means that it is simply convected by the blood in the lumen, neglecting any possible feedback on the hemodynamics. In particular, viscosity and density of blood are assumed independent of the solute concentration. This hypothesis is actually coherent with the models for solute dynamics proposed in [2], [3].

Now, observe that $(2.2)_4$ can be equivalently substituted by the equation

$$\mu_f \frac{\partial C_f}{\partial \mathbf{n}_f} = -\mu_w \frac{\partial C_w}{\partial \mathbf{n}_w} \quad \text{on } \Gamma,$$

prescribing the continuity of the solute flux between the fluid and the wall domains. However, we prefer the formulation (2.2), as it leads to a more efficient subdomain iterative scheme.

REMARK 2.1. *For some gaseous solutes, the diffusivity coefficient μ_f depends on the rate of deformation of blood (see* [4]*). This feature makes the mathematical analysis of the coupled Navier–Stokes/solute dynamics problem more involved, as addressed in* [14]*. However, it does not bring significant differences in the context of the present work, which is mainly focused on the numerical approximation.*

In order to carry out the mathematical analysis of the problem given by (2.1) and (2.2), as well as its numerical discretization, we resort to their "weak" or variational formulations. Concerning the Navier–Stokes equations, this can be done in a very standard way and we refer to, e.g., [12], [17]. For the advection-diffusion problem we introduce the following notation. For all $\psi_f, \phi_f \in H^1_{\partial\Omega_f\setminus\Gamma}(\Omega_f)$ and $\psi_w, \phi_w \in H^1_{\partial\Omega_w\setminus\Gamma}(\Omega_w)$ set

$$a_f(\psi_f,\phi_f) = \mu_f(\nabla\psi_f,\nabla\phi_f) + ((\mathbf{u}\cdot\nabla)\psi_f,\phi_f), \tag{2.3}$$

$$a_w(\psi_w,\phi_w) = \mu_w(\nabla\psi_w,\nabla\phi_w). \tag{2.4}$$

Both $a_f(\cdot,\cdot)$ and $a_w(\cdot,\cdot)$ are continuous and coercive bilinear forms; additionally, $a_w(\cdot,\cdot)$ is *symmetric*. Then, the weak formulation of problem (2.2) reads as follows.

PROBLEM 2.1. *Given the initial condition $C_f(\mathbf{x},t=0) = C_{f,0}, \in H^1_{\partial\Omega_f\setminus\Gamma}(\Omega_f)$ and $C_w(\mathbf{x},t=0) = C_{w,0} \in H^1_{\partial\Omega_w\setminus\Gamma}(\Omega_w)$ find $C_f \in L^2(0,T;H^1_{\partial\Omega_f\setminus\Gamma}(\Omega_f))$, $C_w \in L^2(0,T;H^1_{\partial\Omega_w\setminus\Gamma}(\Omega_w))$ such that for all $\phi_f \in H^1_{\partial\Omega_f\setminus\Gamma}(\Omega_f)$ and $\phi_w \in H^1_{\partial\Omega_w\setminus\Gamma}(\Omega_w)$*

$$\begin{cases} \left(\dfrac{\partial C_f}{\partial t},\phi_f\right) + a_f(C_f,\phi_f) + (C_f - C_w,\phi_f)_\zeta = (s_f,\phi_f), \\ \left(\dfrac{\partial C_w}{\partial t},\phi_w\right) + a_w(C_w,\phi_w) + (C_w - C_f,\phi_w)_\zeta = (s_w,\phi_w). \end{cases} \tag{2.5}$$

If the domain and the data are regular enough, Problem 2.1, coupled with the (weak form of) Navier–Stokes one, is well posed. More precisely, the following proposition can be shown (see [14]).

PROPOSITION 2.1. *If Ω_f is a 2D domain smooth enough, then the coupled blood solute dynamics problem admits a unique solution, depending continuously on the* *data. If Ω_f is a 3D domain, the same conclusion holds true, provided the initial data of the Navier–Stokes problem are sufficiently small.*

Concerning the result of this proposition, we point out that the smallness of the data for the Navier–Stokes problem is actually a standard and an unavoidable hypothesis for ensuring the existence of a convective field in any problem involving an incompressible fluid. Indeed, no unconditional well posedness result is so far available for the Navier–Stokes equations (see [17], [12]). Moreover, observe that if the diffusivity of the solute in the lumen is the function of the shear rate (see Remark 2.1) the smallness of the initial data for the Navier–Stokes problem is necessary even for the 2D case.

We point out that in [14] only full Dirichlet boundary conditions for the blood were considered, which yield a coercive bilinear form. However, in the current case, a standard energy argument allows us to prove that the bilinear form is weakly coercive, which is still sufficient to ensure the well posedness of the associated parabolic problem (see [5], [6], [12]).

3. Discretization and Steklov–Poincaré operators.

3.1. The semidiscrete problem. We will suppose that the blood velocity and pressure are available upon solving the weak counterpart of (2.1) and focus our attention on the computation of Problem 2.1. In view of the subsequent analysis, we first introduce the semidiscrete model, namely the time-discrete counterpart of Problem 2.1. To this end, we subdivide the time interval $[0,T]$ in N time steps $t^n = n\Delta t$, with $\Delta t > 0$ and $n = 1,\dots,N$. Setting $\chi = \frac{1}{\Delta t}$, we obtain the time-discrete problem based on the backward Euler method.

PROBLEM 3.1. *Given C_f^0 and C_w^0 for every $n = 0,1,\dots,N-1$ find $C_f^{n+1} \in H^1_{\partial\Omega_f\setminus\Gamma}(\Omega_f)$ and $C_w^{n+1} \in H^1_{\partial\Omega_f\setminus\Gamma}(\Omega_w)$ such that for all ϕ_f in $H^1_{\partial\Omega_f\setminus\Gamma}(\Omega_f)$ and ϕ_w in $H^1_{\partial\Omega_w\setminus\Gamma}(\Omega_w)$*

$$
\left\{
\begin{aligned}
&\widehat{a}_f\left(C_f^{n+1},\phi_f\right) + \left(C_f^{n+1},\phi_f\right)_\zeta - \left(C_w^{n+1},\phi_f\right)_\zeta = \chi\left(C_f^n,\phi_f\right) + \left(s_f^{n+1},\phi_f\right),\\
&\widehat{a}_w\left(C_w^{n+1},\phi_w\right) + \left(C_w^{n+1},\phi_w\right)_\zeta - \left(C_f^{n+1},\phi_w\right)_\zeta = \chi\left(C_w^n,\phi_w\right) + \left(s_w^{n+1},\phi_w\right),
\end{aligned}
\right.
\tag{3.1}
$$

where $s_f^{n+1} = s_f\left(t^{n+1}\right)$, $s_w^{n+1} = s_w\left(t^{n+1}\right)$, C_f^0 and C_w^0 are the initial data, and

$$
\widehat{a}_f\left(\psi_f,\phi_f\right) = \chi\left(\psi_f,\phi_f\right) + a_f\left(\psi_f,\phi_f\right) \quad \forall \phi_f,\psi_f \in H^1_{\partial\Omega_f\setminus\Gamma}(\Omega_f),
\tag{3.2}
$$

$$
\widehat{a}_w\left(\psi_w,\phi_w\right) = \chi\left(\psi_w,\phi_w\right) + a_w\left(\psi_w,\phi_w\right) \quad \forall \phi_w,\psi_w \in H^1_{\partial\Omega_w\setminus\Gamma}(\Omega_w).
\tag{3.3}
$$

Observe that these bilinear forms are continuous and coercive, and, in particular, $\widehat{a}_w\left(\cdot,\cdot\right)$ is *symmetric*.

3.2. The Steklov–Poincaré interface equation. We are going to provide a different formulation of (3.1) which will be useful in view of the subsequent analysis. In what follows, the specification of the time index $n+1$ will be dropped whenever clear from the context.

Let us start with the "wall-side" of the problem. Let $\mathcal{H}_w : \Lambda \to H^1_{\partial\Omega_w\setminus\Gamma}(\Omega_w)$ be defined such that, for a given function $\rho \in \Lambda$, $u_w = \mathcal{H}_w\rho$ solves

$$
\left\{
\begin{aligned}
&\chi u_w - \mu_w \Delta u_w = 0 \quad \text{in} \quad \Omega_w,\\
&u_w = \rho \quad \text{on} \quad \Gamma, \quad u_w = 0 \quad \text{on} \quad \partial\Omega_w \setminus \Gamma.
\end{aligned}
\right.
\tag{3.4}
$$

Moreover, let $\mathcal{G}_w : L^2(\Omega_w) \to H^1_0(\Omega_w)$ be the operator such that, for a given function $r_w \in L^2(\Omega_w)$, $g_w = \mathcal{G}_w r_w$ satisfies

$$\begin{cases} \chi g_w - \mu_w \Delta g_w = r_w \quad \text{in} \quad \Omega_w, \\ \\ g_w = 0 \quad \text{on} \quad \partial\Omega_w. \end{cases} \tag{3.5}$$

Correspondingly, we introduce the following operator $\mathcal{H}_f : \Lambda \to H^1_{\partial\Omega_f\setminus\Gamma}(\Omega_f)$ for the "fluid-side" of the problem. Given $\rho \in \Lambda$, $u_f = \mathcal{H}_f \rho$ satisfies

$$\begin{cases} \chi u_f - \mu_f \Delta u_f + \mathbf{u}\cdot\nabla u_f = 0 \quad \text{in} \quad \Omega_f, \\ \\ u_f = 0 \quad \text{on} \quad \partial\Omega_f \setminus \Gamma, \quad \mu_f \dfrac{\partial u_f}{\partial \mathbf{n}_f} + \zeta u_f = \zeta\rho \quad \text{on} \quad \Gamma. \end{cases} \tag{3.6}$$

Furthermore, $\mathcal{G}_f : L^2(\Omega_f) \to H^1_{\partial\Omega_f\setminus\Gamma}(\Omega_f)$ is such that, given $r_f \in L^2(\Omega_f)$, $g_f = \mathcal{G}_f r_f$ solves

$$\begin{cases} \chi g_f - \mu_f \Delta g_f + \mathbf{u}\cdot\nabla g_f = r_f \quad \text{in} \quad \Omega_f, \\ \\ g_f = 0 \quad \text{on} \quad \partial\Omega_f \setminus \Gamma, \quad \mu_f \dfrac{\partial g_f}{\partial \mathbf{n}_f} + \zeta g_f = 0 \quad \text{on} \quad \Gamma. \end{cases} \tag{3.7}$$

Finally, for a given function $\rho \in \Lambda$, we define

$$\begin{aligned} &\mathcal{S}_w : \Lambda \to \Lambda' \quad \text{such that} \quad \mathcal{S}_w\rho = \mu_w \frac{\partial \mathcal{H}_w \rho}{\partial \mathbf{n}_w} + \zeta\rho, \\ &\mathcal{S}_f : \Lambda \to \Lambda' \quad \text{such that} \quad \mathcal{S}_f\rho = -\zeta(\gamma_f \mathcal{H}_f \rho). \end{aligned} \tag{3.8}$$

By extending a definition which is common in the framework of domain decomposition methods we call $\mathcal{S}_f$ and $\mathcal{S}_w$ the *Steklov–Poincaré* operators associated with the heterogeneous problem at hand. A general discussion on the role and the properties of Steklov–Poincaré (SP) operators in the framework of domain decomposition can be found in [13]. On the basis of the previous definitions, the time-discrete formulation of the problem can be reformulated in terms of an operatorial *interface equation.* Indeed, if we set

$$\mathcal{S} = \mathcal{S}_f + \mathcal{S}_w \quad \text{and} \quad \eta = -\left(\mu_f \frac{\partial \mathcal{G}_f r_f}{\partial \mathbf{n}_f} + \mu_w \frac{\partial \mathcal{G}_w r_w}{\partial \mathbf{n}_w}\right), \tag{3.9}$$

then, from $(2.2)_4$, the following interface equation needs to be solved for the unknown ρ at each time step:

$$\mathcal{S}\rho = \eta\ . \tag{3.10}$$

The SP operators can be reinterpreted in a weak from by considering (3.10) in a distributional sense. Let us denote by $\langle\cdot,\cdot\rangle$ the duality pairing of Λ with Λ'. By Green formula, we obtain for all $\lambda \in \Lambda$

$$\langle \mathcal{S}_w\rho, \lambda\rangle = \left\langle \mu_w \frac{\partial \mathcal{H}_w \rho}{\partial \mathbf{n}_w}, \lambda \right\rangle + (\rho,\lambda)_\zeta = \widehat{a}_w\left(u_w, \mathcal{L}_w\lambda\right) + (\rho,\lambda)_\zeta \tag{3.11}$$

with $\mathcal{L}_w\lambda \in H^1_{\partial\Omega_w\setminus\Gamma}(\Omega_w)$ and

$$\begin{aligned}\langle \mathcal{S}_f\rho,\lambda\rangle &= -\left(\mathcal{H}_f\rho,\lambda\right)_\zeta = -\left(u_f,\lambda\right)_\zeta = -\left(\rho,\lambda\right)_\zeta + \mu_f\left\langle \frac{\partial u_f}{\partial \mathbf{n}_f},\lambda\right\rangle \\ &= -\left(\rho,\lambda\right)_\zeta + \widehat{a}_f\left(u_f,\mathcal{L}_f\lambda\right)\end{aligned} \tag{3.12}$$

with $\mathcal{L}_f\lambda \in H^1_{\partial\Omega_f\setminus\Gamma}(\Omega_f)$. In the latter equation, we have exploited the definition of u_f and, in particular, $(3.6)_2$. Consequently,

$$\langle \mathcal{S}\rho,\lambda\rangle = \mu_w\left\langle \frac{\partial\mathcal{H}_w\rho}{\partial \mathbf{n}_w},\lambda\right\rangle + (\rho,\lambda)_\zeta - \left(\mathcal{H}_f\rho,\lambda\right)_\zeta = \widehat{a}_w\left(u_w,\mathcal{L}_w\lambda\right) + \widehat{a}_f\left(u_f,\mathcal{L}_f\lambda\right) \tag{3.13}$$

for all $\lambda \in \Lambda$. By proceeding in a similar way for the right-hand side, the weak formulation of the interface relation (3.10) finally reads as follows: find $\rho \in \Lambda$ such that

$$\widehat{a}_w\left(\mathcal{H}_w\rho + \mathcal{G}_w r_w,\mathcal{L}_w\lambda\right) + \widehat{a}_f\left(\mathcal{H}_f\rho + \mathcal{G}_f r_f,\mathcal{L}_f\lambda\right) = \left(r_f,\mathcal{L}_f\lambda\right) + \left(r_w,\mathcal{L}_w\lambda\right) \tag{3.14}$$

for all $\lambda \in \Lambda$. Finally, we notice that, in the special case $\phi_f = \mathcal{L}_f\lambda$ and $\phi_w = \mathcal{L}_w\lambda$, and, provided that $r_f = \chi C_f^n + s_f^{n+1}, r_w = \chi C_w^n + s_w^{n+1}$, (3.14) is equivalent to (2.5).

3.3. The fully discrete problem. The *space discretization* of the problem is carried out using the finite element method (FEM). To this end, let us introduce $\mathcal{T}_{hf}$ and $\mathcal{T}_{hw}$, two admissible triangulations (see, e.g., [12]) of $\overline{\Omega}_f$ and $\overline{\Omega}_w$ respectively. For the sake of simplicity, we assume that $\mathcal{T}_{hf}$ and $\mathcal{T}_{hw}$ are *conforming* triangulations on Γ. Consequently, $\mathcal{T}_h = \mathcal{T}_{hf} \cup \mathcal{T}_{hw}$ is an admissible triangulation for $\overline{\Omega}_f \cup \overline{\Omega}_w$. Let h be a characteristic length of the elements $K \in \mathcal{T}_h$ and V_{hf} and V_{hw} be a couple of finite-dimensional subspaces of $H^1_{\partial\Omega_f\setminus\Gamma}$ and $H^1_{\partial\Omega_w\setminus\Gamma}$, respectively. Moreover, denote by $V_{hw,0}$ a finite-dimensional subspace of $H^1_0(\Omega_w)$. Finally let Λ_h be a finite-dimensional subspace of Λ such that the traces on Γ of functions in V_{hf} or V_{hw} belong to Λ_h. Denote by N_f the dimension of V_{hf}, by N_w the dimension of V_{hw}, and by N_Γ the dimension of Λ_h. Let $\{\phi_{i,f}\}$ $(i = 1,2,\dots,N_f)$ and similarly $\{\phi_{i,w}\}$ $(i = 1,2,\dots,N_w)$ be a basis for V_{hf} and for V_{hw}, respectively; moreover, denote with $\{\phi_{i,\Gamma}\}$ $(i = 1,2,\dots,N_\Gamma)$ a basis for Λ_h. In what follows, the subscript h will identify the space discrete solution.

Based on these definitions, the space discretization of Problem 2.1 reads as follows.

PROBLEM 3.2. *Given the initial data C^0_{fh} and C^0_{wh}, for every $n = 0,1,\dots,N-1$ (being $N\Delta t = T$ the final time), find $C^{n+1}_{fh} \in V_{hf}$ and $C^{n+1}_{wh} \in V_{hw}$ such that for all $i = 1,\dots,N_f$ and $j = 1,\dots,N_w$*

$$\begin{cases} \widehat{a}_f\left(C^{n+1}_{fh},\phi_{i,f}\right) + \left(C^{n+1}_{fh},\phi_{i,f}\right)_\zeta - \left(C^n_{wh},\phi_{i,f}\right)_\zeta = \chi\left(C^n_{fh},\phi_{i,f}\right) + \left(s^{n+1}_f,\phi_{i,f}\right), \\ \widehat{a}_w\left(C^{n+1}_{wh},\phi_{j,w}\right) + \left(C^{n+1}_{wh},\phi_{j,w}\right)_\zeta - \left(C^{n+1}_{fh},\phi_{j,w}\right)_\zeta = \chi\left(C^n_{wh},\phi_{j,w}\right) + \left(s^{n+1}_w,\phi_{j,w}\right). \end{cases} \tag{3.15}$$

REMARK 3.1. *Due to the convection dominated nature of the problem (see* [15]*), the pure Galerkin discrete formulation* (3.15) *does not ensure stability to the numerical solution unless the triangulation is fine enough to ensure that the local Péclet number is less than one. It is therefore mandatory to adopt suitable stabilization techniques. In particular, we will make use of a strongly consistent method like SUPG* *(streamline upwind Petrov Galerkin)—see, e.g.,* [12]. *We address first the analysis of the scheme to the Galerkin formulation* (3.15) *in order to point out the intrinsic features of the method. Later on, we will address the modifications induced by the presence of stabilizing terms (see section* 4.1*).*

3.3.1. The discrete SP operators. Let us introduce the discrete counterpart $\widehat{\mathcal{S}}_{hw}$ and $\widehat{\mathcal{S}}_{hf}$ of the SP operators. Following (3.11), we define

$$\langle \widehat{\mathcal{S}}_{hw}\rho, \lambda\rangle = \widehat{a}_w\left(u_w, \mathcal{L}_w\lambda\right) + \left(\rho, \lambda\right)_\zeta \quad \forall \rho, \lambda \in \Lambda_h, \tag{3.16}$$

where

$$u_w \in V_{hw} : \begin{cases} \widehat{a}_w\left(u_w, \phi_w\right) = 0 \quad \forall \phi_w \in V_{hw,0}, \\ u_w = \rho \quad on\ \Gamma. \end{cases}$$

Similarly we define

$$\langle \widehat{\mathcal{S}}_{hf}\rho, \lambda\rangle = -\left(\rho, \lambda\right)_\zeta + \widehat{a}_f\left(u_f, \mathcal{L}_f\lambda\right), \tag{3.17}$$

where

$$u_f \in V_{hf} : \widehat{a}_f\left(u_f, \phi_f\right) + \left(u_f, \phi_f\right)_\zeta = \left(\rho, \phi_f\right)_\zeta \quad \forall \phi_f \in V_{hf}. \tag{3.18}$$

Finally, we set $\widehat{\mathcal{S}}_h = \widehat{\mathcal{S}}_{hw} + \widehat{\mathcal{S}}_{hf}$. Then, the solution of Problem 3.2 can be reformulated in terms of the interface equation

$$\langle \widehat{\mathcal{S}}_h\rho, \lambda\rangle = \langle \widehat{\eta}, \lambda\rangle \quad \forall \lambda \in \Lambda, \tag{3.19}$$

where ρ stands for the trace of C_{wh} on Γ, i.e.,

$$\rho = C_{wh}\mid_\Gamma$$

and

$$\widehat{\eta} = -\left(\mu_w \frac{\partial \mathcal{G}_w r_w}{\partial \mathbf{n}_w} + \mu_f \frac{\partial \mathcal{G}_f r_f}{\partial \mathbf{n}_f}\right), \tag{3.20}$$

with

$$r_f = r_f^{n+1} = s_f^{n+1} + \chi C_{fh}^n, \quad r_w = r_w^{n+1} = s_w^{n+1} + \chi C_{wh}^n, \tag{3.21}$$

where, for the sake of clarity, the time index has been restored.

In what follows we will use several times the so called *finite element uniform extension theorem* (FEUET), which states that $\|\rho_h\|_\Lambda$ is uniformly equivalent (with respect to h) to $\|u_{h,w}\|_{H^1(\Omega_w)}$, where $u_{h,w}$ is the finite element approximation of problem (3.4), where ρ is replaced by ρ_h; see [13, Theorem 4.1.3].

PROPOSITION 3.1. *The following properties hold for $\widehat{\mathcal{S}}_{hw}$, $\widehat{\mathcal{S}}_{hf}$, and $\widehat{\mathcal{S}}_h$:*

1. *$\widehat{\mathcal{S}}_{hw}$ is continuous, symmetric, and coercive in Λ_h.*
2. *$\widehat{\mathcal{S}}_{hf}$ is continuous and negative; i.e., for any $\rho \in \Lambda_h$, $\langle \widehat{\mathcal{S}}_{hf}\rho, \rho\rangle < 0$.*
3. *$\widehat{\mathcal{S}}_h$ is continuous and coercive in Λ_h.*

Proof. Continuity and symmetry of $\widehat{\mathcal{S}}_{hw}$ can be proven from the continuity and symmetry of $\widehat{a}_w(\cdot,\cdot)$ and the definition (3.16), owing to the FEUET. Moreover, $\widehat{\mathcal{S}}_{hw}$ is coercive thanks to the coercivity of $\widehat{a}_w(\cdot,\cdot)$, the positivity of ζ, and the trace inequality (1.1). Observe in particular that the continuity and coercivity constants do not depend on h.

Continuity of $\widehat{\mathcal{S}}_{hf}$ follows from the coercivity of the bilinear form on the right-hand side of (3.18). Again, the continuity constant does not depend on h, depending on the bilinear form properties. Moreover, from (3.17) and taking again $\phi_f = u_f$ in (3.18), we obtain

$$-\langle \widehat{\mathcal{S}}_{hf}\rho, \rho\rangle = (u_f, \rho)_\zeta = \widehat{a}_f\,(u_f, u_f) + \|u_f\|_\zeta^2 > 0 \quad \forall \rho \in \Lambda_h. \tag{3.22}$$

Item 2 is thus proven. The continuity of $\widehat{\mathcal{S}}_h$ is now a consequence of the continuity of $\widehat{\mathcal{S}}_{hw}$ and $\widehat{\mathcal{S}}_{hf}$. In particular, we stress that the continuity constant is independent on h.

Finally, we prove that $\widehat{\mathcal{S}}_h$ is coercive. Indeed, observe that by the definition of u_f and (3.22)

$$\widehat{a}_f\,(u_f, \mathcal{L}_f\rho) = (\rho - \gamma_f u_f, \rho)_\zeta = (\rho - \gamma_f u_f, \rho - \gamma_f u_f)_\zeta + \widehat{a}_f\,(u_f, u_f)\,. \tag{3.23}$$

Consequently, we obtain

$$\langle \widehat{\mathcal{S}}_h\rho, \rho\rangle = \widehat{a}_w\,(u_w, \mathcal{L}_w\rho) + \widehat{a}_f\,(u_f, \mathcal{L}_f\rho) \geq \widehat{a}_w\,(u_w, u_w) \geq \tau\|\rho\|_\Lambda^2. \tag{3.24}$$

Finally, observe that the coercivity constant considered above is independent of h. Indeed, the coercivity constant τ in (3.24) is independent of h. This is due to the circumstance that u_w is the extension of ρ so that we can advocate once again the FEUET. □

As a direct consequence of the previous properties, we obtain the following corollary.

COROLLARY 3.1. *The interface equation* (3.19) *admits a unique solution which depends continuously on the data.*

Proof. The functional $\mathcal{F}(\lambda) \equiv \langle \widehat{\eta}, \lambda\rangle$ associated with the right-hand side of (3.19) is continuous in Λ_h. Moreover, the bilinear form $\langle \widehat{\mathcal{S}}_h\cdot, \cdot\rangle$ is continuous and coercive in Λ_h, as proved in Proposition 3.1. The result then follows from the Lax–Milgram lemma. □

4. The subdomain iterative method.

In order to reduce the computational cost required by the numerical solution of Problem 3.2, we suitably split the whole problem into a sequence of subproblems to be solved in the two physical subdomains (the lumen Ω_f and the wall Ω_w). In [14] we have introduced and analyzed an iterative scheme based on the interface conditions $(2.2)_3$ and $(2.2)_4$. In the present work, we consider a relaxed extension of the scheme in order to speed up the convergence. We will extend the convergence results proven in [14] to the relaxed algorithm, taking advantage of the reinterpretation of the scheme in terms of SP operators. (That was not considered at all in [14].) In what follows, for notational convenience we will drop the specification h, for the space discrete quantities, as well as the specification $n+1$ for the time-discrete quantities (i.e., we will write $[C_{f,k}, C_{w,k}]$ instead of $[C_{fh,k}^{n+1}, C_{wh,k}^{n+1}]$). It is understood, however, that the iterative method is carried out on the fully discrete problem.

The scheme which we adopt is the following. Given an initial guess ρ_0 for $C_{w,0}$ on the interface Γ, for $k = 0, 1, \ldots$ find the sequence of functions $[C_{f,k}, C_{w,k}] \in V_{hf} \times V_{hw}$ by solving for all $i = 1, \ldots, N_f$ and $j = 1, \ldots, N_w$

$$\widehat{a}_f\,(C_{f,k+1}, \phi_{i,f}) + (C_{f,k+1}, \phi_{i,f})_\zeta = (r_f, \phi_{i,f}) + (\rho_k, \phi_{i,f})_\zeta, \tag{4.1}$$

$$\widehat{a}_w\left(C_{w,k+1},\phi_{j,w}\right)+\left(C_{w,k+1},\phi_{j,w}\right)_\zeta=\left(r_w,\phi_{j,w}\right)+\left(C_{f,k+1},\phi_{j,w}\right)_\zeta, \tag{4.2}$$

$$\rho_{k+1}=C_{w,k+1}|_\Gamma, \tag{4.3}$$

where r_f and r_w are defined in (3.21).

This is a *Robin–Robin* iterative scheme, as it is based on the conditions $(2.2)_3$ and $(2.2)_4$, which are Robin conditions for each subproblem involved.

In order to resort to an iterative scheme for the variable ρ_{k+1} alone, let us eliminate the unknowns $C_{f,k+1}, C_{w,k+1}$ from (4.1)–(4.3).

Let us take $\phi_f=\mathcal{L}_f\lambda$ in (4.1), where λ is any function of Λ_h, and split $C_{f,k+1}$ as $C_{f,k+1}=u_{f,k+1}+g_f$, where $u_{f,k+1}$ and g_f solve, respectively, problems (3.6) and (3.7) in a weak sense. Recalling (3.17), we have

$$\begin{aligned}\left(C_{f,k+1},\lambda\right)_\zeta&=\left(r_f,\mathcal{L}_f\lambda\right)+\left(\rho_k,\lambda\right)_\zeta-\widehat{a}_f\left(u_{f,k+1},\mathcal{L}_f\lambda\right)-\widehat{a}_f\left(g_f,\mathcal{L}_f\lambda\right)\\&=-\langle\nabla g_f\cdot\mathbf{n}_f,\lambda\rangle-\langle\widehat{\mathcal{S}}_{hf}\rho_k,\lambda\rangle\quad\forall\lambda\in\Lambda_h.\end{aligned} \tag{4.4}$$

Proceeding similarly on (4.2) and recalling that $C_{w,k+1}=u_{w,k+1}+g_w$, we obtain

$$\langle\widehat{\mathcal{S}}_{hw}C_{w,k+1},\lambda\rangle=\widehat{a}_w\left(u_{w,k+1},\mathcal{L}_w\lambda\right)+\left(C_{w,k+1},\lambda\right)_\zeta\quad\forall\lambda\in\Lambda_h$$

and

$$\widehat{a}_w\left(g_w,\mathcal{L}_w\lambda\right)=\left(r_w,\mathcal{L}_w\lambda\right)+\langle\nabla g_w\cdot\mathbf{n}_w,\lambda\rangle\quad\forall\lambda\in\Lambda_h.$$

Consequently, taking $\phi_w=\mathcal{L}_w\lambda$ in (4.2), by virtue of the latter two equations we have

$$\langle\widehat{\mathcal{S}}_{hw}C_{w,k+1},\lambda\rangle+\langle\nabla g_w\cdot\mathbf{n}_w,\lambda\rangle=\left(C_{f,k+1},\lambda\right)_\zeta\quad\forall\lambda\in\Lambda_h.$$

Substituting (4.4) in the latter, we obtain

$$\langle\widehat{\mathcal{S}}_{hw}C_{w,k+1},\lambda\rangle=\langle\widehat{\mathcal{S}}_{hw}\rho_{k+1},\lambda\rangle=\langle\widehat{\eta},\lambda\rangle-\langle\widehat{\mathcal{S}}_{hf}\rho_k,\lambda\rangle\quad\forall\lambda\in\Lambda_h, \tag{4.5}$$

where from (3.20)

$$\widehat{\eta}=-\left(\mu_f\nabla g_f\cdot\mathbf{n}_f+\mu_w\nabla g_w\cdot\mathbf{n}_w\right).$$

If we consider a relaxation parameter θ, (4.3) becomes

$$\rho_{k+1}=\theta C_{w,k+1}|_\Gamma+(1-\theta)\rho_k. \tag{4.6}$$

Hence for all $\lambda\in\Lambda_h$, by means of (4.5), we obtain

$$\langle\widehat{\mathcal{S}}_{hw}(\rho_{k+1}-\rho_k),\lambda\rangle=\theta\langle\widehat{\mathcal{S}}_{hw}C_{w,k+1},\lambda\rangle-\theta\langle\widehat{\mathcal{S}}_{hw}\rho_k,\lambda\rangle=\theta\langle\widehat{\eta}-\widehat{\mathcal{S}}_h\rho_k,\lambda\rangle. \tag{4.7}$$

Equation (4.7) actually sheds light on a useful reinterpretation of the Robin–Robin method, which can be regarded as a *preconditioned Richardson method* for problem (3.19), with $\widehat{\mathcal{S}}_{hw}$ playing the role of preconditioner for $\widehat{\mathcal{S}}_h$. This observation allows for a straightforward proof of convergence of the scheme. Indeed, we have the following abstract result (for the proof, see, e.g., [13, Theorem 4.2.2 and Remark 4.2.4]).

THEOREM 4.1. *Let X be a real Hilbert space, X' its dual space, and $\langle\cdot,\cdot\rangle$ the duality paring between X' and X. Let $\mathcal{Q}: X\to X'$ be a linear invertible continuous* *operator which can be split as* $\mathcal{Q} = \mathcal{Q}_1 + \mathcal{Q}_2$*, where* $\mathcal{Q}_i$ $(i = 1, 2)$ *are linear operators. Consider the problem*

$$\mathcal{Q}\rho = \eta, \tag{4.8}$$

where $\eta \in X'$ *is given and* $\rho \in X$ *is to be determined. Suppose that* $\mathcal{Q}_2$ *is continuous, symmetric, and coercive and that* $\mathcal{Q}$ *is coercive. Then, there exists a real value* θ_{max}*, depending on the continuity and the coercivity constants, such that for any* $\theta \in (0, \theta_{max})$*, the sequence*

$$\rho_{k+1} = \rho_k + \theta \mathcal{Q}_2^{-1} (\eta - \mathcal{Q}\rho_k) \tag{4.9}$$

converges in X *to the solution of* (4.8) *for any* $\rho_0 \in X$*.*

We point out that (4.9) is the generic iteration of a preconditioned Richardson method to solve (4.8), with $\mathcal{Q}_2$ acting as preconditioner (see [16], [9]).

Now, if we set $X = \Lambda_h$, $\mathcal{Q} = \widehat{\mathcal{S}}_h$, $\mathcal{Q}_1 = \widehat{\mathcal{S}}_{hf}$, $\mathcal{Q}_2 = \widehat{\mathcal{S}}_{hw}$, we apply this result to prove the convergence of the sequence $\{\rho_k\}, k \geq 0$ to the solution of (3.19). Indeed, all the hypotheses of the theorem are verified, as proven in the previous section.

Observe, moreover, that, in Theorem 4.1, the convergence rate as well as the optimal value for θ are functions of the coercivity and continuity constants of $\mathcal{Q}_2$ and of $\mathcal{Q}$ that in our case read $\widehat{\mathcal{S}}_{hw}$ and $\widehat{\mathcal{S}}_{hw}$, respectively. In proving Proposition 3.1 we pointed out that these constants are *independent of the mesh size h* in our problem. This means that the rate of convergence of the proposed preconditioned iterations is not affected by the mesh size, or, in other words, that the preconditioner is *optimal.*

Altogether, these observations can be collected in the following final result.

PROPOSITION 4.1. *The Robin–Robin iterative scheme* (4.1)*,* (4.2)*, and* (4.6) *is convergent for any* $\theta \in (0, \theta_{max})$*, and its rate of convergence is independent of the spatial discretization. Precisely, there exists* $K < 1$ *such that for any* $\theta \in (0, \theta_{max})$*, there exists a constant* $K_\theta \leq K$ *such that*

$$\|\rho - \rho_{k+1}\|_\Lambda \leq K_\theta \|\rho - \rho_k\|_\Lambda, \qquad k \geq 0.$$

In agreement with this conclusion, Table 4.1 shows that the convergence rate of the *relaxed Robin–Robin* scheme is uniformly independent of the parameter h (which in our computations with uniform grids is related to the number N of finite elements nodes through the law $N \approx \mathcal{O}(h^{-2})$).

REMARK 4.1. *The convergence result of Theorem* 4.1 *refers to the preconditioned Richardson iterative method. However, similar conclusions can be shown when more efficient methods are applied to the same problem (see Table* 4.1*), such as the generalized minimal residual (GMRES) (see* [16]*), as proven in a very abstract form in* [13, *sect.* 4.2.1]. *On the basis of this result, we will proceed with the GMRES methods later on in section* 4.3.

4.1. The iterative method in the convection dominated case.

As previously pointed out, when advection dominated problems are considered, stabilization techniques are required. This means that the bilinear form $\widehat{a}_f(\cdot, \cdot)$ introduced in (3.2) is substituted by

$$\widehat{a}_{f,stab}(C_f, \phi_f) = \widehat{a}_f(C_f, \phi_f) + a_{f,h}(C_f, \phi_f), \tag{4.10}$$

where $a_{f,h}(C_f, \phi_f)$ depends on the specific stabilization method. For instance (see, e.g., [12]), if we set

$$L_s C = -\nabla \cdot \mu_f \nabla C, \quad L_{ss} C = \frac{1}{2} \nabla \cdot \mathbf{u} C + \frac{1}{2} \mathbf{u} \cdot \nabla C \quad (L_f = L_s + L_{ss}),$$

Table 4.1

Comparison of the number of iterations to reach convergence. In all these tests we have taken $\Omega_f = (0,4)\times(0,1)$, $\Omega_w = (0,4)\times(-1,0)$, $u_x = 4u_0(1-y)y$, $u_y = 0$. (A) $\mu_f = \mu_w = 1.0\ cm^2s^{-1}$, *and* $\zeta = 1.0\ cm\ s^{-1}$. *(Values* * *refer to the pure Galerkin method, the other to the stabilized SUPG method.)* (B) $\mu_f = \mu_w = 10^{-3}cm^2s^{-1}$ *and* $\zeta = 1.0\ cm\ s^{-1}$. *Finer grids are obtained by means of a uniform refinement plus regularization; thus* $N = \mathcal{O}(h^{-2})$. *For large values of* ζ *the coupled problem is severely ill-conditioned, yet the number of iterations is uniformly independent of* h.

(A)				
h	N	Unrelax. Rich.	Relax. Rich.	P-GMRES
0.1	4000	4*	4*	3*
0.05	16000	4*	4*	3*
0.025	60000	4*	4*	3*
0.01875	106000	4*	4*	3*
0.012	260000	4*	4*	3*

(B)				
h	N	Unrelax. Rich.	Relax. Rich.	P-GMRES
0.1	4000	8	7	5
0.05	16000	12	10	6
0.025	60000	20	15	7
0.01875	106000	23	17	7
0.012	260000	29-29*	20-20*	8-8*

the symmetric and the skew-symmetric parts of the fluid differential operator L_f, respectively, then the most common strongly consistent stabilization methods resort to set

$$a_{f,h}\left(C_f,\phi_f\right) = \sum_{K\in\mathcal{T}_h} \delta\left(LC_f, \frac{h_K}{|\mathbf{u}|}\left(L_{ss}+\kappa L_s\right)\phi_f\right)_K,$$

where K is the generic element of the triangulation $\mathcal{T}_h$ (supposed to be regular) with diameter h_K, $(\cdot,\cdot)_K$ denotes the $L^2(K)$ scalar product, δ is a parameter to be chosen, and κ identifies the different stabilization techniques. In particular, SUPG corresponds to set $\kappa = 0$, while the Galerkin least squares (GaLS) method is given for $\kappa = 1$. The Douglas–Wang (DW) method, on the other hand, corresponds to $\kappa = -1$. For the SUPG method, we recall that if δ is suitably chosen, it is possible to prove that the stabilized bilinear form $\widehat{a}_{f,stab}(\cdot,\cdot)$ is coercive, the constant of coercivity being independent of h (see [12, Proposition 8.4.1]). A similar result holds for the DW and for the GaLS methods. In the latter case, the coercivity holds for any positive δ.

Starting form these results, we are able to prove that the convergence rate of the iterative subdomains method proposed is independent of h even in the stabilized case. First of all, the problem we consider in this case, in terms of SP operators can be formulated in a way completely similar to the one proposed in the previous section, provided that $\widehat{\mathcal{S}}_{hf}$ defined in (3.17) and (3.18) is substituted by

$$\langle\widehat{\mathcal{S}}_{hf,stab}\rho,\lambda\rangle = -\left(\rho,\lambda\right)_\zeta + \widehat{a}_{f,stab}\left(u_f,\mathcal{L}_f\lambda\right), \tag{4.11}$$

with

$$u_f \in V_{hf} : \widehat{a}_{f,stab}\left(u_f,\phi_f\right) + \left(u_f,\phi_f\right)_\zeta = \left(\rho,\phi_f\right)_\zeta \quad \forall\phi_f\in V_{hf}. \tag{4.12}$$

The crucial point is to prove that the stabilized SP operator $\widehat{\mathcal{S}}_{hf,h}$ is also continuous with a continuity constant independent of h. Actually, observe that by definition

$$\langle\widehat{\mathcal{S}}_{hf,stab}\rho,\lambda\rangle = \left(u_f,\lambda\right)_\zeta.$$

From (4.12) for $\phi_f = u_f$, we deduce that

$$\| \gamma_f u_f \|_\Lambda \leq \beta \| \rho \|_\Lambda,$$

where β is a constant obtained as a function of the trace and coercivity constants and therefore independent of h. The continuity constant of $\widehat{\mathcal{S}}_{hf,stab}$ is thus independent of h. Consequently, the extension of Proposition 3.1 to the stabilized case is straightforward. We therefore conclude that Theorem 4.1 can be applied as well in the stabilized case, again having independence of the convergence rate and the optimal value of θ on h.

4.2. Algebraic reinterpretation of the iterative scheme. The coupled Problem 3.2 requires at each time step the solution of a system in the form

$$A\mathbf{c} = \mathbf{b}. \tag{4.13}$$

Denoted by N_Γ the degrees of freedom associated with the interface Γ, the number of degrees of freedom associated with the inner nodes in Ω_w is given by $N_{w0} = N_w - N_\Gamma$. Consequently, in (4.13), $\mathbf{c} = [\mathbf{c}_f, \mathbf{c}_w, \mathbf{c}_\Gamma]^T \in \mathbb{R}^{N_f+N_w}$ is the vector of unknowns specifying the discrete solution, with $\mathbf{c}_f \in \mathbb{R}^{N_f}$, $\mathbf{c}_w \in \mathbb{R}^{N_{w0}}$, $\mathbf{c}_\Gamma \in \mathbb{R}^{N_\Gamma}$. $\mathbf{b} \in \mathbb{R}^{N_f+N_w}$ is a function of the forcing terms, the boundary conditions, and the solution computed at the previous steps and can be correspondingly split as $\mathbf{b} = [\mathbf{b}_f, \mathbf{b}_w, \mathbf{b}_\Gamma]^T$, $\mathbf{b}_f \in \mathbb{R}^{N_f}$, $\mathbf{b}_w \in \mathbb{R}^{N_{w0}}$, $\mathbf{b}_\Gamma \in \mathbb{R}^{N_\Gamma}$. Finally, $A \in \mathbb{R}^{(N_f+N_w)\times(N_f+N_w)}$ has the following blockwise pattern:

$$A = \begin{bmatrix} A_{ff} & \mathbf{0} & A_{f\Gamma} \\ \mathbf{0} & A_{ww} & A_{w\Gamma} \\ A_{\Gamma f} & A_{\Gamma w} & A_{\Gamma\Gamma} \end{bmatrix}, \tag{4.14}$$

where A_{ff} is the $N_f \times N_f$ matrix associated with the discretization of the bilinear form $\widehat{a}_f(\psi_f, \phi_f) + (\psi_f, \phi_f)_\zeta$ for $\psi_f, \phi_f \in V_{hf}$. Correspondingly, $A_{f\Gamma}$ arises from the discretization of the term $-(\lambda, \phi_f)_\zeta$ (with $\lambda \in \Lambda_h$), while A_{ww} is associated with the discretization of $\widehat{a}_w(\psi_{w0}, \phi_{w0})$ for $\psi_{w0}, \phi_{w0} \in V_{hw,0}$ (functions with null trace on Γ) and $A_{w\Gamma}$ refers to the discretization of $\widehat{a}_w(\mathcal{L}_w\lambda, \phi_{w0})$ for $\lambda \in \Lambda_h$. $A_{\Gamma\Gamma}$ is related to the discretization of $\widehat{a}_w(\mathcal{L}_w\lambda, \mathcal{L}_w\rho) + (\lambda, \rho)_\zeta$ for $\lambda, \rho \in \Lambda_h$. Finally, we note that $A_{\Gamma f} = A_{f\Gamma}^T$ and $A_{\Gamma w} = A_{w\Gamma}^T$ and A_{ww} and $A_{\Gamma\Gamma}$ are symmetric.

The algebraic reinterpretation of the Robin–Robin iterative scheme readily follows. Our substructuring iterative method resorts to a preconditioned Richardson scheme for (4.13),

$$Q(\mathbf{c}_{k+1} - \mathbf{c}_k) = \theta(\mathbf{b} - A\mathbf{c}_k) = \theta\mathbf{r}_k, \qquad k \geq 0, \tag{4.15}$$

in which the matrix

$$Q = \begin{bmatrix} A_{ff} & 0 & 0 \\ 0 & A_{ww} & A_{w\Gamma} \\ A_{\Gamma f} & A_{\Gamma w} & A_{\Gamma\Gamma} \end{bmatrix} \tag{4.16}$$

is the preconditioner.

If we formally eliminate $\mathbf{c}_f$ and $\mathbf{c}_w$ in (4.13), we obtain the reduced system

$$\Sigma \mathbf{c}_\Gamma = \boldsymbol{\eta} \tag{4.17}$$

with $\boldsymbol{\eta} = \mathbf{b}_\Gamma - A_{\Gamma f}A_{ff}^{-1}\mathbf{b}_f - A_{\Gamma w}A_{ww}^{-1}\mathbf{b}_w$ and $\Sigma = A_{\Gamma\Gamma} - A_{\Gamma w}A_{ww}^{-1}A_{w\Gamma} - A_{\Gamma f}A_{ff}^{-1}A_{f\Gamma}$. Consequently, we obtain the splitting $\Sigma = \Sigma_w + \Sigma_f$, where $\Sigma_w = A_{\Gamma\Gamma} - A_{\Gamma w}A_{ww}^{-1}A_{w\Gamma}$, $\Sigma_f = A_{\Gamma f}A_{ff}^{-1}A_{f\Gamma}$. Matrix Σ is the *Schur complement* of matrix A. Equation (4.17) represents the finite-dimensional counterpart of (3.19), and Σ is the algebraic counterpart of the SP operator $\widehat{\mathcal{S}}_h$, while Σ_f and Σ_w play the role of $\widehat{\mathcal{S}}_{hf}$ and $\widehat{\mathcal{S}}_{hw}$, respectively.

By means of the block *LU* factorization of A, we can explicitly compute the iteration matrix associated with the following iterative scheme:

$$I - \theta Q^{-1}A = \begin{bmatrix} (1-\theta)I_f & 0 & -\theta A_{ff}^{-1}A_{f\Gamma} \\ 0 & (1-\theta)I_w & -\theta A_{ww}^{-1}A_{w\Gamma}\Sigma_w^{-1}A_{\Gamma f}A_{ff}^{-1}A_{f\Gamma} \\ 0 & 0 & I - \theta\Sigma_w^{-1}\Sigma \end{bmatrix}.$$

In particular, on the third block the system (4.15) yields

$$\Sigma_w\left(\mathbf{c}_{\Gamma,k+1} - \mathbf{c}_{\Gamma,k}\right) = \theta\left(\boldsymbol{\eta} - \Sigma\mathbf{c}_{\Gamma,k}\right). \tag{4.18}$$

The latter relation enlightens the role of the matrix Σ_w as a preconditioner for the Schur complement in the interface problem (4.17). As the algebraic counterpart of the operator $\widehat{\mathcal{S}}_{hw}$, Σ_w is symmetric and positive definite.

From this perspective, we can reformulate Proposition 4.1 in the following manner.

PROPOSITION 4.2. *The preconditioned Richardson scheme* (4.15) *converges for any θ belonging to a suitable interval* $(0, \theta_{max})$. *The preconditioner Q given in* (4.16) *is optimal, making the rate of convergence independent of the space discretization.*

4.3. Acceleration strategies. The reinterpretation of the Robin–Robin scheme based on the Richardson framework has the advantage of highlighting that Σ_w is an optimal preconditioner for Σ. (As well, Q is an optimal preconditioner for A.) On this ground, we will take advantage of these preconditioners when more efficient iterative procedures will be applied. We will start analyzing the effects of the static relaxation parameter in the Richardson framework, and then we will consider dynamical strategies such as GMRES (see [16]).

Stationary Richardson methods. In order to compare the unrelaxed and the relaxed schemes, observe that the behavior of the iteration matrix $I - \theta Q^{-1}A$ is governed by the third diagonal block $I - \theta\Sigma_w^{-1}\Sigma$. In our case, Σ_w and Σ are positive definite, but Σ is not symmetric since A_{ff} is not symmetric (due to the convection term). Therefore we cannot assert a priori that $\Sigma_w^{-1}\Sigma$ has real positive eigenvalues; thus, an optimal static choice of θ is not straightforward (see, e.g., [9]). However, as a heuristic choice, parameter θ is selected assuming that Σ is symmetric, setting

$$\theta = \frac{2}{\lambda_{\min} + \lambda_{\max}},$$

where $\sigma = \{\lambda_i\}$ $i = 1, \dots, N_\Gamma$ is the spectrum of $\Sigma_w^{-1}\Sigma$, for which a rough estimate can be obtained as follows. We observe that since $I - \Sigma_w^{-1}\Sigma = -\Sigma_w^{-1}\Sigma_f$, the iteration matrix associated with (4.18), in the unrelaxed case, and since $\tau = \{\mu_i\}$ (for $i = 1, \dots, N_\Gamma$) is its spectrum, the following inequalities hold:

$$\lambda_i = 1 - \mu_i \text{ and } \mu_i \leq \|-\Sigma_w^{-1}\Sigma_f\|_p \quad \forall i = 1, \dots, N_\Gamma \quad \forall p > 0, \tag{4.19}$$

$$M_p := \frac{\| \mathbf{c}_\Gamma^k - \mathbf{c}_\Gamma^{k-1} \|_p}{\| \mathbf{c}_\Gamma^{k-1} - \mathbf{c}_\Gamma^{k-2} \|_p} \leq \|-\Sigma_w^{-1}\Sigma_f\|_p. \tag{4.20}$$

Consequently, making the approximation, $M_p \simeq \|-\Sigma_w^{-1}\Sigma_f\|_p$, and noticing that $\lambda_{\min} + \lambda_{\max} = 2 - \mu_{\max} - \mu_{\min}$ we make the following choice: $\theta \simeq \frac{2}{2-M_p}$.

Table 4.1 resumes the comparison of these methods for a diffusion dominated case (A) and an advection dominated one (B). From these results we see that, when the global matrix A is ill-conditioned, the relaxation technique enhances the convergence performances with respect to the unrelaxed case.

Dynamical choice of θ. In this approach the choice of θ is pursued automatically by the chosen algorithm.

Recalling that the matrix A is positive definite but not symmetric, we consider, for instance, the preconditioned generalized minimal residual (P-GMRES) method (see, for example, [12], [16]), where Q is the preconditioner for the global system $A\mathbf{c} = \mathbf{b}$. As we have already pointed out in Remark 4.1, Q being the optimal preconditioner derived from the previous analysis, the convergence rate of the P-GMRES algorithm is independent of the number of degrees of freedom of the global system $A\mathbf{c} = \mathbf{b}$. Table 4.1 shows that the P-GMRES method performs better than the stationary strategies for both test cases.

Finally, we point out that for these tests the coefficients μ_f, μ_w, ζ have been chosen with the purpose of making the matrix A very ill-conditioned; this explains the different performance of the considered iterative methods. However, when μ_f, μ_w, ζ are chosen in the biological range for the specific application at hand, the condition number of A is lower; thus the number of iterations necessary to reach convergence is smaller for each one of the considered methods. On the other hand, for the bioengineering applications, a very large number of unknowns is required; consequently, most of the computational time is spent for solving the subsystems $Q\mathbf{z} = \mathbf{r}$ deriving from the preconditioning step. This explains why we have based the choice of an efficient iterative method not only on the number of iterations but also on the computational cost of each single iteration. In particular, a reduction of the time necessary to pursue each iteration is, for example, obtained by resorting to *flexible preconditioned iterative solvers* (see [16]). Among all the iterative methods considered above, the only one that allows a flexible variant is the P-GMRES. Let us briefly introduce it.

In the framework of descent methods, the computation of the descent direction $\mathbf{z}$ is carried out by solving a system $Q\mathbf{z} = \mathbf{r}$ ($\mathbf{r}$ being the residual) which in our case, from definition (4.16), requires the solution of two subsystems:

$$A_{ff}\mathbf{z}_f = \mathbf{r}_f, \tag{4.21}$$

$$\begin{bmatrix} A_{ww} & A_{w\Gamma} \\ A_{\Gamma w} & A_{\Gamma\Gamma} \end{bmatrix} \begin{bmatrix} \mathbf{z}_w \\ \mathbf{z}_\Gamma \end{bmatrix} = \begin{bmatrix} \mathbf{r}_w \\ \mathbf{r}_\Gamma \end{bmatrix} - \begin{bmatrix} 0 \\ A_{\Gamma f}\mathbf{r}_f \end{bmatrix}, \tag{4.22}$$

which can be carried out by means of iterative methods such as BiCGStab or GMRES. The end of this flexible strategy is to compute the global solution $A\mathbf{c} = \mathbf{b}$ by solving (4.21) and (4.22) with a large tolerance, reducing henceforth the computational cost of the solution process. This can be done through the algorithm described below. Let us define with $\hat{Q}_j^{-1}$ $j = 1, 2, 3, \ldots$ an approximation of Q^{-1} (obtained in our case by a low cost solution of (4.21), (4.22)) then the flexible P-GMRES (F-P-GMRES) algorithm reads (see [16]) as follows.

Table 4.2

Quantitative comparison between P-GMRES and F-P-GMRES for different test cases. This example shows that the F-P-GMRES method and the proposed choice of q work well for different geometries, discretized with different triangulations sharing, however, a comparable number of nodes. Note that the tolerance of the outer iterative procedure is always the same (equal to 10^{-10}).

	P-GMRES ($p = 10$)	F-P-GMRES ($p = 10, q = 5$)
Rectangular domains ($h = 0.012$)		
Iterations	2	2
Time/iteration (s) (aver.)	239.18	115.51
Carotid bifurcation		
Iterations	3	3
Time/iteration (s) (aver.)	35.93	15.96
Stenosed artery		
Iterations	3	3
Time/iteration (s) (aver.)	92.43	40.04

Algorithm 4.1. *F-P-GMRES.*

```
Compute $\mathbf{z}_0 = \hat{Q}_0^{-1}(\mathbf{b} - A\mathbf{x}_0)$, $\beta = \|\mathbf{z}_0\|_2$, $\mathbf{v}_1 = \mathbf{r}_0/\beta$.
For $j = 1, \dots, m$ Do:
    compute $\mathbf{w} = A\mathbf{v}_j$,
    compute $\mathbf{z}_j = \hat{Q}_j^{-1}\mathbf{w}$.
    For $i = 1, \dots, j$ Do:
        $h_{i,j} = (\mathbf{w}, \mathbf{v}_i)$,
        $\mathbf{w} = \mathbf{w} - h_{i,j}\mathbf{v}_i$,
    End Do.
    Compute $h_{j+1,j} = \|\mathbf{w}\|_2$ and $\mathbf{v}_{j+1} = \mathbf{w}/h_{j+1,j}$.
    Define $Z_m = [\mathbf{z}_1, \dots, \mathbf{z}_m]$, $H_m = \{h_{i,j}\}_{1 \le i \le j+1; 1 \le j \le m}$.
End Do.
Compute $\mathbf{y}_m = argmin_y \|\beta\mathbf{e}_1 - H_m\mathbf{y}\|_2$, and $\mathbf{x}_m = \mathbf{x}_0 + Z_m\mathbf{y}_m$.
If $\|\hat{Q}_m^{-1}(\mathbf{b} - A\mathbf{x}_m)\|_2 < tol = 10^{-p}$ $p \in N$ then end, else $\mathbf{x}_0 \leftarrow \mathbf{x}_m$.
```

A possible strategy to choose $\hat{Q}_j^{-1}$ at each iteration is as follows. Should 10^{-p} be the tolerance fixed in Algorithm 4.1, the tolerance of the iterative method applied to solve systems (4.21), (4.22) is set to 10^{-q}, with $q = [p/2]$.

For the application at hand, by adopting this strategy one obtains that F-P-GMRES requires less CPU time to converge than P-GMRES (see Table 4.2).

As shown in Table 4.2, the application of the F-P-GMRES method (always based on the preconditioner provided by $\widehat{\mathcal{S}}_{hw}$) is very efficient. Comparing it with GMRES we see in fact that, although we have reduced the tolerance of the preconditioning step, consequently reducing the efficiency of the preconditioner, the number of global iterations does not increase. Thus the reduction in the computational cost at each iteration directly provides a reduction in the time needed to solve $A\mathbf{c} = \mathbf{b}$. Moreover, Table 4.2 shows that the considered algorithms behave well in the case of different geometries. In particular, we have considered test cases that are relevant for hemodynamics, the stenosed artery, and the carotid bifurcation.

5. Numerical results in case of physiological interest.

In the present section we show some numerical results about blood and oxygen dynamics in the carotid bifurcation, a site of relevant interest in biomedical applications. We point out that our main concern is to test the efficiency of the proposed iterative method and not to give significant results from a quantitative point of view. For this reason, the model

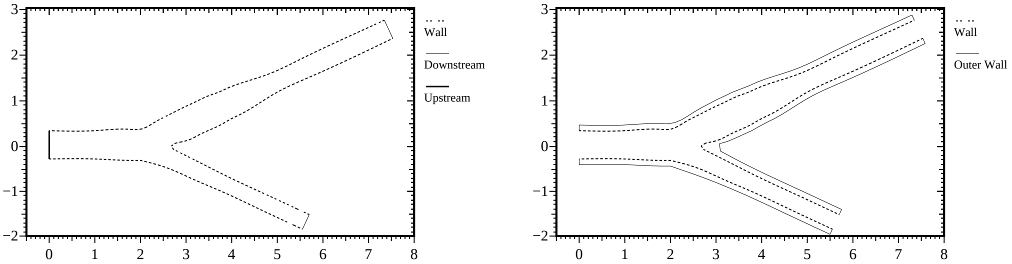


FIG. 5.1. *A 2D model for the lumen (right) and the wall (left) of a carotid bifurcation. The dashed line represents the fluid-wall interface, the thick solid line represents the inflow of the lumen, while the thin solid line represents the outflow of the lumen or the outer surface of the arterial wall.*

and the geometry are simplified (and in particular we deal with a 2D geometry obtained from data proposed in [7]), even if we include realistic numerical values for the physical constants at hand. In a forthcoming paper, on the basis of the good results illustrated in what follows, we plan to extend the methodology to more complex models and 3D geometries.

In the present section, as a realistic example, we apply our iterative substructuring method to the case of a carotid bifurcation, which is a preferential site of atherosclerosis, specifically in the external side of the carotid sinus. A current hypothesis about the possible reasons of this prominence to disease development is related to a reduction in the oxygen absorption by the sinus arterial wall, induced by the local flow patterns (see, e.g., [15]).

For the carotid test case we have considered stationary and pulsatile flow conditions. The selection of the boundary data is based on those proposed in [7]. In the stationary case, a parabolic profile is imposed on the upstream section, $u_x = ku_{\max}(R^2 - y^2)$, $u_y = 0$ where $k = 10.4$, $u_{\max} = 10$ cm s^{-1}, $R = 0.31$ cm, while null velocity, $\mathbf{u} = 0$, is prescribed on the wall boundary and zero traction force is prescribed on the outflow section, $(\rho\nu\nabla\mathbf{u} - PI)\cdot\mathbf{n} = 0$ (see Figure 5.1 for a description of the boundaries $\partial\Omega_f$ and $\partial\Omega_w$). The pulsatile case differs from the stationary one for the inflow conditions, $u_x = kg(t)(R^2 - y^2)$, $u_y = 0$, where $g(t)$, represented in Figure 5.2, describes the blood flow at the entrance of the carotid during a heartbeat. In both cases, a constant blood kinematic viscosity $\nu = 0.033$ cm^2 s^{-1} has been chosen.

With regard to the concentration in the lumen, a reference concentration of oxygen, $C_0 = 0.04$ g cm^{-3}, has been prescribed on the inflow, condition (2.2_3) has been prescribed on the wall boundary, and $\nabla C \cdot \mathbf{n} = 0$ has been prescribed on the outflow boundary. This condition is also prescribed on the outer wall in Ω_w. The oxygen dynamics have been simulated choosing the diffusivity $\mu_f = \mu_w = 10^{-5}$cm^2 s^{-1}, and $\zeta = 10^{-4}(1+|\boldsymbol{\sigma}|)$ cm s^{-1}, where $\boldsymbol{\sigma}$ is the shear stress exerted by the blood on the wall. From the numerical viewpoint, we consider Ω_f discretized with 20494 nodes and Ω_w discretized with 15030 nodes (see Figure 5.3). With regard to boundary conditions, Dirichlet ones are imposed in an essential way, while Neumann and Robin conditions are imposed in a natural way, as is customary in the framework of Galerkin discretization and, in particular, in the framework of the FEM; see [12]. Finally, the numerical method adopted ensures a first order accuracy in time, being based on an implicit Euler scheme. The finite elements adopted are piecewise linear. The Navier–Stokes solver is based on the so-called Yosida method (see [10]) on $\mathbb{P}^1 iso\mathbb{P}^2$ elements.

Concerning the numerical results, the velocity field in the bifurcation shows a recirculation zone (see Figures 5.3 and 5.4) both in the steady and pulsatile cases.

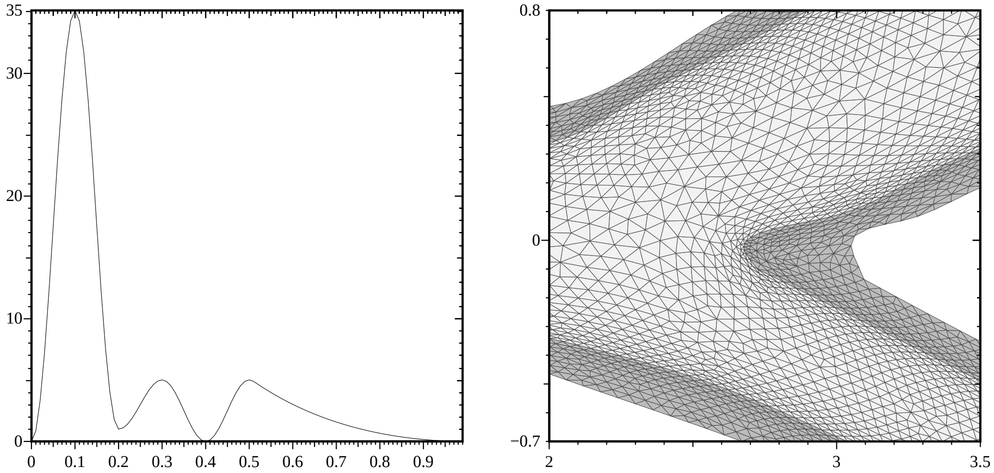


FIG. 5.2. *Maximum inflow velocity (cm/s) during a heartbeat (left) and computational grid for the 2D carotid simulation (right).*

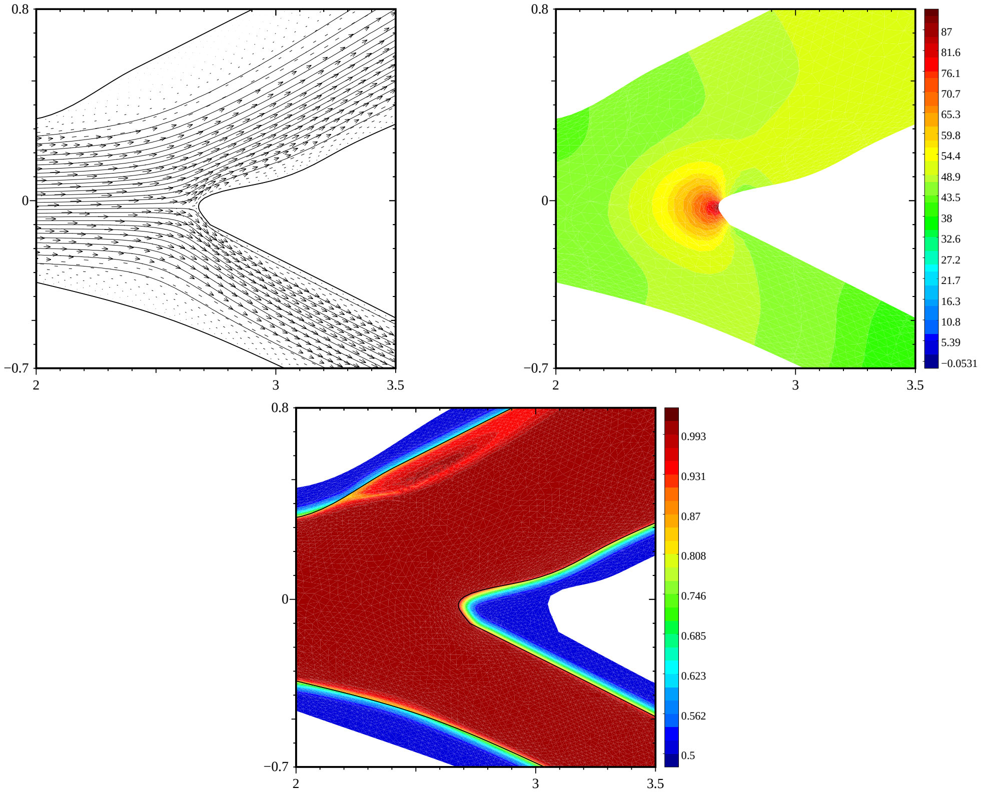


FIG. 5.3. *Velocity field (top-left), pressure field $(P - P_{ext})$ (top-right), and oxygen concentration (bottom) in the carotid bifurcation in the steady case.*

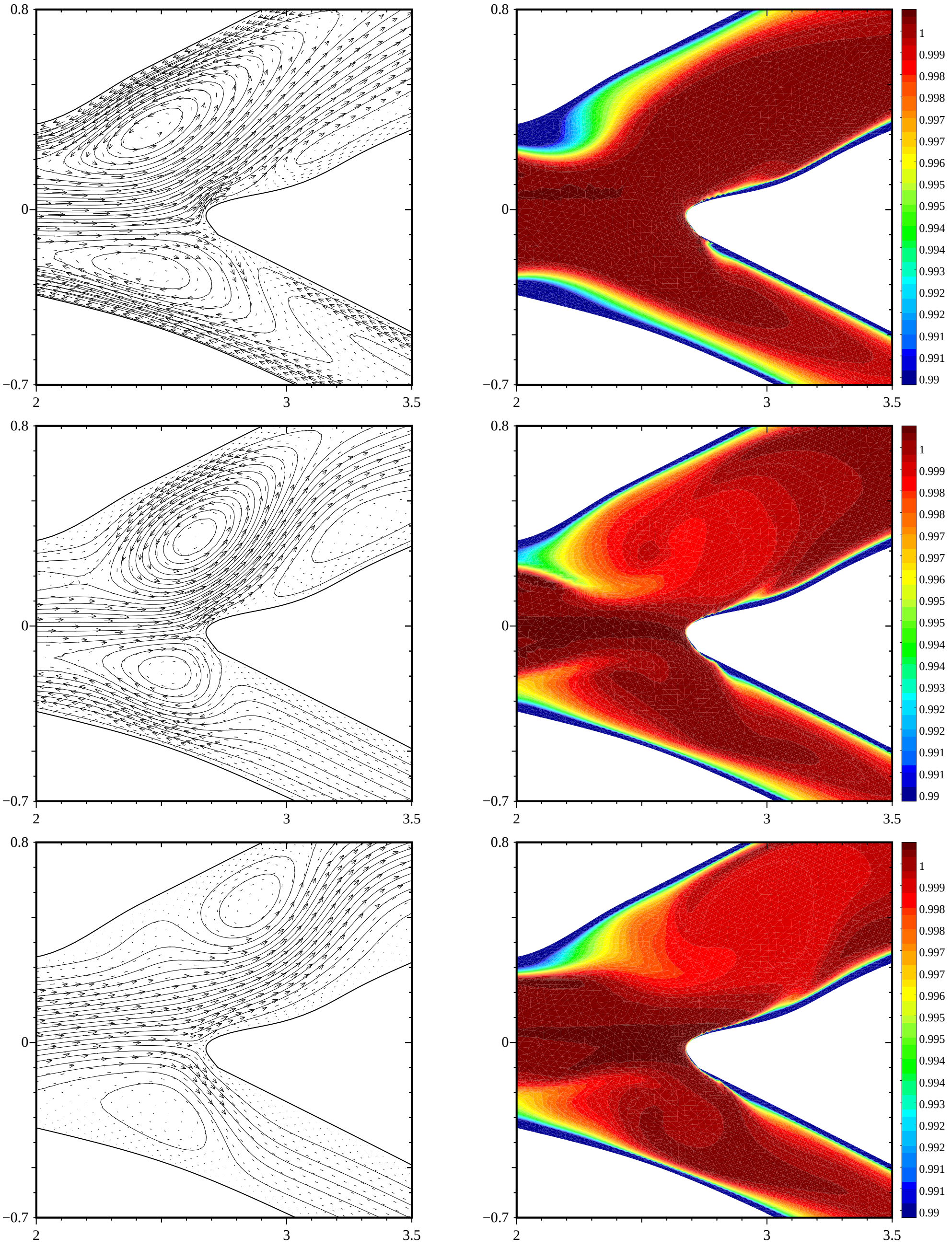


FIG. 5.4. *Blood velocity and oxygen concentration in the carotid bifurcation in the pulsatile case at* 1/5, 2/5, 3/5 *of a heartbeat. In this case only the concentration in* Ω_f *is visualized. The recirculation zone which possibly inhibits the oxygen absorption is evident in the different instants of the heartbeat.*

The presence of recirculations induces in the steady case a sensibly lower oxygen concentration in proximity of the sinus wall and, consequently, the reduction of the oxygen absorption by the wall, which could be related to the disease development.

In the pulsatile case the flow pattern is more complex: a recirculation zone, moving upstream and downstream (yielding a lack of oxygen), is still present next to the external side of the (inner) carotid sinus. In this case, the oxygen absorption is further reduced by the low permeability induced by the small values of $|\boldsymbol{\sigma}|$. Indeed, the flow dynamics in the carotid sinus induce small and oscillating shear stresses which actually determine a permeability reduction according to the law $\zeta = 10^{-4}(1 + |\boldsymbol{\sigma}|)$ cm s^{-1} (see also [18]).

6. Conclusions. In this paper we analyzed from the numerical viewpoint the solution of problem (2.2). The multidomain and the heterogeneous nature of this problem induced us to rely on iterative methods to compute the global solution. Consequently, in the framework of iterative substructuring methods, we introduced special SP operators in order to take into account conditions (2.2_3, 2.2_4). The analysis of the properties of these operators allowed us to apply domain decomposition theory in order to prove the convergence of the iterative methods adopted to solve problem (2.2). Additionally, we proved that the mesh size does not affect the rate of convergence of the methods. Finally the problem of reducing the computational time was taken into account. To this end, we considered a variant of the GMRES algorithm, the so called F-P-GMRES, and we verified, with numerical tests, that this method is particularly efficient in solving the problems arising from the blood solute dynamics. The simplified test case considered, namely the carotid bifurcation, has been discussed from both the numerical and the physiological point of view.

REFERENCES

[1] C.G. Caro, J.M. Fitzgerald, and R.C. Schroter, *Atheroma and wall shear: Observation, correlation and proposal of a shear dependent mass transfer mechanism of atherogenesis*, Proc. Roy. Soc. London Ser. A, 177 (1971), pp. 109–159.

[2] G. Karner and K. Perktold, *Effect of endothelial injury and increased blood pressure on albumin accumulation in the arterial wall: A numerical study*, J. Biomech., 33 (2000), pp. 709–715.

[3] G. Karner, K. Perktold, and H.P. Zehentner, *Mass transport in large arteries and through the arterial wall*, in Intra and Extracorporeal Cardiovascular Fluid Dynamics, WIT Press, Southampton, Boston, 2000, pp. 209–247.

[4] K.H. Keller, *Effect of fluid shear on mass transport in flowing blood*, Fed. Proc., 30 (1971), pp. 1591–1599.

[5] J.L. Lions and E. Magenes, *Problèmes aux Limites non Homogènes et Applications*, Vol. 1, Dunod, Paris, 1968.

[6] J.L. Lions and E. Magenes, *Problèmes aux Limites non Homogènes et Applications*, Vol. 2, Dunod, Paris, 1968.

[7] P. Ma, X. Li, and D.N. Ku, *Convective mass transfer at the carotid bifurcation*, J. Biomech., 30 (1997), pp. 565–571.

[8] J.A. Moore and C.R. Ethier, *Oxygen mass transfer calculations in large arteries*, J. Biomech. Engrg., 119 (1997), pp. 469–475.

[9] A. Quarteroni, R. Sacco, and F. Saleri, *Numerical Mathematics*, Springer-Verlag, New York, 2000.

[10] A. Quarteroni, F. Saleri, and A. Veneziani, *Analysis of the Yosida method for the incompressible Navier-Stokes equations*, J. Math. Pure Appl., 78 (1999), pp. 473–503.

[11] A. Quarteroni, M. Tuveri, and A. Veneziani, *Computational vascular fluid dynamics: Problems, models and methods*, Comput. Visual. Sci., 2 (2000), pp. 163–197.

[12] A. Quarteroni and A. Valli, *Numerical Approximation of Partial Differential Equations*, Springer-Verlag, Berlin, 1994.

[13] A. Quarteroni and A. Valli, *Domain Decomposition Methods for Partial Differential Equations*, Oxford University Press, Oxford, UK, 1999.

[14] A. Quarteroni, A. Veneziani, and P. Zunino, *Mathematical and numerical modeling of solute dynamics in blood flow and arterial walls*, SIAM J. Numer. Anal., 39 (2001), pp. 1488–1511.

[15] G. Rappitsch and K. Perktold, *Pulsatile albumin transport in large arteries: A numerical simulation study*, J. Biomech. Engrg., 118 (1996), pp. 511–519.

[16] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS, Boston, 1996.

[17] R. Temam, *Navier-Stokes Equations. Theory and Numerical Analysis*, 3rd ed., North-Holland, Amsterdam, 1984.

[18] C. Zarins, D. Giddens, B. Bharadvaj, V. Sottiurai, R. Mabon, and S. Glagov, *Carotid bifurcation atherosclerosis: Quantitative correlation of plaque formation with velocity profiles and wall shear stresses*, Circ. Res., 53 (1993), pp. 502–514.

SIAM J. SCI. COMPUT.
Vol. 23, No. 6, pp. 1981–2007

CONVERGENCE OF NONCONVERGENT IRK DISCRETIZATIONS OF OPTIMAL CONTROL PROBLEMS WITH STATE INEQUALITY CONSTRAINTS[*]

J. T. BETTS[†], N. BIEHN[‡], AND S. L. CAMPBELL[§]

Abstract. It has been observed that optimization codes are sometimes able to solve inequality state constrained optimal control problems with discretizations which do not converge when used as integrators on the constrained dynamics. Understanding this phenomenon could lead to a more robust design for direct transcription codes as well as better test problems. This paper examines how this phenomenon can occur. The difference between solving index 3 differential algebraic equations (DAEs) using the trapezoid method in the context of direct transcription for optimal control problems and a straightforward implicit Runge–Kutta (IRK) formulation of the same trapezoidal discretization is analyzed. It is shown through numerical experience and theory that the two can differ as much as $O(1/h^3)$ in the control. The optimization can use a small sacrifice in the accuracy of the states in the early stages of the trapezoidal method to produce better accuracy in the control, whereas more precise solutions converge to an incorrect solution. Convergence independent of the index of the constraints is then proven for one class of problems. The theoretical results are used to explain computational observations.

Key words. optimal control, differential algebraic equations, numerical discretizations

AMS subject classifications. 49M25, 65K10, 65L80

PII. S1064827500383044

1. Introduction.

Many optimal control problems contain inequality path constraints on the states and/or controls. Along the constraints, the resulting system becomes a differential algebraic equation (DAE) even if the original dynamics were an ordinary differential equation (ODE). Depending on which set of variables are constrained, a low or high index DAE results. For more on DAEs and their indices, see [7]. It has been observed that optimization codes are sometimes able to solve state constrained optimal control problems with discretizations which do not converge when used as an integrator on the constrained dynamics. Understanding this phenomenon could lead to a more robust design for direct transcription codes as well as better test problems.

This paper studies the differences between solving index 3 differential algebraic equations using the trapezoid (TR) method, which arise because of inequality constraints in the direct transcription of optimal control problems versus a straightforward implementation of the implicit Runge–Kutta (IRK) formulation of the trapezoid rule. The index 3 case is studied because it often occurs in applications and also because the behavior of interest here is already present [6].

The direct transcription software SOCS (Sparse Optimal Control Software) developed by Boeing [1, 2, 3, 5] is used throughout the paper. The results obtained

[*] Received by the editors December 28, 2000; accepted for publication (in revised form) October 2, 2001; published electronically February 27, 2002.

http://www.siam.org/journals/sisc/23-6/38304.html

[†] Mathematics and Engineering Analysis, The Boeing Company, P.O. Box 3707, MS 7L-21, Seattle, WA 98124-2207 (John.T.Betts@boeing.com).

[‡] PROS Revenue Management, 3100 Main Street, Suite 900, Houston, TX 77021 (nbiehn@prosrm.com).

[§] Department of Mathematics, North Carolina State University, Raleigh, NC 27695-8205 (slc@math.ncsu.edu). This author's research was supported in part by NSF grants DMS-9714811, DMS-9802259, and INT-9605114.

here are relevant to other direct transcription algorithms with similar strategies. In particular, SOCS passes the discretized problem directly to the nonlinear program (NLP) solver and does not first integrate the dynamics to reduce the dimension of the problem. We shall see that this difference is not just a question of whether to solve a larger sparse problem versus a smaller dense problem, as it is often portrayed. Rather this difference also alters questions of convergence in a fundamental way. The discretization formulas normally used by SOCS are mathematically equivalent to the Lobatto IIIA formulas TR and Hermite–Simpson (HS). These methods are symmetric, stiffly stable, and produce desirable sparsity patterns in the Jacobians of the defect constraints in the associated NLP. They are implemented as collocation methods in SOCS. However, neither of these methods converges when applied as an integrator to an index 3 DAE [6]. This paper is concerned primarily with TR. Computational experience and a more technical analysis show the same phenomenon occurring when using HS.

There are other discretizations that one might consider besides TR and HS. However, switching the discretization in an industrial-grade sparse code like SOCS is a nontrivial task since efficiency requires that the Jacobians have their sparsity structure explicitly exploited. Also, other choices which are not collocation methods lead to other difficulties. Given that the current discretizations work so well on such a large variety of problems, it is important to determine the classes of problems on which they can be used.

The TR method implemented as an IRK converges for DAEs of index 0, 1, and 2. For problems with an index greater than 2, simple examples can be constructed for which the trapezoid method converges to an incorrect solution [6]. Numerical experience has shown that when used in SOCS, TR can converge to the true optimal solution, while the straightforward IRK formulation does not [6]. In fact, we have observed TR working on index 4, 5, and, in the case of boundary controlled PDEs, on even higher index problems. There are two questions to be resolved. One is how SOCS is able to converge. That is, what makes convergence possible? The second is why SOCS converges. There has been prior research on the convergence of discretizations of optimization problems. We cite here only [9, 10, 13]. Additional references are in section 5. Previous work on the state constrained problem always assumes that the discretization converges on a problem with the constraint holding identically. This type of analysis does not apply to the particular phenomena we are investigating here. Also, as noted later, analysis which uses convergence of the discrete multipliers is also not appropriate.

In the next section we give the formulation of the optimal control problem and the associated DAE along the constraint. Section 3 sets up and proves the first main theorem of the paper which theoretically predicts the difference between the direct transcription and IRK solutions. Numerical experiments and discussion are given in section 4. A greatly condensed version of this first portion of this paper with less discussion and no proofs appears in [4]. Section 5 develops the convergence of the TR discretization when index 3 state constraints are active.

In order to avoid confusion between the different uses of TR we shall use SOCS TR for the SOCS solution using the TR discretization and IRK TR for the TR solution implemented as an IRK when constraints are active.

Finally, there is an application motivating our investigations which is the optimization of tool paths. In these problems, complexity and problem size sometimes forces the termination of the iterations in SOCS at coarser meshes than those required for termination according to the termination criteria. For this reason we are very interested not only in behavior on sufficiently fine meshes but also on "coarser" meshes. Also, in some problems highly accurate constraint satisfaction is important. For these problems the SOCS philosophy of satisfying all constraints on every mesh level and at every mesh point is important. This should be contrasted with methods which only impose constraints at some mesh points or asymptotically.

2. The control problem.

As noted in [4, 6] convergence of the optimizer in the presence of high index constraints occurs already with linear problems. For our purposes here it suffices to consider optimal control problems with an objective function,

$$\int_{t_0}^{t_f} L(y,u,t)\,dt \tag{1a}$$

with state equations

$$y'(t) = f(y(t),u(t),t) \tag{1b}$$

and inequality constraints

$$g(y(t),u(t),t) \geq g_L, \tag{1c}$$

$$y(t_0) = y_0. \tag{1d}$$

In SOCS the continuous optimal control problem is transcribed to a finite-dimensional NLP. Let $y(t_k) = y_k$ and $u(t_k) = u_k$ where $T_N = \{t_i\}_0^N$ with $t_0 < \cdots < t_N = t_f$. Then

$$x = [y_0^T, u_0^T, y_1^T, u_1^T, \ldots, y_N^T, u_N^T]^T \tag{2}$$

are the NLP variables with $N+1$ mesh points. Using TR, we approximate (1b) using the defect constraints

$$0 = y_k - y_{k-1} - \frac{h_k}{2}(f(y_k,u_k,t_k) + f(y_{k-1},u_{k-1},t_{k-1})), \qquad 1 \leq k \leq N, \tag{3}$$

with $h_k = t_k - t_{k-1}$. The objective function (1a) is approximated using a trapezoidal quadrature. The NLP is solved using a sparse sequential quadratic programming (SQP) algorithm whose solution is a discrete approximation to the continuous optimal control problem [3]. A sophisticated mesh refinement algorithm then refines the mesh for the next iteration [5].

Consider the following optimal control problem from [6]. SOCS correctly finds the solution.

$$\min_u \int_0^4 x^2(t) + 10^{-3}u^2(t)\,dt, \tag{4a}$$

$$x' = v, \tag{4b}$$

$$v' = u, \tag{4c}$$

$$0 \leq x - 15 + (t-4)^4, \tag{4d}$$

$$x(0) = 5, \ v(0) = 0. \tag{4e}$$

The constraint (4d) is active on an interval including $\left[\frac{34}{15},\ 4\right]$. Since all variables are determined once the constraint is active, we examine the following index 3 DAE in

(x, v, u) on $[t_0, t_f] = \left[\frac{34}{15},\ 4\right]$:

$$x' = v, \tag{5a}$$

$$v' = u, \tag{5b}$$

$$x = 15 - (t-4)^4, \tag{5c}$$

$$x(t_0) = 15 - (t_0 - 4)^4, \tag{5d}$$

$$v(t_0) = -4(t_0 - 4)^3, \tag{5e}$$

$$u(t_0) = -12(t_0 - 4)^2. \tag{5f}$$

One can also attempt to numerically solve (5) using the TR directly on the DAE. Figures 1 and 2 give plots of the u obtained by each of the methods with $N = 10$.

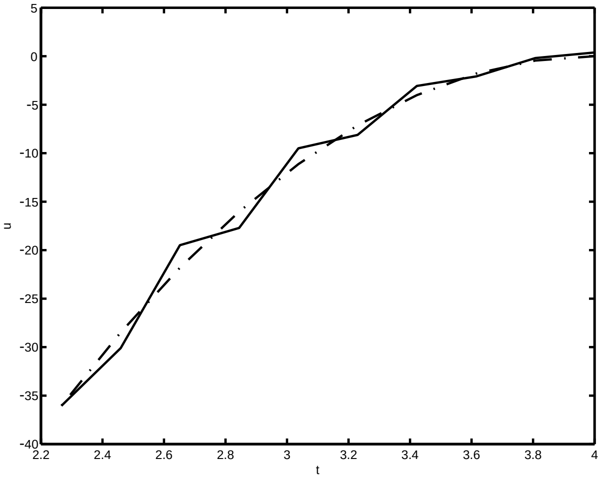


FIG. 1. *Control given by SOCS TR and the true solution.*

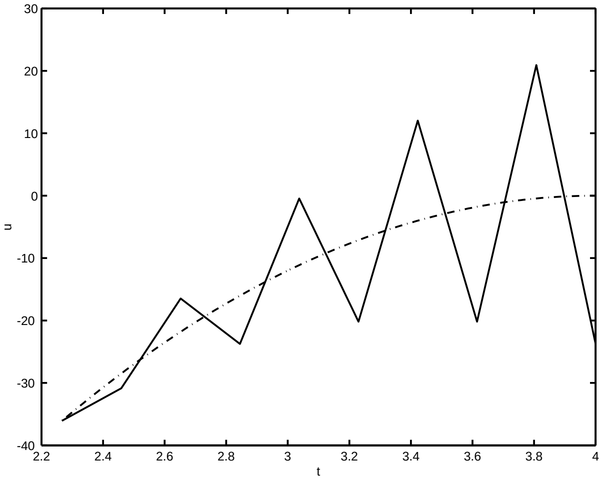


FIG. 2. *Control given by IRK TR and the true solution.*

The SOCS solution approximates the true solution while applying TR directly to the DAE (5) does not. This behavior persists at larger N. A rigorous understanding of how these two methods which appear to be mathematically equivalent can differ by so much is the goal of this paper.

3. Analysis of an index 3 system.

3.1. Background and perturbation theorem. Since the numerical solutions for the control were differing by large amounts, there must be a fundamental difference between using TR directly on the DAE and in a direct transcription formulation. The NLP solution is known to satisfy the given constraints up to a tolerance which is much looser than the accuracy to which the IRK equations are usually solved. It was suggested in [6] that this NLP "slop" provides one possible explanation for the large differences in the computed control between IRK TR and SOCS TR. We will see that the "slop conjecture" can account for computational observations on fine meshes when SOCS is close to convergence but cannot explain what is observed on coarser meshes. A careful analysis of an index 3 problem proves enlightening.

Consider the following state equations with constraints:

$$x' = Ax + Bu, \tag{6a}$$

$$0 \le Cx - \widehat{g}(t), \tag{6b}$$

where A is $m \times m$, B is $m \times q$, C is $r \times m$, and $\widehat{g}(t)$ is $r \times 1$. Suppose that the optimal solution pushes the state x onto the constraint (6b) over an interval $[a, b]$. This gives us a DAE (6) in (x, u) along this interval. Let's first formulate the DAE and then discretize the DAE using TR.

The resulting DAE system can be written as

$$Ey' = \widehat{A}y - g(t) \tag{7}$$

with

$$E = \begin{bmatrix} I_m & 0 \\ 0 & 0 \end{bmatrix}, \quad \widehat{A} = \begin{bmatrix} A & B \\ C & 0 \end{bmatrix}, \quad g(t) = \begin{bmatrix} 0 \\ \widehat{g}(t) \end{bmatrix}, \quad y = \begin{bmatrix} x \\ u \end{bmatrix}.$$

Then TR for (7) implemented as an IRK is

$$EY_1' - \widehat{A}y_{n-1} = -g(t_{n-1}), \tag{8a}$$

$$EY_2' - \widehat{A}y_{n-1} - \frac{h}{2}\widehat{A}(Y_1' + Y_2') = -g(t_n), \tag{8b}$$

$$y_n = y_{n-1} + \frac{h}{2}(Y_1' + Y_2'), \tag{8c}$$

where the Y_i' are the stage derivatives for $i = 1, 2$. In this section we will assume h is constant since that suffices for examining how convergence is possible. h will be allowed to vary in subsequent sections. Exploiting the special structure of E we let $Y_i' = [\begin{smallmatrix} X_i' \\ U_i' \end{smallmatrix}]$, $i = 1, 2$. Then (8) becomes

$$X_1' = Ax_{n-1} + Bu_{n-1}, \tag{9a}$$

$$0 = Cx_{n-1} - \widehat{g}(t_{n-1}), \tag{9b}$$

$$X_2' = Ax_{n-1} + Bu_{n-1} + \frac{h}{2}(A(X_1' + X_2') + B(U_1' + U_2')), \tag{9c}$$

$$0 = Cx_{n-1} + \frac{h}{2}C(X_1' + X_2') - \widehat{g}(t_n), \tag{9d}$$

$$x_n = x_{n-1} + \frac{h}{2}(X_1' + X_2'), \tag{9e}$$

$$u_n = u_{n-1} + \frac{h}{2}(U_1' + U_2'). \tag{9f}$$

Notice that (9e)–(9f) can be substituted into (9c)–(9d) to form the new equations

$$X_1' = Ax_{n-1} + Bu_{n-1}, \tag{10a}$$

$$X_2' = A\left(x_{n-1} + \frac{h}{2}(X_1' + X_2')\right) + Bu_n, \tag{10b}$$

$$x_n = x_{n-1} + \frac{h}{2}(X_1' + X_2'), \tag{10c}$$

$$0 = Cx_n - \widehat{g}(t_n). \tag{10d}$$

The information from (9b) was obtained from the previous step and hence may be eliminated.

The direct transcription software SOCS must satisfy (11a)–(11b) in $\widetilde{x}$ and $\widetilde{u}$ at every mesh point for a solution to be considered feasible.

$$\widetilde{x}_n - \widetilde{x}_{n-1} - \frac{h}{2}(A\widetilde{x}_{n-1} + B\widetilde{u}_{n-1} + A\widetilde{x}_n + B\widetilde{u}_n) + \delta_{n-1} = 0, \tag{11a}$$

$$C\widetilde{x}_n - \widehat{g}_n + \epsilon_{n-1} = 0. \tag{11b}$$

The parameters δ_n and ϵ are included as tolerances for the accuracy to which we solve (11a) and (11b). We initially view these as the tolerances in the NLP solve. Let us define

$$\widetilde{X}_1' = A\widetilde{x}_{n-1} + B\widetilde{u}_{n-1}, \tag{12}$$

$$\widetilde{X}_2' = A\widetilde{x}_n + B\widetilde{u}_n \tag{13}$$

so that we may now write (11a) as

$$\widetilde{x}_n - \widetilde{x}_{n-1} - \frac{h}{2}(\widetilde{X}_1' + \widetilde{X}_2') + \delta_{n-1} = 0. \tag{14}$$

Solving for $\widetilde{x}_n$ in (14) and substituting in (13) we have

$$\widetilde{X}_1' = A\widetilde{x}_{n-1} + B\widetilde{u}_{n-1}, \tag{15}$$

$$\widetilde{X}_2' = A\left(\widetilde{x}_{n-1} + \frac{h}{2}(\widetilde{X}_1' + \widetilde{X}_2')\right) + B\widetilde{u}_n + A\delta_{n-1}. \tag{16}$$

Taking (15), (16) and (14), (11b), the SOCS discretization looks like,

$$\widetilde{X}_1' = A\widetilde{x}_{n-1} + B\widetilde{u}_{n-1}, \tag{17a}$$

$$\widetilde{X}_2' = A\left(\widetilde{x}_{n-1} + \frac{h}{2}(\widetilde{X}_1' + \widetilde{X}_2')\right) + B\widetilde{u}_n + A\delta_{n-1}, \tag{17b}$$

$$\widetilde{x}_n = \widetilde{x}_{n-1} + \frac{h}{2}(\widetilde{X}_1' + \widetilde{X}_2') + \delta_{n-1}, \tag{17c}$$

$$C\widetilde{x}_n = \widehat{g}_n + \epsilon_{n-1}. \tag{17d}$$

Comparing (10) to (17) we see that we may consider the SOCS solution as a perturbation of the IRK solution. If we subtract the IRK system from the SOCS discretization we get a system for the difference between the SOCS TR and the IRK TR solutions in the form

$$Gz_n = Hz_{n-1} + \gamma_{n-1} \tag{18}$$

with

$$G = \begin{bmatrix} I & -\frac{h}{2} & -\frac{h}{2} & 0 \\ 0 & I - \frac{h}{2}A & -\frac{h}{2}A & -B \\ 0 & 0 & I & 0 \\ C & 0 & 0 & 0 \end{bmatrix}, \quad H = \begin{bmatrix} I & 0 & 0 & 0 \\ A & 0 & 0 & 0 \\ A & 0 & 0 & B \\ 0 & 0 & 0 & 0 \end{bmatrix}, \tag{19}$$

$$\gamma_n = \begin{bmatrix} \delta_n \\ A\delta_n \\ 0 \\ \epsilon_n \end{bmatrix}, \quad z_n = \begin{bmatrix} \widetilde{x}_n - x_n \\ \widetilde{X}_2' - X_2' \\ \widetilde{X}_1' - X_1' \\ \widetilde{u}_n - u_n \end{bmatrix}. \tag{20}$$

Suppose that G is invertible and solve (18) for z_n,

$$z_{n+1} = G^{-1}Hz_n + G^{-1}\gamma_n. \tag{21}$$

The solution of (21) is $z_k = (G^{-1}H)^k z_0 + \sum_{j=0}^{k-1}(G^{-1}H)^{k-j-1}G^{-1}\gamma_j$. We assume that $z_0 = 0$ and obtain the difference between the IRK and SOCS TR solutions starting from the same initial conditions as

$$z_k = \sum_{j=0}^{k-1}(G^{-1}H)^{k-j-1}G^{-1}\gamma_j. \tag{22}$$

The analysis and discussion that follows differs from the usual error analysis of IRK methods in that it is not enough that we convert (22) to an order estimate since we need to know the actual size of the perturbation z_k and we need to consider mesh widths that may not be small.

Equation (22) holds for all DAEs of index 1 or higher. We know from previous theory [6] that IRK TR does converge for DAEs of index 1 and 2 but not for DAEs of index greater than 2. We assume the DAE (7) is an index 3 problem. For (7) to be index 3 we must have that CB is singular. Since for time invariant problems we can decouple the different index subsystems using linear time invariant coordinate changes, we may assume in our analysis that $CB = 0$. Without loss of generality we may further assume that C is full row rank, B is full column rank, and there exist matrices $\widehat{P}$ and $\widehat{Q}$ so that

$$\widehat{P}C\widehat{Q}^{-1} = \begin{bmatrix} I & 0 \end{bmatrix}, \quad \widehat{Q}B = \begin{bmatrix} 0 \\ I \end{bmatrix}, \tag{23}$$

where $\widehat{Q}$ represents a coordinate change. Thus we may assume that

$$C = \begin{bmatrix} I & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ I \end{bmatrix}. \tag{24}$$

A linear time invariant index 3 DAE with $CB = 0$ must also have CAB invertible. Define

$$Q = I - \frac{h}{2}A \tag{25}$$

and let

$$Q = \begin{bmatrix} Q_1 & Q_2 \\ Q_3 & Q_4 \end{bmatrix}, \quad A = \begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix}, \tag{26}$$

where the partitioning conforms to C and B. From the special form given for C and B we know that CAB is invertible if and only if A_2 is invertible. Q is invertible for all but a finite number of nonzero h. The value of $Q_2 = C(I - \frac{h}{2}A)^{-1}B$ is given analytically in terms of A by

$$Q_2 = -2hA_2(-4I + 2hA_4 + 2hA_1 - h^2A_1A_4 + h^2A_2A_3)^{-1},$$

which is invertible for small nonzero h since A_2 is invertible. Then Q_2 is invertible for all but a finite number of h.

Theorem 3.1. *Suppose that the DAE* (7) *is index* 3, *h is constant, and* (10), (17), *and* (24) *hold. Assume that CAB and $C(I - \frac{h}{2}A)^{-1}B$ are invertible. Let $W = Q_4Q_2^{-1}Q_1 - Q_3, Y = Q_4Q_2^{-1} - 2W$. Then the difference in the states between the IRK TR and SOCS TR solutions is*

$$\begin{aligned} \bar{x}_k - x_k = \sum_{j=0}^{k-1}(-1)^{k-1-j} \Bigg(& \begin{bmatrix} 0 & 0 \\ -Q_4Q_2^{-1} & I \end{bmatrix} \delta_j \\ & + \begin{bmatrix} 0 & 0 \\ -\frac{h}{2}W & 0 \end{bmatrix} A\delta_j + \begin{bmatrix} 0 \\ -Y + Q_4Q_2^{-1} \end{bmatrix} \epsilon_j \Bigg). \end{aligned} \tag{27}$$

The difference between the controls is

$$\begin{aligned} \bar{u}_k - u_k = \sum_{j=0}^{k-1}(-1)^{k-1-j}\frac{2}{h} \Bigg\{ & [-2(k-j-1)Q_4Q_2^{-1} - Q_2^{-1} \quad 2(k-j-1)I]\delta_j \\ & + \frac{h}{2}[-2(k-j-1)W - Q_2^{-1}Q_1 \quad - I]A\delta_j \\ & + [-2(k-j-2)Y + 2Q_2^{-1}Q_1 + 2(k-j-1)Q_4Q_2^{-1}]\epsilon_j \Bigg\}. \end{aligned} \tag{28}$$

Note that the controls can differ considerably depending on the size of δ and ϵ. If we suppose that we are on the interval $[0, 1]$, then the value of k at the end of the interval would act like $1/h$. Q_2^{-1} may possess a $1/h$ factor as well. Thus a single δ_j will introduce a difference between the controls of at least

$$\bar{u}_k - u_k \sim O\left(\frac{1}{h^3}\|\delta_j\|\right).$$

The difference in the controls for a constant perturbation δ is also $O(h^{-3})$ due to cancellation. The worst case given by δ_j with alternating sign is $O(h^{-4})$. However, this is not possible on coarse meshes because of the one-sided nature of constraint chatter.

Proof of Theorem 3.1. Since B is full column rank and C is full row rank, if h is small enough so that $I - \frac{h}{2}A$ is invertible, then G is invertible. Thus G is invertible for all but a finite number of nonzero h. Row operations yield

$$G^{-1} = \begin{bmatrix} I - \frac{h}{2}QBPC & \frac{h}{2}(Q - \frac{h}{2}QBPCQ) & \frac{h}{2}(Q - \frac{h}{2}QBPCQ) & \frac{h}{2}QBP \\ -QBPC & -\frac{h}{2}QBPCQ + Q & -\frac{h}{2}QBPCQ + \frac{h}{2}QA & QBP \\ 0 & 0 & I & 0 \\ -PC & -\frac{h}{2}PCQ & -\frac{h}{2}PCQ & P \end{bmatrix},$$

where $P = \frac{2}{h}(CQB)^{-1}$. To simplify, note the identities

$$\begin{gathered} Q = I + \frac{h}{2}QA, \\ -\frac{h}{2}PCQB = -I, \\ \left(-\frac{h}{2}QBPCQ + \frac{h}{2}QA\right)B = -B. \end{gathered}$$

Let $N = I + \frac{h}{2}A$. Then

$$G^{-1}H = \begin{bmatrix} (I - \frac{h}{2}QBPC)QN & 0 & 0 & 0 \\ (QA - QBPCQ)N & 0 & 0 & -B \\ A & 0 & 0 & B \\ -PCQN & 0 & 0 & -I \end{bmatrix}.$$

To evaluate (22) we need to compute products $(G^{-1}H)^k$. The special forms of C and B allow us to simplify the different terms in $G^{-1}H$. The $(1,1)$ term is

$$(G^{-1}H)_{1,1} = \begin{bmatrix} 0 & 0 \\ -2Q_4Q_2^{-1}Q_1 + Q_4Q_2^{-1} + 2Q_3 & -I \end{bmatrix}. \tag{29}$$

As we compute a power of $G^{-1}H$, the $(1,1)$ term is itself powered. Let $(G^{-1}H)^k_{i,j}$ be the i,j term of $(G^{-1}H)^k$. Examining (29), it is clear that

$$(G^{-1}H)^k_{1,1} = (-1)^k \begin{bmatrix} 0 & 0 \\ 2Q_4Q_2^{-1}Q_1 - Q_4Q_2^{-1} - 2Q_3 & I \end{bmatrix}.$$

Next we examine powers of the $(4,1)$ term, since the other terms will depend upon it. Let

$$\begin{aligned} W &= Q_4Q_2^{-1}Q_1 - Q_3, \\ Y &= Q_4Q_2^{-1} - 2W, \\ Z &= 2Q_2^{-1}Q_1 - Q_2^{-1}. \end{aligned}$$

A direct calculation gives

$$(G^{-1}H)^k_{4,1} = \frac{2}{h}[(-1)^{k+1}2(k-1)Y + (-1)^k Z \qquad (-1)^k 2kI],$$

$$(G^{-1}H)^k_{3,1} = \begin{bmatrix} (-1)^k A_2 Y & (-1)^{k+1}A_2 \\ (-1)^k(A_4Y + (k-2)\frac{4}{h}Y - \frac{2}{h}Z) & (-1)^{k+1}(A_4 + I(k-1)4/h) \end{bmatrix}.$$

Then

$$QBPCQN = \frac{2}{h}\begin{bmatrix} 2Q_1 - I & 2Q_2 \\ Q_4 Z & 2Q_4 \end{bmatrix}.$$

In addition, we see that $QAN = \frac{4}{h}Q - \frac{4}{h}I - A$, which leads to

$$QAN - QBPCQN = \begin{bmatrix} -\frac{2}{h}I - A_1 & -A_2 \\ \frac{4}{h}Q_3 - A_3 - \frac{2}{h}Q_4 Z & -\frac{4}{h}I - A_4 \end{bmatrix}.$$

Thus

$$(G^{-1}H)^k_{2,1} = \begin{bmatrix} (-1)^{k+1}A_2 Y & (-1)^k A_2 \\ (-1)^k((A_4 + \frac{4(k-1)}{h}I)(2W - Q_4 Q_2^{-1}) + \frac{2}{h}Z) & (-1)^k(A_4 + \frac{4k}{h}I) \end{bmatrix}.$$

We next compute $(G^{-1}H)^k G^{-1}$, which is also done term by term. Some key cancellations can be made. For example, by multiplying both sides of $(I - \frac{h}{2}A)Q = I$ on the right by

$$\begin{bmatrix} 0 & -I \\ 0 & Q_2^{-1}Q1 \end{bmatrix},$$

we see that $\frac{h}{2}A_2(Q_4 Q_2^{-1}Q_1 - Q_3) = I$. Below we give the 16 different terms of $(G^{-1}H)^k G^{-1}$.

1. (1, 1)

$$(-1)^k \begin{bmatrix} 0 & 0 \\ -Q_4 Q_2^{-1} & -I \end{bmatrix}.$$

2. (1, 2), (1, 3)

$$(-1)^k \begin{bmatrix} 0 & 0 \\ -\frac{h}{2}W & 0 \end{bmatrix}.$$

3. (1, 4)

$$(-1)^k \begin{bmatrix} 0 \\ -Y + Q_4 Q_2^{-1} \end{bmatrix}.$$

4. (2, 1)

$$(-1)^k \begin{bmatrix} -A_2 Q_4 Q_2^{-1} & A_2 \\ -(A_4 Q_4 Q_2^{-1} + \frac{4k}{h}Q_4 Q_2^{-1} + \frac{2}{h}Q_2^{-1}) & A_4 + \frac{4k}{h}I \end{bmatrix}.$$

5. (2, 2), (2, 3)

$$(-1)^k \begin{bmatrix} -I & 0 \\ -\frac{h}{2}(A_4 + \frac{4k}{h}I)W + Q_2^{-1}Q_1 & -I \end{bmatrix}.$$

6. (2, 4)

$$(-1)^k \begin{bmatrix} -A_2(Y - Q_4 Q_2^{-1}) \\ (2A_4 + \frac{8(k-1)}{h}I)W - \frac{4}{h}(Q_4 Q_2^{-1} + Q_2^{-1}Q_1) \end{bmatrix}.$$

7. $(3,1)$

$$(-1)^k \begin{bmatrix} A_2 Q_4 Q_2^{-1} & -A_2 \\ (A_4 + 4\frac{k-1}{h})Q_4 Q_2^{-1} + \frac{2}{h}Q_2^{-1} & -A_4 - 4\frac{k-1}{h} \end{bmatrix}.$$

8. $(3,2)$, $(3,3)$

$$(-1)^k \begin{bmatrix} I & 0 \\ \frac{2}{h}(A_4 + \frac{4(k-1)}{h}I)W + Q_2^{-1}Q_1 & -I \end{bmatrix}.$$

9. $(3,4)$

$$(-1)^k \begin{bmatrix} -A_2(Y - Q_4 Q_2^{-1}) \\ -2(A_4 + \frac{4}{h}I)W - \frac{4}{h}Q_4 Q_2^{-1} - \frac{4}{h}Q_2^{-1}Q_1 \end{bmatrix}.$$

10. $(4,1)$

$$(-1)^k \ \frac{2}{h}[-2kQ_4 Q_2^{-1} - Q_2^{-1} \qquad 2kI].$$

11. $(4,2)$, $(4,3)$

$$(-1)^k[-2kW - Q_2^{-1}Q_1 \qquad -I].$$

12. $(4,4)$

$$(-1)^k \frac{2}{h}[-2(k-1)Y + 2Q_2^{-1}Q_1 + 2kQ_4 Q_2^{-1}].$$

Finally, computing the first and last entries of $(G^{-1}H)^k G^{-1}\gamma$ gives Theorem 3.1. □

4. Numerical experiments. The preceding analysis shows that the feasible solutions examined by SOCS can differ substantially from the IRK solutions. Once h is sufficiently small, approximations of the true optimal solution become feasible. In order to determine if this is the full story we conduct some computational tests on coarser meshes.

Recall the optimal control problem given previously in (4). The constraint is active on the subinterval $[\frac{34}{15}, 4]$. The numerical solution for x, v, and u obtained from SOCS TR and IRK TR were subtracted with $N = 10$. The results are given in Table 1. Recall that the tolerances δ_k and ϵ_k are those for SOCS to solve (11a), (11b).

SOCS solves (11a) up to the square root of machine precision. In addition, when the constraint is active, (11b) is also solved to the square root of machine precision. Theorem 3.1 gives absolute error bounds. After substituting

$$\epsilon_k = \delta_k = \sqrt{\varepsilon_{mach}}$$

into the formulas of Theorem 3.1, we observe that the difference between the two solutions are much larger than the estimates allow. For problem (4) on a mesh of size $N = 10$, Theorem 3.1 predicted a difference on the order of 10^{-5}. As seen in Figures 1 and 2 and Table 1, the two solutions differ by much more than that. The NLP tolerances are not enough to explain what we are seeing computationally on coarse meshes.

Table 1
Absolute difference of solutions.

Time	$\lvert\widetilde{x}-x\rvert$	$\lvert\widetilde{v}-v\rvert$	$\lvert\widetilde{u}-u\rvert$
2.2667	0.0000	0.0000	0.0000
2.4593	0.0070	0.0726	0.7540
2.6519	0.0000	0.1452	3.0159
2.8444	0.0000	0.1452	6.0318
3.0370	0.0000	0.1452	9.0477
3.2296	0.0000	0.1452	12.0637
3.4222	0.0000	0.1452	15.0796
3.6148	0.0000	0.1452	18.0955
3.8074	0.0000	0.1452	21.1114
4.0000	0	0.1452	24.1273

Table 2
Predicted and absolute differences of solutions.

$\lvert\widetilde{v}-v\rvert$		$\lvert\widetilde{u}-u\rvert$	
Actual	Pred.	Actual	Pred.
0.0000	0	0.0000	0
0.0726	0.0726	0.7540	0.7544
0.1452	0.1453	3.0159	3.0177
0.1452	0.1453	6.0318	6.0355
0.1452	0.1453	9.0477	9.0533
0.1452	0.1453	12.0637	12.0710
0.1452	0.1453	15.0796	15.0888
0.1452	0.1453	18.0955	18.1065
0.1452	0.1453	21.1114	21.1243
0.1452	0.1453	24.1273	24.1420

We now turn to identifying what is happening on coarser meshes. Upon closer examination of the solution obtained from SOCS, we see that not all variables $\widetilde{x}_k$ lie directly on the constraint. In fact, $\widetilde{x}_1$ is consistently above the constraint for meshes larger than $N = 5$, as illustrated by the second line of the $|\widetilde{x} - x|$ column in Table 1.

We again set the tolerances

$$\delta_k = \begin{bmatrix} \sqrt{\varepsilon_{mach}} \\ \sqrt{\varepsilon_{mach}} \end{bmatrix} \tag{30}$$

since they are the defaults in SOCS. However, in light of the fact that $\widetilde{x}_1$ does not lie on the constraint, we set ϵ_1 equal to the distance $\widetilde{x}_1$ is from the true solution. In the case when $N = 10$ we have

$$\epsilon_1 = 0.007, \quad \epsilon_k = \sqrt{\varepsilon_{mach}}, \quad k \geq 2. \tag{31}$$

The results of applying Theorem 3.1 using these new tolerances and $N = 10$ are given in Table 2.

The small error created by the constraint chatter in the early stages of SOCS TR propagates down the numerical solution as seen in (22) to create large differences between IRK and SOCS later in time. In addition, as h gets smaller the distance from the constraint also gets smaller.

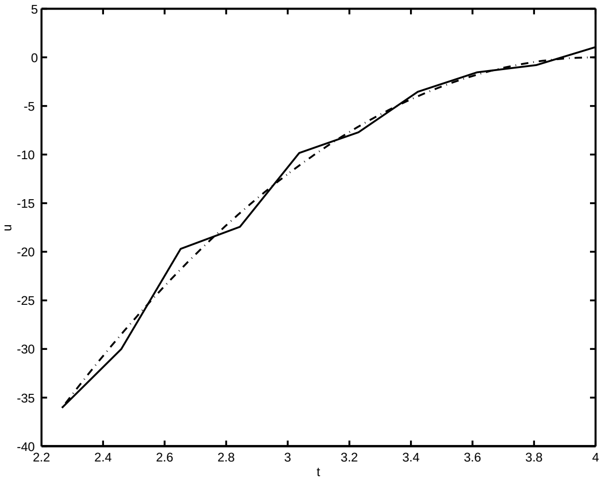


Fig. 3. *TR on the DAE with perturbation.*

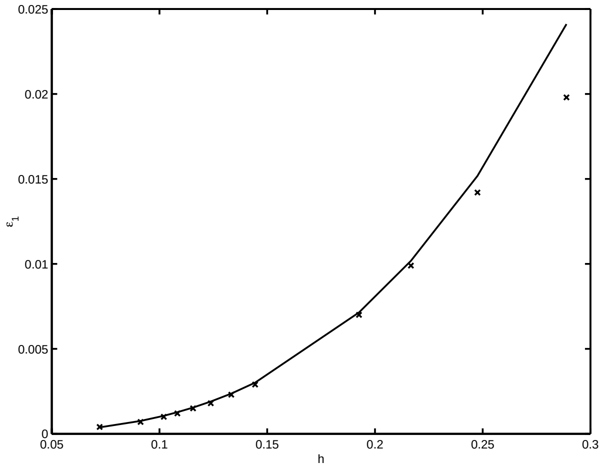


Fig. 4. *Perturbation size versus step size.*

Table 2 seems to suggest that for this test problem the perturbation (31) accounts for the ability of SOCS to find the optimum. As an additional check, rather than simply integrating the DAE (7) using IRK TR, we force the TR equations to include the addition of ϵ_1 from (31) at the first time step. In other words, at t_1 we have

$$0 = Cx_1 - g(t_1) - \epsilon_1$$

as part of the IRK equations given in (10a)–(10d). This results in a much more accurate approximation of the control as seen in Figure 3. The perturbed IRK TR solution of Figure 3 is also now very close to the SOCS TR solution of Figure 1.

The initial constraint chatter shown in Table 1 is present at a wide range of mesh sizes. Figure 4 gives a plot of the size of the constraint chatter (perturbation) versus the step size.

It appears that the constraint chatter is following the curve h^3. It is interesting

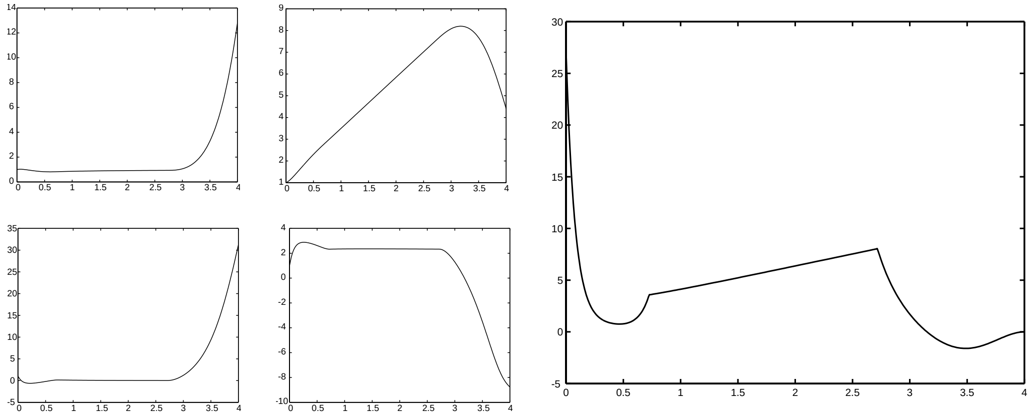


FIG. 5. *State and control solutions of* (32).

to note that Theorem 3.1 says perturbations can be enlarged by $1/(h^3)$. Thus an h^3 perturbation is reasonable for inducing a difference between IRK and SOCS on the order of $O(1)$.

The constraint chatter seen in the test problem (4) is not unique to that problem. We have performed a number of numerical experiments on a variety of index 3 and index 4 problems, both linear and nonlinear. In each problem we saw chattering occurring on many levels. When solving an index 4 problem the constrained solution bounced on and off the curve with greater frequency than the index 3 case. This behavior also occurs with nonlinear problems. In addition, the chatter seems to occur at both the beginning and the end of the state constrained interval for boundary value problems.

For example, consider the nonlinear problem

$$\min \int_0^4 \left(x_1^2 + x_2^2 + u^2\right)\, dt, \tag{32a}$$

$$x_1' = x_3, \tag{32b}$$

$$x_2' = x_4, \tag{32c}$$

$$x_3' = x_2 x_1 - u + 1, \tag{32d}$$

$$x_4' = x_1 u - x_2, \tag{32e}$$

$$\frac{1}{2} \geq x_2 - x_2 x_1. \tag{32f}$$

State and control plots for (32) computed by SOCS are given in Figure 5.

A close-up examination of the constraint residual on the interval on which it is active shows that there is again chatter which goes to zero with the mesh size. However, now the chatter is asymmetrical and changes shape with the mesh. Figure 6 gives an enlarged view of the plot of $x_2 - x_2 x_1$ which is equal to $\frac{1}{2}$ when the constraint is active.

5. Convergence to optimal solution on index 3 problems. We now proceed to show that even though the discretization cannot integrate the constrained dynamics directly that the approximations of the optimal solution do converge.

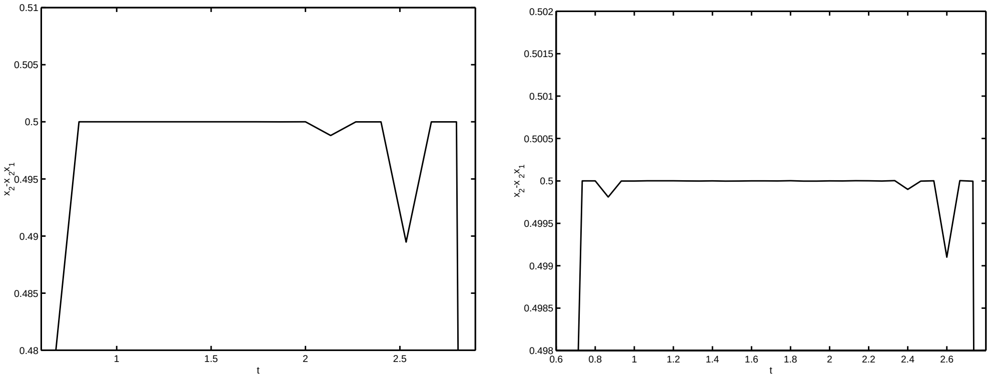


FIG. 6. *Chatter along constraint for* (32), $N = 31$ *and* $N = 61$.

We consider the optimal control problem (P)

$$\min \int_{t_0}^{t_f} L(x(t), u(t))\, dt, \tag{33a}$$

$$x'(t) = Ax(t) + Bu(t), \tag{33b}$$

$$0 \leq Cx(t) - g(t), \tag{33c}$$

$$x(t_0) = x_0, \tag{33d}$$

where $x(t)$ is a real vector-valued function of length m and $u(t)$ is length r. Also, A is $m \times m$, B is $m \times r$, C is $q \times m$, and $g(t)$ is a vector-valued function of length q and is continuous. It is assumed that the functional $L : \Re^{m+r} \rightarrow \Re$ is strictly convex. Note that we now make no restrictions on A, B, C concerning the index when the constraints are active. This is consistent with our numerical observations showing convergence using TR on active inequality constraints of index 4, 5, and higher.

Many papers have discussed approximating a state constrained continuous infinite-dimensional optimal control problem with a discrete one. One of the first discussions [8] gave an algorithm to solve general problems which include (33). In that paper, the state constraint needed to be differentiated until the control appeared. This relates closely to the *order* of a constraint or the index of a DAE. Loosely stated, the index of a DAE is the number of differentiations of the system equations needed to determine the derivative of the algebraic variables uniquely. For more on order and DAEs, see [7] and [14].

Convergence rates when approximating the infinite problem by a finite-dimensional NLP were developed in [9, 11, 12]. [9] gives a convergence rate for the Euler integration scheme using both state and control constraints. [11] uses the Euler integration scheme and develops convergence rates for problems with constraints on the states only. Using Runge–Kutta methods, [12] considers control constrained problems. Each of these papers assumes that the state constraints satisfy a stability requirement. In terms of (33), this implies that the matrix CB must be nonsingular. Thus the DAE that arises when the constraints are active is only index 2. Furthermore, each of the papers listed, and other related work such as [13], assumes or proves under some assumptions that all of the associated multipliers converge.

In [16, 17, 20], Polak developed the theory of consistent approximations. A sequence of finite NLPs (involving only the discretized control variables) consistently approximate an optimal control problem if the sequence of minimizers of the finite problems converge to the minimizers of the original problem. Convergence of the minimizers is related to convergence of the epigraphs of the associated problems. In [16], the Euler integration scheme is used to show that general control constrained problems can be consistently approximated by a sequence of NLPs. Explicit Runge–Kutta methods can also be used for control constrained problems [20]. Finally, it has been shown [17] that, using the Euler integration scheme, optimal control problems with general state constraints can be consistently approximated.

Pytlak discusses algorithms that solve state constrained optimization problems where the dynamics may be replaced by an index 1 DAE [18, 19]. The algorithm optimizes only over the discretized control variables. This allows the use of a reduced gradient algorithm based on the discretized control and adjoint variables [18]. Although this algorithm seems to work quite well, the general strategy is very different from that of SOCS.

In each of the above papers convergence depends on convergence of the multipliers. Figure 7 shows two of the discrete multipliers for (4), indicated by ×'s, and the continuous multipliers, which are the smooth curves. The discrete multipliers are not converging to any function. Thus the proof that SOCS can use TR with high index inequality constraints cannot be based on the convergence of multipliers. There is some indication that the multipliers may converge in a weak sense.

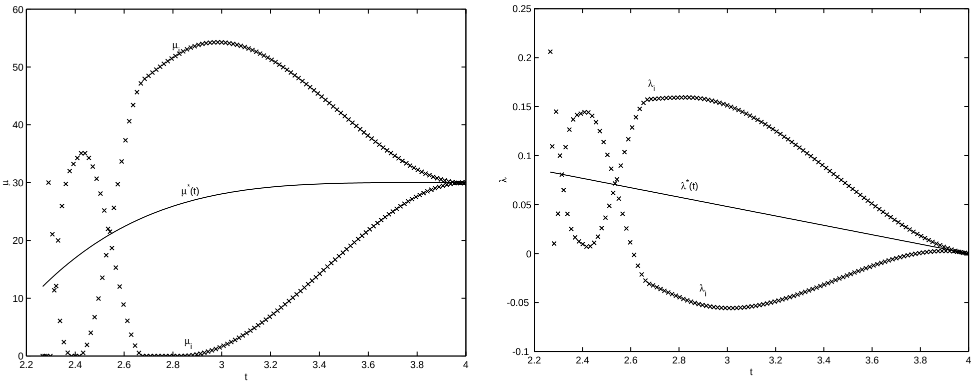


FIG. 7. *Multipliers for* (4).

In summary, our situation is different from the papers listed above in that (i) the dynamics are discretized using the TR or HS methods, (ii) the discretized controls and states are included as NLP variables, (iii) the discretized dynamics are included as constraints of the NLP, (iv) experimental results show that the associated multipliers of the equality and inequality constraints may not converge, (v) experiments show apparent convergence for problems with purely state constrained problems where CB is singular, and (vi) as seen in [6], the TR and HS methods may not converge when applied to the DAEs which arise when constraints are active.

5.1. Consistent approximations. A detailed discussion of consistent approximations is given in [17]. We review only needed notation.

Consider the two problems

$$(P) \qquad \min_{x \in X} f(x)$$

and

$$(P_N) \qquad \min_{x \in X_N} f_N(x).$$

The epigraphs of the functions f and f_N are defined as $E = \{(x^0, x) \mid x \in X,\ x^0 \geq f(x)\}$ and $E_N = \{(x^0, x) \mid x \in X_N,\ x^0 \geq f_N(x)\}$, respectively. The convergence of a sequence of NLPs to an optimal control problem is given in terms of the convergence of the associated epigraphs. Theorems 5.1 and 5.2 are the two main theorems of consistent approximations [17]. In the following let $\mathcal{N}$ be any infinite subset of the natural numbers.

THEOREM 5.1. *The epigraphs E_N, $N \in \mathcal{N}$, of the problems P_N converge to the epigraph E of the problem P if and only if*

1. *for every $x \in X$, there exists a sequence $\{x_N\}_{N \in \mathcal{N}}$ with $x_N \in X_N$ such that $x_N \to x$ as $N \to \infty$;*
2. *for every infinite sequence $\{x_N\}_{N \in K}$, $K \subset \mathcal{N}$ such that $x_N \in X_N$ and $x_N \to x$ as $N \to \infty$, then $x \in X$;*
3. $\lim f_N(x_N) = f(x)$ *if $x_N \to x$, $x_N, \in X_N$, and $x \in X$.*

THEOREM 5.2. *If the epigraphs of P_N converge to the epigraph of P, then if $\{\hat{x}_N\}_{N \in \mathcal{N}}$ is a sequence of global minimizers of the problems P_N and $\hat{x}$ is any accumulation point of this sequence, then $\hat{x}$ is a global minimizer of P.*

If we can show items 1, 2, and 3 in Theorem 5.1, then we have shown that if the global minimizers of the approximating problem converge, it must be to a minimum of the original problem. It is important to note that for some problems, a stationary point of the approximating problems P_N may converge to a nonstationary point of the problem P. However, this is not the case when the approximating problems P_N and the problem P have convex objective functions with convex constraint sets. The convergence takes place in a Banach space. But X, X_N need not be closed. Getting the correct X, X_N is a key technical point.

5.2. Problem statement.

Consider again the optimal control problem (P) in (33). Let the mesh T_N be $t_0 < t_1 < \cdots < t_n = t_f$ where $N = n+1$. Let $h_i = t_{i+1} - t_i$. The associated NLP from discretization using the trapezoid rule is

$$\min \ \frac{h_0}{2} L(x_0, u_0) + \frac{h_{n-1}}{2} L(x_n, u_n) + \sum_{i=1}^{n-1} \frac{h_{i-1} + h_i}{2} L(x_i, u_i),$$

$$\begin{aligned} 0 &= x_{i+1} - x_i - \frac{h_i}{2}(A(x_i + x_{i+1}) + B(u_i + u_{i+1})), \\ 0 &\leq Cx_i - g(t_i), \end{aligned}$$

where $i = 0, 1, \ldots, n-1$. Let

$$x_N = [x_0,\ x_1,\ \ldots,\ x_n]^T, \qquad u_N = [u_0,\ u_1,\ \ldots,\ u_n]^T$$

be the matrices which approximate $x(t)$ and $u(t)$ at the mesh points. Let the ordered set of mesh points be denoted T_N and the norm of the mesh be

$$\|T_N\| \doteq \max_i \ h_i = \max_i \ (t_{i+1} - t_i).$$

Note that $x_i \in R^m$, $u_i \in R^r$. For ease of notation, if $z(t)$ is a vector-valued function over the interval $[t_0, t_f]$, let $z(T_N)$ be the matrix of values of $z(t)$.

The linear interpolation of x_i and u_i are

$$\bar{u}_N(t) = \sum_{i=0}^{n} u_i \phi_i(t), \quad \bar{x}_N(t) = \sum_{i=0}^{n} x_i \phi_i(t),$$

where the $\phi_i(t)$ are the linear hat functions with the property

$$\phi_i(t_j) = \begin{cases} 1 & \text{if} \quad i = j, \\ 0, & i \neq j. \end{cases}$$

Given $(x_N, u_N) \in \Re^{(m \times N)} \times \Re^{(r \times N)}$, the pair $(\bar{x}_N(t), \bar{u}_N(t))$ is in the finite-dimensional space $\Phi \times \Phi$ where $\Phi = \text{span}\{\phi_0, \phi_1, \ldots, \phi_n\}$.

If $z(t)$ is a vector-valued function of length l and $s \in [t_0, t_f]$, then the norm of the vector $z(s)$ is $\|z(s)\|_\infty = \max_i |z_i(s_1)|, 1 \leq i \leq l$. The function space norm is given by $\|z(t)\| = \sup_t \|z(t)\|_\infty$. Finally, if A is a matrix, then $\|A\|_\infty$ is the norm induced by $\| \cdot \|_\infty$.

Figure 5 is typical in that the optimal controls with higher index inequality constraints are rarely smooth. But they are often continuous and piecewise smooth. Continuity is also not enough for the analysis to follow. Accordingly, we take the set of admissible controls as the set of functions which are Lipschitz continuous. That is, for each u, there exists some $L < \infty$ such that

$$\|u(s_1) - u(s_2)\|_\infty \leq L|s_1 - s_2| \quad \text{for all } s_1, s_2. \tag{34}$$

If u is Lipschitz continuous, let

$$\|u(t)\|_L = \sup_{s_1, s_2 \in [t_0, t_f]} \frac{\|u(s_1) - u(s_2)\|_\infty}{|s_1 - s_2|}, \quad s_1 \neq s_2.$$

While we need $\|u\|_L < \infty$, computational examples show that the approximations do not converge in $\|u\|_L$. Thus the underlying Banach space has to have a weaker type of convergence which is compensated for by the definition of the finite-dimensional approximating spaces. The basic underlying Banach space is

$$\mathcal{B} = \{(x, u) \mid x \in C^0[t_0, t_f], \ u \in C^0[t_0, t_f]\}$$

with

$$\|(x, u)\|_{\mathcal{B}} = \max\{\|x\|, \|u\|\}.$$

The finite-dimensional subspace $\mathcal{B}_N \subset \mathcal{B}$ is defined to be

$$\mathcal{B}_N = \{(x, u) \mid x \in \Phi, \ u \in \Phi\}.$$

We now define problems P and P_N. Let P be

$$\min_{(x,u) \in X} f(x, u),$$

$$f(x, u) = \int_{t_0}^{t_f} L(x, u) \, dt,$$

where $f : \mathcal{B} \to \Re$ and $X \subset \mathcal{B}$ is

$$X = \{(x, u) \mid x' = Ax + Bu,\ Cx - g(t) \geq 0,\ x(t_0) = x_0,\ \|u\|_L \leq \Lambda\}.$$

Here Λ is a fixed finite positive number. The reader should think of Λ as being large but fixed. In particular, it is larger than $\|u^*\|_L$ for the optimal control u^*, which is also assumed Lipschitz. It is assumed that the set X is nonempty.

Let T_N be any ordered mesh. Define P_N to be

$$\min_{(\bar{x}_N, \bar{u}_N) \in X_N} f_N(\bar{x}_N, \bar{u}_N),$$

$$\begin{aligned} f_N(\bar{x}_N, \bar{u}_N) = {} & \frac{h_0}{2} L(\bar{x}_N(t_0), \bar{u}_N(t_0)) + \frac{h_{n-1}}{2} L(\bar{x}_N(t_n), \bar{u}_N(t_n)) \\ & + \sum_{i=1}^{n-1} \frac{h_i + h_{i-1}}{2} L(\bar{x}_N(t_i), \bar{u}_N(t_i)), \end{aligned}$$

where $f_N : \mathcal{B}_N \to \Re$ and $X_N \subset \mathcal{B}_N$ is the set of $(\bar{x}_N, \bar{u}_N)$ such that

$$\begin{aligned} \bar{x}_N(t_{i+1}) - \bar{x}_N(t_i) - \frac{h_i}{2}(A(\bar{x}_N(t_{i+1}) + \bar{x}_N(t_i)) + B(\bar{u}_N(t_{i+1}) + \bar{u}_N(t_i))) = 0, \\ C\bar{x}_N(t_{i+1}) - g(t_{i+1}) \geq 0, \quad \bar{x}_N(t_0) = x_0, \\ \|\bar{u}_N(t_{i+1}) - \bar{u}_N(t_i)\|_\infty \leq \Lambda |t_{i+1} - t_i|, \qquad 0 \leq i \leq n-1. \end{aligned}$$

Here $t_i \in T_N$ and Λ is the same as in the definition of X. The constraint $\|\bar{u}_N(t_{i+1}) - \bar{u}_N(t_i)\|_\infty \leq \Lambda |t_{i+1} - t_i|$ is not present in SOCS and is included only for theoretical reasons.

Since all affine, nonempty, linearly constrained sets are convex we have that the sets X and X_N defined above are convex. Convexity plays an important role. Since $L(x, u)$ is strictly convex and the constraint sets are convex, there exist unique global minimizers for the problems P and P_N. Thus, if the problems P_N approximate the problems P, by Theorem 5.1 the limit of global minimizers of the P_N is the global minimizer of P.

The usual results in the literature on using trapezoidal discretizations assume twice differentiable solutions, which are not present in our application. The control in Figure 5 is typical for inequality constrained problems in that it is only piecewise smooth. Accordingly we must first examine the convergence of the TR method for linear ODEs of the form $x' = Ax + f(t)$ where $f(t)$ is only Lipschitz continuous. Let $z_N^I(t)$ be the linear interpolation of the function $z(t)$ over the mesh T_N.

LEMMA 5.3. *If v is Lipschitz continuous with Lipschitz constant $\|v\|_L$, then its linear interpolation v_N^I on a mesh T_N satisfies $\|v_N^I(t) - v(t)\| \leq \|v\|_L \|T_N\|$.*

Proof. Take $s \in [t_0, t_f]$ and let $\hat{t}$ be the closest mesh point to s, $\hat{t} = \min_{t_i} |t_i - s|$. Then

$$\begin{aligned} \|v_N^I(s) - v(s)\|_\infty &= \|v_N^I(s) - v(\hat{t}) + v(\hat{t}) - v(s)\|_\infty \\ &\leq \|v_N^I(s) - v(\hat{t})\|_\infty + \|v(\hat{t}) - v(s)\|_\infty \\ &= \|v_N^I(s) - v_N^I(\hat{t})\|_\infty + \|v(\hat{t}) - v(s)\|_\infty \\ &\leq \|v_N^I(t)\|_L\ |s - \hat{t}| + \|v\|_L |\hat{t} - s| \leq 2\|v\|_L |\hat{t} - s| \leq \|v\|_L \|T_N\|. \qquad \square \end{aligned}$$

LEMMA 5.4. *If $z(t)$ is a Lipschitz continuous function, then its trapezoid quadrature is locally order* 2. *That is, if $t_i, t_{i+1} \in T_N$, then*

$$\left\| \int_{t_i}^{t_{i+1}} z(s)\, ds - \frac{h_i}{2}(z(t_i) + z(t_{i+1})) \right\|_\infty < \|z\|_L \|T_N\|^2.$$

Proof. The trapezoid rule is exact for the linear interpolants so that

$$\int_{t_i}^{t_{i+1}} z_N^I(s)\, ds = \frac{h_i}{2}(z_N^I(t_i) + z_N^I(t_{i+1})) = \frac{h_i}{2}(z(t_i) + z(t_{i+1})).$$

Define $E = \int_{t_i}^{t_{i+1}} z(s)\, ds - \frac{h_i}{2}(z(t_i) + z(t_{i+1}))$. Then

$$\begin{aligned} E &= \int_{t_i}^{t_{i+1}} (z(s) - z_N^I(s))\, ds + \int_{t_i}^{t_{i+1}} z_N^I(s)\, ds - \frac{h_i}{2}(z(t_i) + z(t_{i+1})) \\ &= \int_{t_i}^{t_{i+1}} (z(s) - z_N^I(s))\, ds. \end{aligned}$$

Taking norms and using Lemma 5.3 we have

$$\|E\|_\infty \le \int_{t_i}^{t_{i+1}} \|z(s) - z_N^I(s)\|\, ds \le \int_{t_i}^{t_{i+1}} \|z\|_L \|T_N\|\, ds \le \|z\|_L \|T_N\|^2. \qquad \square$$

Before we prove the next lemma, we must assume that the mesh norm multiplied by the size of the mesh is bounded. That is, for all $N \in \mathcal{N}$, there exists $C_6 > 0$ such that

$$\|T_N\| N \le C_6. \tag{35}$$

Moreover, for any positive integer k we have $T_N \subset T_{N+k}$. That is, when we have a sequence of meshes $\{T_N\}$ we assume that T_{N+k} is gotten by refining T_N. This is consistent with the strategy in SOCS. Assumptions such as (35) are common when nonuniform meshes are used.

LEMMA 5.5. *Let $0 < \delta < 1$. If we apply the TR method to the initial value problem*

$$x'(t) = Ax(t) + f(t), \quad x(t_0) = x_0, \quad t \in [t_0, t_f] \tag{36}$$

with $f(t)$ Lipschitz continuous, then for $\|T_N\|$ small enough so that $\frac{\|T_N\|}{2}\|A\|_\infty \le 1 - \delta$, the error $e_j = x_j - x(t_j)$ satisfies

$$\|e_j\|_\infty \le K_1 \|T_N\| \tag{37}$$

for all j where x_j is the solution of the trapezoid difference equation applied to (36).

Proof. The TR method applied to the ODE $x' = Ax + f(t)$ is

$$x_{i+1} - x_i = \frac{h_i}{2}(Ax_{i+1} + Ax_i + f(t_{i+1}) + f(t_i)).$$

The true solution discretized is

$$x(t_{i+1}) - x(t_i) = \frac{h_i}{2}(Ax(t_{i+1}) + Ax(t_i) + f(t_{i+1}) + f(t_i)) + \gamma_i.$$

Here γ_i is the local truncation error of the trapezoid method. Define $e_i = x(t_i) - x_i$ to obtain

$$e_{i+1} - e_i = \frac{h_i}{2}(Ae_{i+1} + Ae_i) + \gamma_i, \quad e_0 = 0. \tag{38}$$

Equation (38) is equivalent to

$$e_{i+1} = \left(I - \frac{h_i}{2}A\right)^{-1}\left(I + \frac{h_i}{2}A\right)e_i + \left(I - \frac{h_i}{2}A\right)^{-1}\gamma_i.$$

Taking norms gives

$$\|e_{i+1}\|_\infty \leq \left\|\left(I - \frac{h_i}{2}A\right)^{-1}\right\|_\infty \|e_i\|_\infty + \left\|\left(I - \frac{h_i}{2}A\right)^{-1}\frac{h_i}{2}A\right\|_\infty \|e_i\|_\infty + \left\|\left(I - \frac{h_i}{2}A\right)^{-1}\right\|_\infty \|\gamma_i\|_\infty.$$

Since $\frac{\|T_N\|}{2}\|A\|_\infty < 1$ we have that $\|(I - \frac{h_i}{2}A)^{-1}\|_\infty \leq C_1$ for some constant C_1. We may also write

$$\left\|\left(I - \frac{h_i}{2}A\right)^{-1}\right\|_\infty \leq \frac{\|T_N\|}{2}\|A\|_\infty\left(1 + \sum_{k=2}^{\infty}\left(\frac{\|T_N\|}{2}\|A\|_\infty\right)^k\right) + 1.$$

There exists a C_2 such that

$$1 + \sum_{k=2}^{\infty}\left(\frac{\|T_N\|}{2}\|A\|_\infty\right)^k \leq C_2.$$

Finally, let $C_5 = \frac{\|A\|_\infty(C_1+C_2)}{2}$ to get

$$\|e_{i+1}\|_\infty \leq (1 + C_5\|T_N\|)\, \|e_i\|_\infty + C_1\, \|\gamma_i\|_\infty.$$

Recalling that $e_0 = 0$, we have that at the nth step

$$\begin{aligned}\|e_n\|_\infty &\leq C_1\sum_{i=0}^{n-1}(1 + C_5\|T_N\|)^{n-i-1}\|\gamma_i\|_\infty \\ &\leq C_1 \max_i \|\gamma_i\|_\infty \sum_{i=0}^{N-1}(1 + C_5\|T_N\|)^{n-i-1}.\end{aligned}$$

But from the formula for a finite geometric series and (35) we have

$$\begin{aligned}\sum_{i=0}^{N-1}(1 + C_5\|T_N\|)^i &= \frac{(1 + C_5\|T_N\|)^N - 1}{C_5\|T_N\|} \\ &\leq \frac{(1 + \frac{C_7}{N})^N - 1}{C_5\|T_N\|} < \frac{e^{C_7} - 1}{C_5\|T_N\|} = \frac{C_8}{\|T_N\|},\end{aligned}$$

where $C_7 = C_5C_6$ and $C_8 = \frac{e^{C_7}-1}{C_5}$. Therefore

$$\|e_n\|_\infty \leq \frac{C_1C_8}{\|T_N\|}\max_i \|\gamma_i\|_\infty. \tag{39}$$

But

$$\begin{aligned}\|\gamma_i\|_\infty &= \left\| x(t_{i+1}) - x(t_i) - \frac{h_i}{2}(Ax(t_{i+1}) + x(t_i) + f(t_{i+1}) + f(t_i)) \right\|_\infty \\ &= \left\| x(t_{i+1}) - x(t_i) - \frac{h_i}{2}(x'(t_{i+1}) + x'(t_i)) \right\|_\infty \\ &= \left\| \int_{t_i}^{t_{i+1}} x'(t)\, dt - \frac{h_i}{2}(x'(t_{i+1}) + x'(t_i)) \right\|_\infty ,\end{aligned}$$

which is simply the trapezoid quadrature on the function $x'(t)$. Since $x'(t)$ is Lipschitz, Lemma 5.4 gives $\|\gamma_i\|_\infty < K_4\|T_N\|^2$ and (37) follows from (39). □

Lemma 5.5 states that for every mesh point t_i, $\|\bar{x}_N(t_i) - x(t_i)\|_\infty \le K_1\|T_N\|$ if the function $\bar{x}_N(t)$ satisfies the trapezoid equations. We now show this holds for all t. Recall that $u(t)$ is Lipschitz continuous; hence $x(t)$ and $x'(t)$ are also Lipschitz, since $x' = Ax + Bu$.

LEMMA 5.6. *If $\bar{x}_N(t)$ satisfies the assumptions of Lemma 5.5, then $\|x(t) - \bar{x}_N(t)\| \le K_2\|T_N\|$.*

Proof. Since $\bar{x}_N(t)$ satisfies the trapezoid equations, by Lemma 5.5 we have $\|\bar{x}_N(t_i) - x(t_i)\|_\infty \le K_1\|T_N\|$. In addition, $x(t)$ is Lipschitz continuous, and hence its linear interpolation is close, $\|x_N^I(t) - x(t)\| \le K_4\|T_N\|$. At each mesh point $x_N^I(t_i) = x(t_i)$ and so $\|\bar{x}_N(t_i) - x_N^I(t_i)\|_\infty \le K_1\|T_N\|$. Since linear functions are furthest from each other at the boundary, we have $\|\bar{x}_N(t) - x_N^I(t)\| \le K_1\|T_N\|$ and thus

$$\begin{aligned}\|\bar{x}_N(t) - x(t)\| &= \|\bar{x}_N(t) - x_N^I(t) + x_N^I(t) - x(t)\| \\ &\le \|\bar{x}_N(t) - x_N^I(t)\| + \|x_N^I(t) - x(t)\| \\ &\le K_1\|T_N\| + K_4\|T_N\| \le K_2\|T_N\|. \qquad \square\end{aligned}$$

Finally, we make the following assumption on the constraint sets.

ASSUMPTION 5.1. *Given problem P defined above suppose that for every $(x, u) \in X$, there exists a sequence of functions $\{(y_M(t), v_M(t))\} \in X$ such that*

$$(40) \qquad Cy_M(t) - g(t) > 0, \qquad y_M(t_0) = x_0, \qquad y_M'(t) = Ay_M(t) + Bv_M(t)$$

for all integers M and $(y_M(t), v_M(t)) \to^{\mathcal{B}} (x(t), u(t))$ as $M \to \infty$.

The above assumption is on the set $X \in \mathcal{B}$ and has no relation to the mesh. It says that all feasible solutions are the limit of feasible solutions that satisfy the inequality constraints with a strict inequality. This does not appear to be very restrictive.

5.3. Satisfying the assumptions of Theorem 5.1.

We begin by showing item 1 in Theorem 5.1.

LEMMA 5.7. *Let assumption (40) hold and let $(x, u) \in X$. Then there exists a sequence $\{\bar{x}_N, \bar{u}_N\}_{N\in\mathcal{N}}$ so that for each N, $(\bar{x}_N, \bar{u}_N) \in X_N$ and $(\bar{x}_N, \bar{u}_N) \to^{\mathcal{B}} (x, u)$.*

Proof. Let $(x, u) \in X$. By (40) we have that there is a sequence of continuous functions $\{y_M, v_M\} \in X$ such that $(y_M, v_M) \to (x, u)$ and $Cy_M - g > 0$ for all M. For each M there exists a $\rho(M) > 0$, so that if a function y lies within the ball of radius $\rho(M)$ around y_M, $y \in B_{\rho(M)}(y_M)$, we have that $Cy - g > 0$.

Let $\epsilon > 0$. Define a sequence $\bar{x}_N, \bar{u}_N \in X_N$ with $N \ge M$ as $\bar{u}_N(t_i) = v_M(t_i)$ and $\bar{x}_N(T_N)$ as the unique matrix in $\Re^{m\times N}$ that satisfies the trapezoid equations associated with $\bar{u}_N$ and $\bar{x}_N(t_0) = x_0$. Choose M large enough so that

$$\|y_M(t) - x(t)\| < \frac{\epsilon}{2}, \quad \|v_M(t) - u(t)\| < \frac{\epsilon}{2}.$$

Since $\bar{u}_N(t_i) = v_M(t_i)$ and $\bar{x}_N(T_N)$ is the unique matrix which solves the trapezoid equations, we choose N large enough so that Lemma 5.6 holds with $\|T_N\| < \epsilon/2K_2$ and $\|\bar{x}_N(t) - y_M(t)\| < \min\left\{\rho(M), \frac{\epsilon}{2}\right\}$ and $\|\bar{u}_N(t) - v_M(t)\| < \frac{\epsilon}{2}$. Thus

$$\begin{aligned}\|\bar{x}_N(t) - x(t)\| &= \|\bar{x}_N(t) - y_M(t) + y_M(t) - x(t)\| \\ &\leq \|\bar{x}_N(t) - y_M(t)\| + \|y_M(t) - x(t)\| \leq \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon\end{aligned}$$

and

$$\begin{aligned}\|\bar{u}_N(t) - u(t)\| &= \|\bar{u}_N(t) - v_M(t) + v_M(t) - u(t)\| \\ &\leq \|\bar{u}_N(t) - v_M(t)\| + \|v_M(t) - u(t)\| \leq \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon.\end{aligned}$$

Finally, note that $\|u_N\|_L < \Lambda$ since it is the linear interpolation of v_M with $\|v_M\|_L < \Lambda$. □

We now show item 2 in Theorem 5.1. Let $\{(\bar{x}_N, \bar{u}_N)\}_{N\in\mathcal{N}} \subset X_N$ be such that $(\bar{x}_N, \bar{u}_N) \to^{\mathcal{B}} (x, u)$. We must show that $(x, u) \in X$. We begin by examining convergence when $\bar{x}_N$ is unconstrained.

Lemma 5.8. *If $(\bar{x}_N, \bar{u}_N) \in \mathcal{B}_N$ with $(\bar{x}_N, \bar{u}_N) \to^{\mathcal{B}} (x, u)$ and*

$$\|u_N(t_{i+1}) - u(t_i)\|_\infty \leq \Lambda|t_{i+1} - t_i| \quad \text{for all } t_i \in T_N, \tag{41}$$

then u is Lipschitz continuous with $\|u\|_L \leq \Lambda$.

Proof. Suppose this is not the case. Then there exists an $s_1, s_2 \in [t_0, t_f]$ such that $\|u(s_1) - u(s_2)\|_\infty > \hat{\Lambda}|s_1 - s_2|$ where $\hat{\Lambda} > \Lambda$. Let $\epsilon > 0$ be such that $\Lambda + \epsilon < \hat{\Lambda}$. Then

$$\begin{aligned}\|u(s_1) - u(s_2)\|_\infty &= \|u(s_1) - \bar{u}_N(s_1) + \bar{u}_N(s_1) - u(s_2)\|_\infty \\ &\leq \|u(s_1) - \bar{u}_N(s_1)\|_\infty + \|\bar{u}_N(s_1) - \bar{u}_N(s_2)\|_\infty + \|\bar{u}_N(s_2) - u(s_2)\|_\infty.\end{aligned}$$

Note that if (41) holds for u_N, then $\|u_N\|_L \leq \Lambda$. Choose N large enough so that $\|\bar{u}_N - u\| < \frac{\epsilon}{2}|s_1 - s_2|$. Then

$$\|u(s_1) - u(s_2)\|_\infty < \epsilon|s_1 - s_2| + \Lambda|s_1 - s_2| < \hat{\Lambda}|s_1 - s_2|,$$

which is a contradiction. Thus u is Lipschitz continuous with constant less than or equal to Λ. □

The following lemma was adapted from [15] and uses a standard Gronwall's lemma argument.

Lemma 5.9. *If x_1 and x_2 are solutions of $x_i' = Ax_i + Bu_i$, $x_i(t_0) = x_0$ where u_1 and u_2 are Lipschitz continuous functions, then $\|x_1 - x_2\| \leq \|B\|_\infty(t_f - t_0)\|u_1 - u_2\|e^{\|A\|_\infty(t_f - t_0)}$.*

We now restrict the space $\mathcal{B}_N$ to the subspace for which $(\bar{x}_N(t), \bar{u}_N(t))$ satisfies the trapezoid equations and $\bar{x}_N(t_0) = x_0$.

Lemma 5.10. *If $(\bar{x}_N, \bar{u}_N) \to^{\mathcal{B}} (x, u)$, $\bar{x}_N(t_0) = x_0$, and $(\bar{x}_N, \bar{u}_N)$ satisfies the trapezoid rule*

$$\bar{x}_N(t_{i+1}) - \bar{x}_N(t_i) - \frac{h_i}{2}(A(\bar{x}_N(t_{i+1}) + \bar{x}_N(t_i)) + B(\bar{u}_N(t_{i+1}) + \bar{u}_N(t_i))) = 0, \tag{42}$$

then the pair (x, u) satisfies $x'(t) = Ax + Bu$, $x(t_0) = x_0$.

Proof. We have that $(\bar{x}_N, \bar{u}_N)$ satisfies (42). Adding zero we have

$$\bar{x}_N(t_{i+1}) - \bar{x}_N(t_i) - \frac{h_i}{2}(A(\bar{x}_N(t_{i+1}) + \bar{x}_N(t_i)) + B(u(t_{i+1}) + u(t_i)) \\ +B(\bar{u}_N(t_{i+1}) - u(t_{i+1}) + \bar{u}_N(t_i) - u(t_i))) = 0.$$

Let M be any positive integer and replace $\bar{u}_N(t)$ above with $\bar{u}_M(t)$, which is fixed. Define $e_M(t) = \bar{u}_M(t) - u(t)$ to obtain

$$\bar{x}_N(t_{i+1}) - \bar{x}_N(t_i) - \frac{h_i}{2}(A(\bar{x}_N(t_{i+1}) + \bar{x}_N(t_i)) \\ + B(u(t_{i+1}) + u(t_i)) + B(e_M(t_{i+1}) + e_M(t_i))) = 0.$$

Since u and $\bar{u}_M$ are Lipschitz we have that e_M is also Lipschitz. If $\|T_N\|$ is small, then by Lemma 5.6, $\|\bar{x}_N(t) - y(t)\| \leq K_2\|T_N\|$ where y satisfies the differential equation $y' = Ay + Bu + Be_M,\ \ y(t_0) = x_0$. Let z be the solution of the differential equation $z' = Az + Bu,\ \ z(t_0) = x_0$. Lemma 5.9 implies that

$$\|y - z\| \leq \|B\|_\infty(t_f - t_0)\|e_M\|e^{\|A\|_\infty(t_f - t_0)}.$$

Now note that M was arbitrary. Since $\bar{u}_N(t) \to u(t)$ uniformly, let us choose $\hat{M}$ large enough so that

$$\|e_{\hat{M}}(t)\| < \frac{\epsilon}{2\|B\|_\infty(t_f - t_0)(e^{\|A\|_\infty(t_f - t_0)})},$$

which gives $\|y(t) - z(t)\| < \frac{\epsilon}{2}$. Finally choose $N \geq \hat{M}$ large enough so that $\|T_N\| \leq \epsilon/2K_2$ to get $\|\bar{x}_N(t) - y(t)\| < \frac{\epsilon}{2}$, and hence $\|\bar{x}_N(t) - z(t)\| < \epsilon$. Thus $\bar{x}_N(t)$ converges to both $x(t)$ and $z(t)$, which implies that $x(t) = z(t)$ and (x, u) satisfies the differential equation with initial conditions. □

Finally, we consider only the $(\bar{x}_N, \bar{u}_N)$ pairs which satisfy $C\bar{x}(t_i) \geq g(t_i)$, which is the set X_N defined earlier.

LEMMA 5.11. *If $(\bar{x}_N(t), \bar{u}_N(t)) \in X_N$ and converges to $(x(t), u(t))$ in $\mathcal{B}$, then $(x(t), u(t)) \in X$.*

Proof. First note that since $Cx(t) - g(t) \geq 0$ holds on the dense set $\cup_N T_N$, it holds for all t by continuity. By Lemmas 5.8 and 5.10, we have that (x, u) also satisfies $x' = Ax + Bu,\ x(t_0) = x_0$ with u Lipschitz continuous. Thus $(x, u) \in X \subset \mathcal{B}$. □

We now show item 3 in Theorem 5.1. Let

$$f(x, u) = \int_{t_0}^{t_f} L(x, u)\, dt,$$

and on a given mesh T_N,

$$f_N(\bar{x}_N, \bar{u}_N) = \frac{h_0}{2}L(\bar{x}_N(t_0), \bar{u}_N(t_0)) + \frac{h_{n-1}}{2}L(\bar{x}_N(t_n), \bar{u}_N(t_n)) \\ + \sum_{i=1}^{n-1} \frac{h_{i-1} + h_i}{2} L(\bar{x}_N(t_i), \bar{u}_N(t_i)).$$

ASSUMPTION 5.2. *$L(x, u)$ is a strictly convex functional and if $(x, u) \in \mathcal{B}$, then for all (y, v) such that*

$$\|(x, u) - (y, v)\|_{\mathcal{B}} \leq \Delta, \qquad \Delta > 0,$$

L has the property that

$$|L(x,u) - L(y,v)| \le K_3 \|(x,u) - (y,v)\|_{\mathcal{B}}.$$

In addition, if (x,u) is Lipschitz in t, then L is also Lipschitz in t.

Although Δ can be any finite positive integer, the reader should think of Δ as being a large number.

For ease of notation, we use w_i as the weights associated with the trapezoid rule.

Lemma 5.12. *Given $(x,u) \in \mathcal{B}$ and f and f_N defined above, then $|f(x,u) - f_N(x,u)| \le K_6\|T_N\|$*

Proof. Since x and u are fixed, the functions $f(x,u)$ and $f_N(x,u)$ can be considered functions of t and the mesh T_N. Hence,

$$|f(x,u) - f_N(x,u)| = \left| \int_{t_0}^{t_f} L(t)\, dt - \sum_{i=0}^{n} w_i L(t_i) \right|,$$

where w_i are the weights corresponding to the trapezoid quadrature formula. Define $L_N^I(t)$ as the linear approximation to $L(t)$ on a mesh T_N. Recall that the trapezoid approximation is exact for linear functions. Adding zero we have

$$\begin{aligned} |f(x,u) - f_N(x,u)| &= \left| \int_{t_0}^{t_f} L(t) - L_N^I(t)\, dt + \int_{t_0}^{t_f} L_N^I(t)\, dt - \sum_{i=0}^{n} w_i L(t_i) \right| \\ &= \left| \int_{t_0}^{t_f} L(t) - L_N^I(t)\, dt \right| \le \int_{t_0}^{t_f} |L(t) - L_N^I(t)|\, dt \\ &\le \int_{t_0}^{t_f} K_4 \|T_N\|\, dt = (t_f - t_0) K_4 \|T_N\| = K_6 \|T_N\|. \qquad \square \end{aligned}$$

Theorem 5.13. *Given $\epsilon > 0$, $(x,u) \in X$, and any sequence $\{(\bar{x}_N, \bar{u}_N)\} \in X_N$ which converges in norm to (x,u), then there is an N large enough so that $|f(x,u) - f_N(\bar{x}_N, \bar{u}_N)| < \epsilon$.*

Proof. We have that

$$\begin{aligned} |f_N(x,u) - f_N(\bar{x}_N, \bar{u}_N)| &= \left| \sum_{i=0}^{n} w_i (L(x(t_i), u(t_i)) - L(\bar{x}_N(t_i), \bar{u}_N(t_i))) \right| \\ &\le \sum_{i=0}^{n} w_i |L(x(t_i), u(t_i)) - L(\bar{x}_N(t_i), \bar{u}_N(t_i))|. \end{aligned}$$

Since $(\bar{x}_N, \bar{u}_N) \to^{\mathcal{B}} (x,u)$ we can choose N large enough such that

$$\|(x,u) - (\bar{x}_N, \bar{u}_N)\|_{\mathcal{B}} < \min\left\{ \Delta, \frac{\epsilon}{2K_3(t_f - t_0)} \right\},$$

so that

$$|f_N(x,u) - f_N(\bar{x}_N, \bar{u}_N)| \le \sum_{i=0}^{n} w_i K_3 \|(x(t_i), u(t_i)) - (\bar{x}_N(t_i), \bar{u}_N(t_i))\|_{\mathcal{B}}$$

implies

$$|f_N(x,u) - f_N(\bar{x}_N, \bar{u}_N)| < \frac{\epsilon}{2(t_f - t_0)} \sum_{i=0}^{n} w_i\ .$$

Note that the trapezoid quadrature is exact for constant functions, namely,

$$\sum_{i=0}^{n} w_i = t_f - t_0.$$

Hence $|f_N(x,u) - f_N(\bar{x}_N, \bar{u}_N)| < \frac{\epsilon}{2}$. But then

$$|f(x,u) - f_N(\bar{x}_N, \bar{u}_N)| \le |f(x,u) - f_N(x,u)| + |f_N(x,u) - f_N(\bar{x}_N, \bar{u}_N)|.$$

Choose N large enough so that $\|T_N\| < \frac{\epsilon}{2K_6}$. Then $|f(x,u) - f_N(\bar{x}_N, \bar{u}_N)| < \epsilon$. $\square$

6. Conclusion. We have explained why software such as SOCS can converge for inequality constrained optimal control problems where the discretization is not convergent when integrating the constrained dynamics. The optimization uses small perturbations of the inequality constraints to cancel out the large integration error. In classical numerical theory the rapid growth of perturbations is considered bad. Here it is good in that it allows the optimizer to make a small inequality perturbation counteract the large error of the nonconvergent discretization. For linear problems with convex cost functions, linear inequality constraints, and Lipschitz optimal control we have proved convergence of the discrete approximations even though the discrete multipliers may not converge in the usual sense.

It appears that the proof could be extended to more general situations. The three key assumptions appear to be the Lipschitz optimal control, convexity, and that the optimum can be approached from within the feasible set.

Consistent with numerical experience the results hold independent of the order of the constraint, or equivalently the index of the DAE when the constraints are active. This is somewhat surprising in that classical DAE integrator theory does have a strong dependence on the index. Numerical experiments show that there is definite degradation in convergence rate with increasing index, but the relationship is not obvious. This remains to be investigated.

REFERENCES

[1] J. T. Betts, *Path constrained trajectory optimization using sparse sequential quadratic programming*, J. Guidance Control Dynam., 16 (1993), pp. 59–68.

[2] J. T. Betts and P. D. Frank, *A sparse nonlinear optimization algorithm*, J. Optim. Theory Appl., 82 (1994), pp. 519–541.

[3] J. T. Betts and W. P. Huffman, *Application of sparse nonlinear programming to trajectory optimization*, J. Guidance Control Dynam., 15 (1992), pp. 198–206.

[4] J. T. Betts, N. Biehn, S. L. Campbell, and W. P. Huffman, *Convergent IRK discretizations of optimal control problems*, in 16th IMACS World Congress 2000 Proceedings, Lausanne, Switzerland, CD-ROM, IMACS, 2000.

[5] J. T. Betts, N. Biehn, S. L. Campbell, and W. P. Huffman, *Compensating for order variation in mesh refinement for direct transcription methods*, J. Comput. Appl. Math., 125 (2000), pp. 147–158.

[6] N. Biehn, S. L. Campbell, L. Jay, and T. Westbrook, *Some comments on DAE theory for IRK methods and trajectory optimization*, J. Comput. Appl. Math, 120 (2000), pp. 109–132.

[7] K. E. Brenan, S. L. Campbell, and L. R. Petzold, *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, SIAM, Philadelphia, 1996.

[8] J. Cullum, *Finite-dimensional approximations of state-constrained continuous optimal control problems*, SIAM J. Control, 10 (1972), pp. 649–670.

[9] A. L. Dontchev, *Error estimates for a discrete approximation to constrained control problems*, SIAM J. Numer. Anal., 18 (1981), pp. 500–514.

[10] A. L. Dontchev and W. W. Hager, *A new approach to Lipschitz continuity in state constrained optimal control*, Systems Control Lett., 35 (1998), pp. 137–143.

[11] A. L. Dontchev and W. W. Hager, *The Euler approximation in state constrained optimal control*, Math. Comp., 70 (2000), pp. 173–203.

[12] A. L. Dontchev, W. W. Hager, and V. M. Veliov, *Second-order Runge–Kutta approximations in control constrained optimal control*, SIAM J. Numer. Anal., 38 (2000), pp. 202–226.

[13] P. J. Enright and B. A. Conway, *Discrete approximations to optimal trajectories using direct transcription and nonlinear programming*, J. Guidance Control Dynam., 15 (1992), pp. 994–1002.

[14] R. F. Hartl, S. P. Sethi, and R. G. Vickson, *A survey of the maximum principles for optimal control problems with state constraints*, SIAM Rev., 37 (1995), pp. 181–218.

[15] R. M. M. Mattheij and J. Molenaar, *Ordinary Differential Equations in Theory and Practice*, Wiley, New York, 1996.

[16] E. Polak, *On the use of consistent approximations in the solution of semi-infinite optimization and optimal control problems*, Math. Program., 62 (1993), pp. 385–414.

[17] E. Polak, *Optimization: Algorithms and Consistent Approximations*, Springer-Verlag, New York, 1997.

[18] R. Pytlak, *Runge-Kutta based procedure for the optimal control of differential-algebraic equations*, J. Optim. Theory Appl., 97 (1998), pp. 675–705.

[19] R. Pytlak, *Numerical Methods for Optimal Control Problems with State Constraints*, Springer-Verlag, Berlin, 1999.

[20] A. Schwartz and E. Polak, *Consistent approximations for optimal control problems based on Runge–Kutta integration*, SIAM J. Control Optim., 34 (1996), pp. 1235–1269.

SIAM J. SCI. COMPUT.
Vol. 23, No. 6, pp. 2008–2040

PRINCIPAL ANGLES BETWEEN SUBSPACES IN AN A-BASED SCALAR PRODUCT: ALGORITHMS AND PERTURBATION ESTIMATES*

ANDREW V. KNYAZEV† AND MERICO E. ARGENTATI‡

Abstract. Computation of principal angles between subspaces is important in many applications, e.g., in statistics and information retrieval. In statistics, the angles are closely related to measures of dependency and covariance of random variables. When applied to column-spaces of matrices, the principal angles describe canonical correlations of a matrix pair. We highlight that all popular software codes for canonical correlations compute only cosine of principal angles, thus making impossible, because of round-off errors, finding small angles accurately. We review a combination of sine and cosine based algorithms that provide accurate results for all angles. We generalize the method to the computation of principal angles in an A-based scalar product for a symmetric and positive definite matrix A. We provide a comprehensive overview of interesting properties of principal angles. We prove basic perturbation theorems for absolute errors for sine and cosine of principal angles with improved constants. Numerical examples and a detailed description of our code are given.

Key words. principal angles, canonical correlations, subspaces, scalar product, orthogonal projection, algorithm, accuracy, round-off errors, perturbation analysis

AMS subject classifications. 65F30, 65G50, 68P20

PII. S1064827500377332

1. Introduction.

Let us consider two real-valued matrices F and G, each with n rows, and their corresponding column-spaces $\mathcal{F}$ and $\mathcal{G}$, which are subspaces in $\mathbf{R^n}$, assuming that

$$p = \dim\mathcal{F} \geq \dim\mathcal{G} = q \geq 1.$$

Then the *principal angles*

$$\theta_1, \dots, \theta_q \in [0, \pi/2]$$

between $\mathcal{F}$ and $\mathcal{G}$ may be defined, e.g., [17, 13], recursively for $k = 1, \dots, q$ by

$$\cos(\theta_k) = \max_{u\, \in\mathcal{F}} \max_{v\, \in\mathcal{G}} u^T v \;=\; u_k^T v_k$$

subject to

$$\|u\| = \|v\| = 1, \quad u^T u_i = 0, \quad v^T v_i = 0, \quad i = 1, \dots, k-1.$$

The vectors $u_1, \dots, u_q$ and $v_1, \dots, v_q$ are called principal vectors. Here and below $\|\cdot\|$ denotes the standard Euclidean norm of a vector or, when applied to a matrix, the corresponding induced operator norm, also called the spectral norm, of the matrix.

*Received by the editors August 23, 2000; accepted for publication (in revised form) November 6, 2001; published electronically February 27, 2002.
http://www.siam.org/journals/sisc/23-6/37733.html

†Department of Mathematics, University of Colorado at Denver, P.O. Box 173364, Campus Box 170, Denver, CO 80217-3364 (andrew.knyazev@cudenver.edu, http://www-math.cudenver.edu/~aknyazev)

‡Department of Mathematics, University of Colorado at Denver, P.O. Box 173364, Campus Box 170, Denver, CO 80217-3364 (rargenta@math.cudenver.edu).

According to [26, 23], the notion of canonical angles between subspaces goes back to Jordan (1875). Principal angles between subspaces, and particularly the smallest and the largest angles, serve as important tools in functional analysis (see books [1, 12, 18] and a survey [9]) and in perturbation theory of invariant subspaces, e.g., [6, 26, 24, 19, 22].

Computation of principal angles between subspaces is needed in many applications. For example, in statistics, the angles are closely related to measures of dependency and covariance of random variables; see a canonical analysis of [5]. When applied to column-spaces $\mathcal{F}$ and $\mathcal{G}$ of matrices F and G, the principal angles describe canonical correlations $\sigma_k(F, G)$ of a matrix pair, e.g., [17, 15], which is important in applications such as system identification and information retrieval. Principal angles between subspaces also appear naturally in computations of eigenspaces, e.g., [20, 21], where angles provide information about solution quality and need to be computed with high accuracy.

In such large-scale applications, it is typical that $n \gg p$; in other words, informally speaking, we are dealing with a small number of vectors having a large number of components. Because of this, we are interested in "matrix-free" methods; i.e., no n-by-n matrices need to be stored in memory in our algorithms.

A singular value decomposition (SVD)-based algorithm [11, 3, 4, 13, 15] for computing cosines of principal angles can be formulated as follows. Let columns of matrices $Q_F \in \mathbf{R^{n\times p}}$ and $Q_G \in \mathbf{R^{n\times q}}$ form orthonormal bases for the subspaces $\mathcal{F}$ and $\mathcal{G}$, respectively. The reduced SVD of $Q_F^T Q_G$ is

$$Y^T Q_F^T Q_G Z \;=\; \operatorname{diag}(\sigma_1, \sigma_2, \ldots, \sigma_q), \qquad 1 \ge \sigma_1 \ge \sigma_2 \ge \cdots \ge \sigma_q \ge 0, \tag{1.1}$$

where $Y \in \mathbf{R^{p\times q}}$, $Z \in \mathbf{R^{q\times q}}$ both have orthonormal columns. Then the principal angles can be computed as

$$\theta_k = \arccos(\sigma_k), \qquad k = 1, \ldots, q, \tag{1.2}$$

where

$$0 \le \theta_1 \le \cdots \le \theta_q \le \frac{\pi}{2},$$

while principal vectors are given by

$$u_k = Q_F y_k, \quad v_k = Q_G z_k, \quad k = 1, \ldots, q.$$

The equivalence [3, 13] of the original geometric definition of principal angles and the SVD-based approach follows from the next general theorem on an equivalent representation of singular values.

THEOREM 1.1. *If $M \in \mathbf{R^{m\times n}}$, then the singular values of M are defined recursively by*

$$\sigma_k = \max_{y\,\in \mathbf{R^m}} \max_{z\,\in \mathbf{R^n}} y^T M z \;=\; y_k^T M z_k, \qquad k = 1, \ldots, \min\{m, n\}, \tag{1.3}$$

subject to

$$\|y\| = \|z\| = 1, \quad y^T y_i = 0,\; z^T z_i = 0, \quad i = 1, \ldots, k-1. \tag{1.4}$$

The vectors y_i and z_i are, respectively, left and right singular vectors.

Proof. The proof of the theorem is straightforward if based on Allakhverdiev's representation (see [12]) of singular numbers,

$$\sigma_k = \left\| M - \sum_{i=1}^{k-1} v_i u_i^T \sigma_i \right\|,$$

and using the well-known formula of the induced Euclidean norm of a matrix as the norm of the corresponding bilinear form. □

To apply the theorem to principal angles, one takes $M = Q_F^T Q_G$.

In the most recent publication on the subject, [10], the SVD-based algorithm for cosine is proved to be mixed stable, and QR factorizations with the complete pivoting are recommended for computing Q_F and Q_G.

The SVD-based algorithm for cosine is considered as the standard one at present and is implemented in software packages, e.g., in MATLAB, version 5.3, 2000, code SUBSPACE.m, revision 5.5,[1] where $Q_F \in \mathbf{R^{n \times p}}$ and $Q_G \in \mathbf{R^{n \times q}}$ are computed using the QR factorization.

However, this algorithm cannot provide accurate results for small angles because of the presence of round-off errors. Namely, when using the standard double-precision arithmetic with $EPS \approx 10^{-16}$ the algorithm fails to accurately compute angles smaller than 10^{-8} (see section 2). The problem has been highlighted in the classical paper [3], as well as a cure has been suggested (see also publications on cosine-sine (CS) decomposition methods [25, 28, 26, 23]), but apparently it did not attract enough attention.

In statistics, most software packages include a code for computing $\sigma_k = cos(\theta_k)$, which are called *canonical correlations*; see, e.g., CANCOR Fortran code in FIRST MDS Package of AT&T, CANCR (DCANCR) Fortran subroutine in IMSL STAT/ LIBRARY, G03ADF Fortran code in NAG package, CANCOR subroutine in Splus, and CANCORR procedure in SAS/STAT Software. While accurately computing the cosine of principal angles in corresponding precision, these codes do not compute the sine. However, the cosine simply equals one in double precision for all angles smaller than 10^{-8} (see next section). Therefore, it is impossible in principal to observe an improvement in canonical correlations for angles smaller than 10^{-8} in double precision. It might not be typically important when processing experimental statistical data because the expected measurement error may be so great that a statistician would deem the highly correlated variable essentially redundant and therefore not useful as a further explanatory variable in their model. Statistical computer experiments are different, however, as there is no measurement error, so accurate computation of very high correlations may be important in such applications.

The largest principal angle is related to the notion of distance, or a gap, between equidimensional subspaces. If $p = q$, the distance is defined [1, 12, 13, 18] as

$$\text{gap}(\mathcal{F}, \mathcal{G}) = \|P_F - P_G\| = \sin(\theta_q) = \sqrt{1 - (\cos(\theta_q))^2}, \tag{1.5}$$

where P_F and P_G are orthogonal projectors onto $\mathcal{F}$ and $\mathcal{G}$, respectively.

This formulation provides insight into a possible alternative algorithm for computing the sine of principal angles. The corresponding algorithm, described in [3],

[1] Revision 5.8 of SUBSPACE.m in the current MATLAB release 12.1, version 6.1.0.450, May 18, 2001, is still identical to revision 5.5, which we have used for numerical tests in the present paper.

while being mathematically equivalent to the previous one in exact arithmetic, is accurate for small angles in computer arithmetic as it computes the sine of principal angles directly, without using SVD (1.1) leading to the cosine. We review the algorithm of [3] based on a general form of (1.5) in section 3 and suggest an improved version, similar to the CS decomposition algorithm of [28], with the second SVD of the reduced size.

The CS decomposition methods, e.g., [25, 28, 26, 23], one of which we just mentioned, provide a well-known and popular alternative approach for computing principal angles between subspaces given by selected p (q) columns of orthogonal matrices of the size n. For example, if the matrix $Q_{F^\perp}$, with orthonormal columns that span the subspace $\mathcal{F}^\perp$, the orthogonal complement of $\mathcal{F}$, is available to us, the CS decomposition methods compute the SVD of $(Q_F)^T Q_G$ together with the SVD of $(Q_{F^\perp})^T Q_G$, thus providing cosine and sine of principal angles. When p is of the same order as $n/2$, matrix $Q_{F^\perp}$ is about of the same size as matrix Q_F, and the CS decomposition methods are effective and are recommended for practical computations. However, when $n \gg p$, the CS decomposition methods, explicitly using matrix $Q_{F^\perp}$ of the size n-by-$n-p$, will be less efficient compared to "matrix-free" methods we consider in the present paper. Let us highlight that the cosine- and sine-based methods of [3] that we investigate here in section 3, while different algorithmically from the CS decomposition methods, are very close mathematically to them.

A different sine-based approach, using eigenvalues of $P_F - P_G$, is described in [4, 23]; see a similar statement of Theorem 3.4. It is also not attractive numerically, when $n \gg p$, as it requires computing an n-by-n matrix and finding all its nonzero eigenvalues.

In some applications, e.g., when solving symmetric generalized eigenvalue problems [20], the default scalar product $u^T v$ cannot be used and needs to be replaced with an A-based scalar product $(u, v)_A = u^T A v$, where A is a symmetric positive definite matrix. In statistics, a general scalar product for computing canonical correlations gives a user an opportunity, for example, to take into account a priori information that some vector components are more meaningful than others. In a purely mathematical setting, generalization to A-based scalar products brings nothing really new. In practical computations, however, it carries numerous algorithmic and numerical problems, especially for ill-conditioned cases, which are important in applications.

In section 4, we propose extension of the algorithms to an A-based scalar product and provide the corresponding theoretical justification.

In section 5, we turn our attention to perturbation estimates, which generalize the following trigonometric inequalities: if an angle $\theta \in [0, \pi/2]$ is perturbed by $\epsilon \in [0, \pi/2]$ such that $\theta + \epsilon \in [0, \pi/2]$, then

$$0 \le \cos(\theta) - \cos(\theta + \epsilon) \le \sin(\theta + \epsilon)\sin(\epsilon) \le \sin(\epsilon),$$

$$0 \le \sin(\theta + \epsilon) - \sin(\theta) \le \cos(\theta)\sin(\epsilon) \le \sin(\epsilon).$$

We prove new absolute perturbation estimates for the sine and cosine of principal angles computed in the A-based scalar product. When $A = I$, our estimates are similar to those of [3, 29, 27, 15, 14], but the technique we use is different. More importantly, our constants are somewhat better, in fact, in the same way the constants in the middle terms of the trigonometric inequalities above are less than one.

We consider particular implementation of algorithms used in our MATLAB code SUBSPACEA.m in section 6, with emphasis on the large-scale case, $n \gg p$, and sparse ill-conditioned matrix A, which may be specified only as a function that multiplies A by a given vector. When matrices F and G are sparse, our code can still be used even though it performs orthogonalization of columns of matrices F and G that increases the fill-in; cf. [15]. Also, we do not even touch here upon a practically important issue of the possibility of recomputing the correlations with an increase in the data; see again [15].

Finally, numerical results, presented in section 7, demonstrate the practical robustness of our code.

For simplicity, we discuss only real spaces and real scalar products; however, all results can be trivially generalized to cover complex spaces as well. In fact, our code SUBSPACEA.m is written for the general complex case.

As pointed out by an anonymous referee, several natural questions are left unanswered here.

- Our algorithms are based on SVD. How does SVD accuracy (cf. [2, 8, 7, 10]), especially for small singular values, or in ill-conditioned cases, affect the results?
- In [10], a formal stability analysis is done for the SVD-based algorithm for cosine, which is proved to be mixed stable. In our numerical tests, practical robustness of our algorithms is encouraging. Are our methods accurate and stable theoretically, e.g., see [16]?
- For A-based scalar products, how does the increase of the condition number of A influence the accuracy? Which parts of our algorithm are responsible for the main error growth?

We feel, however, that investigating these matters is not within the limited scope of the present paper, which is already quite long. They may rather serve as interesting directions for future research.

2. Inaccuracy in the cosine-based algorithm.

Let d be a constant and

$$\mathcal{F} = \operatorname{span}\left\{(1\ 0)^T\right\}, \qquad \mathcal{G} = \operatorname{span}\left\{(1\ d)^T\right\}.$$

Then the angle between the one-dimensional subspaces $\mathcal{F}$ and $\mathcal{G}$ can be computed as

$$\theta = \arcsin\left(\frac{d}{\sqrt{1+d^2}}\right). \tag{2.1}$$

In the table below d varies from one to small values. Formula (2.1) is accurate for small angles, so we use it as an "exact" answer in the second column of the table. We use the MATLAB built-in function SUBSPACE.m (revision 5.5) which implements (1.1) to compute values for the third column of the table.

It is apparent that SUBSPACE.m returns inaccurate results for $d \le 10^{-8}$, which is approximately $\sqrt{EPS}$ for double precision.

d	Formula (2.1)	SUBSPACE.m
1.0	7.853981633974483e-001	7.853981633974483e-001
1.0e-04	9.999999966666666e-005	9.999999986273192e-005
1.0e-06	9.999999999996666e-007	1.000044449242271e-006
1.0e-08	1.000000000000000e-008	-6.125742274543099e-017
1.0e-10	1.000000000000000e-010	-6.125742274543099e-017
1.0e-16	9.999999999999998e-017	-6.125742274543099e-017
1.0e-20	9.999999999999998e-021	-6.125742274543099e-017
1.0e-30	1.000000000000000e-030	-6.125742274543099e-017

In this simple one-dimensional example the algorithm of SUBSPACE.m is reduced to computing

$$\theta = \arccos\left(\frac{1}{\sqrt{1+d^2}}\right).$$

This formula clearly shows that the inability to compute accurately small angles is integrated in the standard algorithm and cannot be fixed without changing the algorithm itself. The cosine, that is, a canonical correlation, is computed accurately and simply equals to one for all positive $d \leq 10^{-8}$. However, one cannot determine small angles from a cosine accurately in the presence of round-off errors. In statistical terms, it illustrates the problem we already mentioned above that the canonical correlation itself does not show any improvement in correlation when d is smaller than 10^{-8} in double precision.

In the next section, we consider a formula [3] that directly computes the sine of principal angles as in (2.1).

3. Properties of principal angles and a sine-based algorithm. We first review known sine-based formulas for the largest principal angle. Results of [1, 18] concerning the aperture of two linear manifolds give

$$\|P_F - P_G\| = \max\left\{\max_{x\in\mathcal{G},\|x\|=1}\|(I-P_F)x\|,\ \max_{y\in\mathcal{F},\|y\|=1}\|(I-P_G)y\|\right\}. \tag{3.1}$$

Let columns of matrices $Q_F \in \mathbf{R^{n\times p}}$ and $Q_G \in \mathbf{R^{n\times q}}$ form orthonormal bases for the subspaces $\mathcal{F}$ and $\mathcal{G}$, respectively. Then orthogonal projectors on $\mathcal{F}$ and $\mathcal{G}$ are $P_F = Q_F Q_F^T$ and $P_G = Q_G Q_G^T$, respectively, and

$$\|P_F - P_G\| = \max\{\|(I - Q_F Q_F^T)Q_G\|, \|(I - Q_G Q_G^T)Q_F\|\}. \tag{3.2}$$

If $p \neq q$, then expression of (3.2) is always equal to one; e.g., if $p > q$, then the second term under the maximum is one. If $p = q$, then both terms are the same and yield $\sin(\theta_q)$ by (1.5). Therefore, under our assumption $p \geq q$, only the first term is interesting to analyze. We note that the first term is the largest singular value of $(I - Q_F Q_F^T)Q_G$. What is the meaning of other singular values of the matrix?

This provides an insight into how to find a sine-based formulation to obtain the principal angles, which is embodied in the following theorem [3].

THEOREM 3.1. *Singular values $\mu_1 \leq \mu_2 \leq \cdots \leq \mu_q$ of the matrix $(I - Q_F Q_F^T)Q_G$ are given by $\mu_k = \sqrt{1-\sigma_k^2}$, $k = 1, \dots, q$, where σ_k are defined in* (1.1). *Moreover, the principal angles satisfy the equalities $\theta_k = \arcsin(\mu_k)$.*

The right principal vectors can be computed as

$$v_k = Q_G z_k, \qquad k = 1, \dots, q,$$

where z_k are corresponding orthonormal right singular vectors of matrix $(I - Q_F Q_F^T)Q_G$. The left principal vectors are then computed by

$$u_k = Q_F Q_F^T v_k/\sigma_k \quad \text{if } \sigma_k \neq 0, \quad k = 1, \dots, q.$$

Proof. Our proof is essentially the same as that of [3]. We reproduce it here for completeness as we use a similar proof later for a general scalar product.

Let $B = (I - P_F)Q_G = (I - Q_F Q_F^T)Q_G$. Using the fact that $I - P_F$ is a projector and that $Q_G^T Q_G = I$, we have

$$\begin{aligned} B^T B &= Q_G^T(I - P_F)(I - P_F)Q_G = Q_G^T(I - P_F)Q_G \\ &= I - Q_G^T Q_F Q_F^T Q_G. \end{aligned}$$

Utilizing the SVD (1.1), we obtain $Q_F^T Q_G = Y \Sigma Z^T$, where $\Sigma = \operatorname{diag}(\sigma_1, \sigma_2, \ldots, \sigma_q)$; then

$$Z^T B^T B Z = I - \Sigma^2 = \operatorname{diag}(1 - \sigma_1^2, 1 - \sigma_2^2, \ldots, 1 - \sigma_q^2).$$

Thus, the singular values of B are given by $\mu_k = \sqrt{1 - \sigma_k^2}$, $k = 1, \ldots, q$, and the formula for the principal angles $\theta_k = \arcsin(\mu_k)$ follows directly from (1.2). □

We can now use the theorem to formulate an algorithm for computing all the principal angles. This approach meets our goal of a sine-based formulation, which should provide accurate computation of small angles. However, for large angles we keep the cosine-based algorithm.

```
ALGORITHM 3.1: Modified SUBSPACE.m.
Input: real matrices $F$ and $G$ with the same number of rows.
1. Compute orthonormal bases $Q_F = \operatorname{orth}(F)$, $Q_G = \operatorname{orth}(G)$ of column-spaces of $F$ and $G$.
2. Compute SVD for cosine: $[Y, \Sigma, Z] = \operatorname{svd}(Q_F^T Q_G)$, $\Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_q)$.
3. Compute matrices of left $U_{\cos} = Q_F Y$ and right $V_{\cos} = Q_G Z$ principal vectors.
4. Compute matrix $B = \begin{cases} Q_G - Q_F(Q_F^T Q_G) & \text{if } \operatorname{rank}(Q_F) \ge \operatorname{rank}(Q_G); \\ Q_F - Q_G(Q_G^T Q_F) & \text{otherwise.} \end{cases}$
5. Compute SVD for sine: $[Y, \operatorname{diag}(\mu_1, \ldots, \mu_q), Z] = \operatorname{svd}(B)$.
6. Compute matrices $U_{\sin}$ and $V_{\sin}$ of left and right principal vectors:
       $V_{\sin} = Q_G Z$, $U_{\sin} = Q_F(Q_F^T V_{\sin})\Sigma^{-1}$   if $\operatorname{rank}(Q_F) \ge \operatorname{rank}(Q_G)$;
       $U_{\sin} = Q_F Z$, $V_{\sin} = Q_G(Q_G^T U_{\sin})\Sigma^{-1}$   otherwise.
7. Compute the principal angles, for $k = 1, \ldots, q$:
       $\theta_k = \begin{cases} \arccos(\sigma_k) & \text{if } \sigma_k^2 < 1/2; \\ \arcsin(\mu_k) & \text{if } \mu_k^2 \le 1/2. \end{cases}$
8. Form matrices $U$ and $V$ by picking up corresponding columns of $U_{\sin}$, $V_{\sin}$ and $U_{\cos}$, $V_{\cos}$, according to the choice for $\theta_k$ above.
Output: Principal angles $\theta_1, \ldots, \theta_q$ between column-spaces of matrices $F$ and $G$, and
    corresponding matrices $U$ and $V$ of left and right principal vectors, respectively.
```

REMARK 3.1. *In step 1 of the algorithm, the orthogonalization can be performed using the QR method or the SVD. In our actual code, an SVD-based built-in MATLAB function ORTH.m is used for the orthogonalization.*

It is pointed out in [10] *that errors in computing Q_F and Q_G, especially expected for ill-conditioned F and G, may lead to an irreparable damage in final answers. A proper column scaling of F and G could in some cases significantly reduce condition numbers of F and G. We highlight that an explicit columnwise normalization of matrices F and G is not required prior to orthogonalization if a particular orthogonalization algorithm used here is invariant under column scaling in finite precision arithmetic. Our numerical tests show that the explicit column scaling is not needed if we utilize a built-in MATLAB function QR.m for orthonormalization. However, the explicit column scaling apparently helps to improve the accuracy when the SVD-based* *built-in MATLAB function ORTH.m is used for orthonormalization. In* [10]*, QR factorizations with complete pivoting are recommended for computing Q_F and Q_G.*

REMARK 3.2. *A check* $\operatorname{rank}(Q_F) \geq \operatorname{rank}(Q_G)$ *in steps* 4 *and* 6 *of the algorithm removes the need for our assumption* $p = \operatorname{rank}(Q_F) \geq \operatorname{rank}(Q_G) = q$.

REMARK 3.3. *We replace here in step* 4

$$(I - Q_F Q_F^T)Q_G = Q_G - Q_F(Q_F^T Q_G), \quad (I - Q_G Q_G^T)Q_F = Q_F - Q_G(Q_G^T Q_F)$$

to avoid storing in memory any n-by-n matrices in the algorithm, which allows us to compute principal angles efficiently for large $n \gg p$ as well.

If the matrix $Q_{F^\perp}$, with orthonormal columns that span the subspace $\mathcal{F}^\perp$, the orthogonal complement of $\mathcal{F}$, was available to us when $p \geq q$, we could take here

$$B = Q_{F^\perp} Q_G,$$

as in the CS decomposition methods; see [25, 28, 26, 23]*. Under our assumption $n \gg p$, however, the matrix $Q_{F^\perp}$ is essentially of the size n and thus shall be avoided.*

The 1/2 threshold used in Algorithm 3.1 in steps 7 and 8 to separate small and large principal angles and corresponding vectors seems to be a natural choice. However, such an artificial fixed threshold may cause troubles with orthogonality in the resulting choice of vectors in step 8 if there are several angles close to each other but on different sides of the threshold. The problem is that the corresponding principal vectors, picked up from two orthogonal sets computed by different algorithms, may not be orthogonal. A more accurate approach would be to identify such possible cluster of principal angles around the original threshold and to make sure that all principal vectors corresponding to the cluster are chosen according to either step 3, or step 6, but not both.

ALGORITHM 3.2: **Modified and Improved SUBSPACE.m.**

Input: *real matrices F and G with the same number of rows.*

```
1.  Compute orthonormal bases $Q_F = \operatorname{orth}(F)$, $Q_G = \operatorname{orth}(G)$ of column-spaces of $F$ and $G$.
2.  Compute SVD for cosine: $[Y, \Sigma, Z] = \operatorname{svd}(Q_F^T Q_G)$, $\Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_q)$.
3.  Compute matrices of left $U_{\cos} = Q_F Y$ and right $V_{\cos} = Q_G Z$ principal vectors.
4.  Compute large principal angles, for $k = 1, \ldots, q$:
        $\theta_k = \arccos(\sigma_k)$ if $\sigma_k^2 < 1/2$.
5.  Form parts of matrices $U$ and $V$ by picking up corresponding columns of $U_{\cos}$, $V_{\cos}$,
    according to the choice for $\theta_k$ above. Put columns of $U_{\cos}$, $V_{\cos}$, which are left,
    in matrices $R_F$ and $R_G$. Collect the corresponding $\sigma$'s in a diagonal matrix $\Sigma_R$.
6.  Compute the matrix $B = R_G - Q_F(Q_F^T R_G)$.
7.  Compute SVD for sine: $[Y, \operatorname{diag}(\mu_1, \ldots, \mu_q), Z] = \operatorname{svd}(B)$.
8.  Compute matrices $U_{\sin}$ and $V_{\sin}$ of left and right principal vectors:
        $V_{\sin} = R_G Z$, $U_{\sin} = R_F(R_F^T V_{\sin})\Sigma_R^{-1}$.
9.  Recompute the small principal angles, for $k = 1, \ldots, q$:
        $\theta_k = \arcsin(\mu_k)$ if $\mu_k^2 \leq 1/2$.
10. Complete matrices $U$ and $V$ by adding columns of $U_{\sin}$, $V_{\sin}$.
```

Output: *Principal angles $\theta_1, \ldots, \theta_q$ between column-spaces of matrices F and G, and corresponding matrices U and V of left and right principal vectors, respectively.*

Let us repeat that, in exact arithmetic, the sine and cosine based approaches give the same results; e.g., columns of $U_{\sin}$ and $V_{\sin}$ must be the same as those of $U_{\cos}$

and $V_{\cos}$. Why do we need to recompute essentially the same vectors a second time? What if we compute only $U_{\cos}$, $V_{\cos}$ and then recompute just small principal angles using, e.g., the obvious formula

$$\mu_k = \|u_k - \sigma_k v_k\|? \tag{3.3}$$

An anonymous referee recommended this approach and suggested that it would resolve the inaccuracy in the cosine-based algorithm illustrated in the previous section, without the need for the second SVD.

The answer is that the cosine-based algorithm fails to compute accurately not only the small principal angles but also the corresponding principal vectors. The reason for this is that singular values computed in step 2 of Algorithm 3.1 are the cosines of principal angles, while singular values of the matrix B in step 5 are the sines of principal angles. Thus, the distribution of singular values is different in steps 2 and 5; e.g., singular values corresponding to small angles are much better separated in step 5 than in step 2. For example, angles 10^{-10} and 10^{-12} will produce a multiple singular value 1 in step 2 in double precision but will produce two distinct small singular values in step 5. This means that singular vectors, corresponding to small principal angles, might not be computed accurately in computer arithmetic using only SVD in step 2, which will also lead to inaccurate computation of the small principal angles by formula (3.3). Our numerical tests support this conclusion.

There is some obvious redundancy in Algorithm 3.1. Indeed, we do not need to calculate columns of $U_{\sin}$ and $V_{\sin}$, corresponding to large sines, and columns of $U_{\cos}$ and $V_{\cos}$, corresponding to large cosines, as they are computed inaccurately in computer arithmetic and we just discard them later in the algorithm. However, first, for large-scale applications with $n \gg p$ that we are interested in, the redundancy is insignificant. Second, this allows us to perform steps 2–3 and steps 4–5 of Algorithm 3.1 independently in parallel. We note that Σ must be invertible in Algorithm 3.1.

For sequential computations, we now describe Algorithm 3.2. Here, we reduce computational costs of the second SVD by using already computed vectors $U_{\cos}$ and $V_{\cos}$ for the cosines. The cosine-based algorithm computes inaccurately individual principal vectors corresponding to small principal angles. However, it may find accurately the corresponding invariant subspaces spanned by all these vectors. Thus, the idea is that, using $U_{\cos}$ and $V_{\cos}$, we can identify invariant subspaces in $\mathcal{F}$ and $\mathcal{G}$, which correspond to all small principal angles. Then, we perform the second SVD only for these subspaces, computing only columns of $U_{\sin}$ and $V_{\sin}$ that we actually need, which may significantly reduce the size of the matrix in the second SVD. This idea is used in the CS decomposition algorithm of [28].

We keep steps 1–3 of Algorithm 3.1 unchanged but modify accordingly later steps to obtain Algorithm 3.2. Such changes may significantly reduce the size of matrix B and, thus, the costs, if large principal angles outnumber small ones; e.g., if there are no small principal angles at all, the algorithm simply stops at step 3. We note that matrix Σ_R is always invertible, unlike matrix Σ.

Remark 3.4. *By construction, matrices R_F and R_G have the same number of already orthogonal columns, which removes the need for orthogonalization and for comparing, in step* 6*, their ranks.*

Remark 3.5. *We have three, equivalent in exact arithmetic, possibilities to compute matrix B:*

$$B = (I - R_F R_F^T)R_G = R_G - R_F(R_F^T R_G) = R_G - Q_F(Q_F^T R_G).$$

The first formula is ruled out to avoid storing in memory any n-by-n matrices in the algorithm. Our numerical tests show that the third expression, though somewhat more expensive than the second one, often provides more accurate results in the presence of round-off errors.

To summarize, Algorithms 3.1 and 3.2 use the cosine-based formulation (1.1), (1.2) for large angles and the sine-based formulation of Theorem 3.1 for small angles, which allows accurate computation of all angles. The algorithms are reasonably efficient for large-scale applications with $n \gg p$ and are more robust than the original cosine-based only version.

In the rest of the section, we describe some useful properties of principal angles not yet mentioned. In the present paper, we follow [19] and make use of an orthogonal projectors technique. For an alternative approach, popular in matrix theory, which is based on representation of subspaces in a canonical CS form, we refer to [26].

Theorem 3.1 characterizes singular values of the product $(I - P_F)Q_G$, which are the sine of the principal angles. What are singular values of the matrix $P_F Q_G$? A trivial modification of the previous proof leads to the following not really surprising result that these are the cosine of the principal angles.

THEOREM 3.2. *Singular values of the matrix $Q_F Q_F^T Q_G$ are exactly the same as σ_k, defined in* (1.1).

We conclude this section with other simple and known (e.g., [29, 26, 30]) sine and cosine representations of principal angles and principal vectors, this time using orthogonal projectors P_F and P_G on subspaces $\mathcal{F}$ and $\mathcal{G}$, respectively.

THEOREM 3.3. *Let assumptions of Theorem* 3.1 *be satisfied. Then $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_q$ are the q largest singular values of the matrix $P_F P_G$; in particular,*

$$\sigma_1 = \|P_F P_G\|.$$

Other $n - q$ singular values are all equal to zero.

REMARK 3.6. *As singular values of $P_F P_G$ are the same as those of $P_G P_F$, subspaces $\mathcal{F}$ and $\mathcal{G}$ play symmetric roles in Theorem* 3.3*; thus, our assumption that $p = \dim\mathcal{F} \geq \dim\mathcal{G} = q$ is irrelevant here.*

THEOREM 3.4. *Let assumptions of Theorem* 3.1 *be satisfied. Then $\mu_1 \leq \mu_2 \leq \cdots \leq \mu_q$ are the q largest singular values of the matrix $(I - P_F)P_G$; in particular,*

$$\mu_q = \|(I - P_F)P_G\|.$$

Other $n - q$ singular values are all equal to zero.

REMARK 3.7. *Comparing Theorems* 3.3 *and* 3.4 *shows trivially that sine of principal angles between $\mathcal{F}$ and $\mathcal{G}$ are the same as cosine of principal angles between $\mathcal{F}^\perp$ and $\mathcal{G}$ because $I - P_F$ is an orthogonal projector on $\mathcal{F}^\perp$. If $p > n/2 > q$, it may be cheaper to compute principal angles between $\mathcal{F}^\perp$ and $\mathcal{G}$ instead of principal angles between $\mathcal{F}$ and $\mathcal{G}$.*

What can we say about singular values of the matrix $(I - P_G)P_F$? In other words, how do cosine of principal angles between subspaces $\mathcal{F}^\perp$ and $\mathcal{G}$ compare to cosine of principal angles between their orthogonal complements $\mathcal{F}$ and $\mathcal{G}^\perp$? If $p = q$, they are absolutely the same; in particular, the minimal angle between subspaces $\mathcal{F}^\perp$ and $\mathcal{G}$ is in this case the same as the minimal angle between their orthogonal complements $\mathcal{F}$ and $\mathcal{G}^\perp$, e.g., in [9], and, in fact, is equal to $\text{gap}(\mathcal{F}, \mathcal{G}) = \|P_F - P_G\|$ as we already discussed. When $p > q$, subspaces $\mathcal{F}$ and $\mathcal{G}^\perp$ must have a nontrivial intersection because the sum of their dimensions is too big; thus, the minimal angle between subspaces $\mathcal{F}$ and $\mathcal{G}^\perp$ must be zero in this case, which corresponds to $\|(I - P_G)P_F\| = 1$, while $\|(I - P_F)P_G\|$ may be less than one. To be more specific, $\dim(\mathcal{F} \cap \mathcal{G}^\perp) \geq p - q$; thus, at least $p - q$ singular values of the matrix $(I - P_G)P_F$ are equal to one. Then, we have the following statement, which completely clarifies the issue of principal angles between orthogonal complements; cf. Ex. 1.2.6 of [4].

THEOREM 3.5. *The set of singular values of $(I - P_G)P_F$, when $p > q$, consists of $p - q$ ones, q singular values of $(I - P_F)P_G$, and $n - p$ zeros.*

In particular, this shows that the smallest positive sine of principal angles between $\mathcal{F}$ and $\mathcal{G}$, called the *minimum gap*, is the same as that between $\mathcal{F}^\perp$ and $\mathcal{G}^\perp$; see [18]. This theorem can also be used to reduce the costs of computing the principal angles between subspaces $\mathcal{F}$ and $\mathcal{G}$, when their dimensions p and q are greater than $n/2$, by replacing $\mathcal{F}$ and $\mathcal{G}$ with their orthogonal complements.

Let us finally mention a simple property of principal vectors, emphasized in [29], which helps us to understand a geometric meaning of pairs of corresponding principal vectors from different subspaces.

THEOREM 3.6. *We have*

$$P_F v_k = \sigma_k u_k, \quad P_G u_k = \sigma_k v_k, \quad k = 1, \ldots, q,$$

and

$$u_i^T v_j = (P_F u_i)^T v_j = u_i^T P_F v_j = \sigma_j u_i^T u_j = \sigma_j \delta_{ij}, \qquad i, j = 1, \ldots, q.$$

In other words, a chosen pair u_k, v_k spans a subspace, invariant with respect to orthoprojectors P_F and P_G and orthogonal to all other such subspaces. The kth principal angle θ_k is simply the angle between u_k and v_k; see (3.3).

Moreover, the subspace $\operatorname{span}\{u_k, v_k\}$ *is also invariant with respect to orthoprojectors $I - P_F$ and $I - P_G$. Let us define two other unit vectors in this subspace:*

$$u_k^\perp = (v_k - \sigma_k u_k)/\mu_k \in \mathcal{F}^\perp, \quad v_k^\perp = (u_k - \sigma_k v_k)/\mu_k \in \mathcal{G}^\perp$$

such that $u_k^T u_k^\perp = v_k^T v_k^\perp = 0$. Then

- *u_k, v_k are principal vectors for subspaces $\mathcal{F}$ and $\mathcal{G}$;*
- *$u_k^\perp, v_k$ are principal vectors for subspaces $\mathcal{F}^\perp$ and $\mathcal{G}$;*
- *$u_k, v_k^\perp$ are principal vectors for subspaces $\mathcal{F}$ and $\mathcal{G}^\perp$;*
- *$u_k^\perp, -v_k^\perp$ are principal vectors for subspaces $\mathcal{F}^\perp$ and $\mathcal{G}^\perp$,*

which concludes the description of all cases.

In the next section, we deal with an arbitrary scalar product.

4. Generalization to an A-based scalar product.

Let $A \in \mathbf{R}^{\mathbf{n}\times\mathbf{n}}$ be a fixed symmetric positive definite matrix. Let $(x, y)_A = (x, Ay) = y^T A x$ be an A-based scalar product, $x, y \in \mathbf{R}^{\mathbf{n}}$. Let $\|x\|_A = \sqrt{(x, x)_A}$ be the corresponding vector norm and let $\|B\|_A$ be the corresponding induced matrix norm of a matrix $B \in \mathbf{R}^{\mathbf{n}\times\mathbf{n}}$. We note that $\|x\|_A = \|A^{1/2}x\|$ and $\|B\|_A = \|A^{1/2}BA^{-1/2}\|$.

In order to define principal angles based on this scalar product, we will follow arguments of [3, 13] but in an A-based scalar product instead of the standard Euclidean scalar product. Again, we will assume for simplicity of notation that $p \geq q$.

Principal angles

$$\theta_1, \ldots, \theta_q \in [0, \pi/2]$$

between subspaces $\mathcal{F}$ and $\mathcal{G}$ in the A-based scalar product $(\cdot, \cdot)_A$ are defined recursively for $k = 1, \ldots, q$ by analogy with the previous definition for $A = I$ as

$$\cos(\theta_k) = \max_{u \in \mathcal{F}} \max_{v \in \mathcal{G}} (u, v)_A = (u_k, v_k)_A \tag{4.1}$$

subject to

$$\|u\|_A = \|v\|_A = 1, \quad (u, u_i)_A = 0, \quad (v, v_i)_A = 0, \quad i = 1, \ldots, k-1. \tag{4.2}$$

The vectors $u_1, \ldots, u_q$ and $v_1, \ldots, v_q$ are called principal vectors relative to the A-based scalar product.

The following theorem justifies the consistency of the definition above and provides a cosine-based algorithm for computing the principal angles in the A-based scalar product. It is a direct generalization of the cosine-based approach of [3, 13].

THEOREM 4.1. *Let columns of* $Q_F \in \mathbf{R^{n\times p}}$ *and* $Q_G \in \mathbf{R^{n\times q}}$ *now be A-orthonormal bases for the subspaces F and G, respectively. Let* $\sigma_1 \ge \sigma_2 \ge \cdots \ge \sigma_q$ *be singular values of* $Q_F^T A Q_G$ *with corresponding left and right singular vectors y_k and z_k,* $k = 1, \ldots, q$. *Then the principal angles relative to the scalar product* $(\cdot,\cdot)_A$ *as defined in* (4.1) *and* (4.2) *are computed as*

$$\theta_k = \arccos(\sigma_k), \qquad k = 1, \ldots, q, \tag{4.3}$$

where

$$0 \le \theta_1 \le \cdots \le \theta_q \le \frac{\pi}{2},$$

while the principal vectors are given by

$$u_k = Q_F y_k, \quad v_k = Q_G z_k, \quad k = 1, \ldots, q.$$

Proof. We first rewrite definition (4.1) and (4.2) of principal angles in the following equivalent form. For $k = 1, \ldots, q$,

$$\cos(\theta_k) = \max_{y\, \in \mathbf{R^p}} \max_{z\, \in \mathbf{R^q}} y^T Q_F^T A Q_G z \;=\; y_k^T Q_F^T A Q_G z_k$$

subject to

$$\|y\| = \|z\| = 1, \quad y^T y_i = 0, \quad z^T z_i = 0, \quad i = 1, \ldots, k-1,$$

where $u = Q_F y \in \mathcal{F}$, $v = Q_G z \in \mathcal{G}$ and $u_k = Q_F y_k \in \mathcal{F}$, $v_k = Q_G z_k \in \mathcal{G}$.

Since Q_F and Q_G have A-orthonormal columns, $Q_F^T A Q_F = I$ and $Q_G^T A Q_G = I$. This implies

$$\|u\|_A^2 = y^T Q_F^T A Q_F y = y^T y = \|y\|^2 = 1$$

and

$$\|v\|_A^2 = z^T Q_G^T A Q_G z = z^T z = \|z\|^2 = 1.$$

For $i \ne j$, we derive

$$(u_i, u_j)_A = y_i^T Q_F^T A Q_F y_j = y_i^T y_j = 0$$

and

$$(v_i, v_j)_A = z_i^T Q_G^T A Q_G z_j = z_i^T z_j = 0.$$

Now, let the reduced SVD of $Q_F^T A Q_G$ be

$$Y^T Q_F^T A Q_G Z \;=\; \operatorname{diag}(\sigma_1, \sigma_2, \ldots, \sigma_q), \tag{4.4}$$

where $Y \in \mathbf{R}^{\mathbf{p}\times\mathbf{q}}$, $Z \in \mathbf{R}^{\mathbf{q}\times\mathbf{q}}$ both have orthonormal columns.

Then, by Theorem 1.1 with $M = Q_F^T A Q_G$, the equality $\cos(\theta_k) = \sigma_k$, $k = 1, \ldots, q$, just provides two equivalent representations of the singular values of $Q_F^T A Q_G$, and y_z and z_k can be chosen as columns of matrices Y and Z, respectively. The statement of the theorem follows. □

Let us now make a trivial but important observation that links principal angles in the A-based scalar product with principal angles in the original standard scalar product, when a factorization of $A = K^T K$, e.g., $K = A^{1/2}$, is available. We formulate it as the following theorem.

THEOREM 4.2. *Let $A = K^T K$. Under assumptions of Theorem* 4.1 *the principal angles between subspaces $\mathcal{F}$ and $\mathcal{G}$ relative to the scalar product $(\cdot,\cdot)_A$ coincide with the principal angles between subspaces $K\mathcal{F}$ and $K\mathcal{G}$ relative to the original scalar product $(\cdot,\cdot)$.*

Proof. One way to prove this is to notice that our definition of the principal angles between subspaces $\mathcal{F}$ and $\mathcal{G}$ relative to the scalar product $(\cdot,\cdot)_A$ turns into a definition of the principal angles between subspaces $K\mathcal{F}$ and $K\mathcal{G}$ relative to the original scalar product $(\cdot,\cdot)$ if we make substitutions $Ku \mapsto u$ and $Kv \mapsto v$.

Another proof is to use the representation

$$Q_F^T A Q_G = (KQ_F)^T KQ_G,$$

where columns of matrices KQ_F and KQ_G are orthonormal with respect to the original scalar product $(\cdot,\cdot)$ and span subspaces $K\mathcal{F}$ and $K\mathcal{G}$, respectively. Now Theorem 4.1 is equivalent to the traditional SVD theorem on cosine of principal angles between subspaces $K\mathcal{F}$ and $K\mathcal{G}$ relative to the original scalar product $(\cdot,\cdot)$, formulated in the introduction. □

The A-orthogonal projectors on subspaces $\mathcal{F}$ and $\mathcal{G}$ are now defined by formulas

$$P_F = Q_F Q_F^{*_A} = Q_F Q_F^T A \text{ and } P_G = Q_G Q_G^{*_A} = Q_G Q_G^T A,$$

where $*_A$ denotes the A-adjoint.

To obtain a sine-based formulation in the A-based scalar product that is accurate for small angles, we first adjust (1.5) and (3.1) to the new A-based scalar product:

$$\begin{aligned} \operatorname{gap}_A(\mathcal{F},\mathcal{G}) &= \|P_F - P_G\|_A \\ &= \max\left\{\max_{x\in\mathcal{G},\|x\|_A=1} \|(I-P_F)x\|_A,\ \max_{y\in\mathcal{F},\|y\|_A=1} \|(I-P_G)y\|_A\right\}. \end{aligned} \tag{4.5}$$

If $p = q$, this equation will yield $\sin(\theta_q)$, consistent with Theorem 4.1. Similar to the previous case $A = I$, only the first term under the maximum is of interest under our assumption that $p \geq q$. Using the fact that

$$\|x\|_A = \|Q_G z\|_A = \|z\| \ \forall x \in \mathcal{G}, \qquad x = Q_G z,\ z \in \mathbf{R}^{\mathbf{q}},$$

the term of interest can be rewritten as

$$\max_{x\in\mathcal{G},\|x\|_A=1} \|(I-P_F)x\|_A = \|A^{1/2}(I - Q_F Q_F^T A)Q_G\|. \tag{4.6}$$

Here we use the standard induced Euclidean norm $\|\cdot\|$ for computational purposes. Similar to our arguments in the previous section, we obtain a more general formula for all principal angles in the following.

THEOREM 4.3. *Let $S = (I - Q_F Q_F^T A)Q_G$. Singular values $\mu_1 \le \mu_2 \le \cdots \le \mu_q$ of matrix $A^{1/2}S$ are given by $\mu_k = \sqrt{1-\sigma_k^2}$, $k = 1, \ldots, q$, where σ_k are defined in* (4.4)*. Moreover, the principal angles satisfy the equalities $\theta_k = \arcsin(\mu_k)$. The right principal vectors can be computed as*

$$v_k = Q_G z_k, \qquad k = 1, \ldots, q,$$

where z_k are corresponding orthonormal right singular vectors of matrix $A^{1/2}S$. The left principal vectors are then computed by

$$u_k = Q_F Q_F^T A v_k / \sigma_k \quad \text{if } \sigma_k \neq 0, \quad k = 1, \ldots, q.$$

Proof. We first notice that squares of the singular values μ_k of the matrix $A^{1/2}S$, which appear in (4.6), coincide with eigenvalues $\nu_k = \mu_k^2$ of the product $S^T AS$. Using the fact that $Q_F^T A Q_F = I$ and $Q_G^T A Q_G = I$, we have

$$\begin{aligned} S^T AS &= Q_G^T (I - AQ_F Q_F^T) A (I - Q_F Q_F^T A) Q_G \\ &= I - Q_G^T A Q_F Q_F^T A Q_G. \end{aligned}$$

Utilizing the SVD (4.4), we obtain $Q_F^T A Q_G = Y \Sigma Z^T$, where $\Sigma = \operatorname{diag}(\sigma_1, \sigma_2, \ldots, \sigma_q)$; then

$$Z^T S^T ASZ = I - \Sigma^2 = \operatorname{diag}(1 - \sigma_1^2, 1 - \sigma_2^2, \ldots, 1 - \sigma_q^2).$$

Thus, the eigenvalues of $S^T AS$ are given by $\nu_k = 1 - \sigma_k^2$, $k = 1, \ldots, q$, and the formula for the principal angles follows directly from (4.3). □

For computational reasons, when n is large, we need to avoid dealing with the square root $A^{1/2}$ explicitly. Also, A may not be available as a matrix but only as a function performing the multiplication of A by a given vector. Fortunately, the previous theorem can be trivially reformulated as follows to resolve this issue.

THEOREM 4.4. *Eigenvalues $\nu_1 \le \nu_2 \le \cdots \le \nu_q$ of matrix $S^T AS$, where $S = (I - Q_F Q_F^T A)Q_G$, are equal to $\nu_k = 1 - \sigma_k^2$, $k = 1, \ldots, q$, where σ_k are defined in* (4.4)*. Moreover, the principal angles satisfy the equalities $\theta_k = \arcsin\left(\sqrt{\nu_k}\right)$, $k = 1, \ldots, q$. The right principal vectors can be computed as*

$$v_k = Q_G z_k, \qquad k = 1, \ldots, q,$$

where z_k are corresponding orthonormal right eigenvectors of matrix $S^T AS$. The left principal vectors are then computed by

$$u_k = Q_F Q_F^T A v_k / \sigma_k \quad \text{if } \sigma_k \neq 0, \quad k = 1, \ldots, q.$$

We can easily modify the previous proof to obtain the following analogue of Theorem 3.2.

THEOREM 4.5. *Singular values of the matrix $A^{1/2} Q_F Q_F^T A Q_G = A^{1/2} P_F Q_G$ coincide with σ_k, defined in* (4.4)*.*

It is also useful to represent principal angles using exclusively A-orthogonal projectors P_F and P_G on subspaces $\mathcal{F}$ and $\mathcal{G}$, respectively, similarly to Theorems 3.3 and 3.4.

THEOREM 4.6. *Under assumptions of Theorem* 4.1*, $\sigma_1 \ge \sigma_2 \ge \cdots \ge \sigma_q$ are the q largest singular values of the matrix $A^{1/2} P_F P_G A^{-1/2}$; in particular,*

$$\sigma_1 = \|P_F P_G\|_A.$$

Other $n-q$ singular values are all equal to zero.

Proof. First, we rewrite

$$\begin{aligned} A^{1/2}P_FP_GA^{-1/2} &= A^{1/2}Q_FQ_F^TAQ_GQ_G^TAA^{-1/2} \\ &= A^{1/2}Q_F\left(A^{1/2}Q_F\right)^T A^{1/2}Q_G\left(A^{1/2}Q_G\right)^T. \end{aligned}$$

As columns of matrices $A^{1/2}Q_F$ and $A^{1/2}Q_G$ are orthonormal with respect to the original scalar product $(\cdot,\cdot)$ bases of subspaces $A^{1/2}\mathcal{F}$ and $A^{1/2}\mathcal{G}$, respectively, the last product is equal to the product of orthogonal (not A-orthogonal!) projectors $P_{A^{1/2}\mathcal{F}}$ and $P_{A^{1/2}\mathcal{G}}$ on subspaces $A^{1/2}\mathcal{F}$ and $A^{1/2}\mathcal{G}$.

Second, we can now use Theorem 3.3 to state that cosine of principal angles between subspaces $A^{1/2}\mathcal{F}$ and $A^{1/2}\mathcal{G}$ with respect to the original scalar product $(\cdot,\cdot)$ are given by the q largest singular values of the product $P_{A^{1/2}\mathcal{F}}P_{A^{1/2}\mathcal{G}} = A^{1/2}P_FP_GA^{-1/2}$.

Finally, we use Theorem 4.2 to conclude that these singular values are, in fact, $\sigma_k,\ k=1,\ldots,q$, i.e., the cosine of principal angles between subspaces $\mathcal{F}$ and $\mathcal{G}$ with respect to the A-based scalar product. □

THEOREM 4.7. *Let assumptions of Theorem 4.4 be satisfied. Then $\mu_1 \le \mu_2 \le \cdots \le \mu_q$ are the q largest singular values of the matrix $A^{1/2}(I-P_F)P_GA^{-1/2}$; in particular,*

$$\mu_q = \|(I-P_F)P_G\|_A.$$

The other $n-q$ singular values are all equal to zero.

Proof. We rewrite

$$\begin{aligned} A^{1/2}(I-P_F)P_GA^{-1/2} &= \left(I - A^{1/2}Q_F\left(A^{1/2}Q_F\right)^T\right)A^{1/2}Q_G\left(A^{1/2}Q_G\right)^T \\ &= \left(I-P_{A^{1/2}\mathcal{F}}\right)P_{A^{1/2}\mathcal{G}} \end{aligned}$$

and then follow arguments similar to those of the previous proof, but now using Theorem 3.4 instead of Theorem 3.3. □

Remarks 3.6–3.7 and Theorems 3.5–3.6 for the case $A=I$ hold in the general case, too, with obvious modifications.

Our final theoretical results are perturbation theorems in the next section.

5. Perturbation of principal angles in the A-based scalar product.

In the present section, for simplicity, we *always assume* that matrices F, G and their perturbations $\tilde{F}$, $\tilde{G}$ have the same rank; thus, in particular, $p=q$.

We notice that F and G appear symmetrically in the definition of the principal angles, under our assumption that they and their perturbations have the same rank. This means that we do not have to analyze the perturbation of F and G together at the same time. Instead, we first study only a perturbation in G.

Before we start with an estimate for cosine, let us introduce a new notation $\ominus$ using an example:

$$(\mathcal{G}+\tilde{\mathcal{G}}) \ominus \mathcal{G} = (\mathcal{G}+\tilde{\mathcal{G}}) \cap \mathcal{G}^{\perp},$$

where $\ominus$ and the orthogonal complement to $\mathcal{G}$ are understood in the A-based scalar product.

Lemma 5.1. *Let $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_q$ and $\hat{\sigma}_1 \geq \hat{\sigma}_2 \geq \cdots \geq \hat{\sigma}_q$ be cosine of principal angles between subspaces $\mathcal{F}$, $\mathcal{G}$ and $\mathcal{F}$, $\tilde{\mathcal{G}}$, respectively, computed in the A-based scalar product. Then, for $k = 1, \ldots, q$,*

$$|\sigma_k - \hat{\sigma}_k| \leq \max\{\cos(\theta_{\min}\{(\mathcal{G}+\tilde{\mathcal{G}})\ominus\mathcal{G},\, \mathcal{F}\});\, \cos(\theta_{\min}\{(\mathcal{G}+\tilde{\mathcal{G}})\ominus\tilde{\mathcal{G}},\, \mathcal{F}\})\}\operatorname{gap}_A(\mathcal{G},\tilde{\mathcal{G}}), \tag{5.1}$$

where $\theta_{\min}$ is the smallest angle between corresponding subspaces, measured in the A-based scalar product.

Proof. The proof is based on the following identity:

$$A^{1/2}Q_FQ_F^TAQ_{\tilde{G}} = A^{1/2}Q_FQ_F^TAQ_GQ_G^TAQ_{\tilde{G}} + A^{1/2}Q_FQ_F^TA(I - Q_GQ_G^TA)Q_{\tilde{G}}, \tag{5.2}$$

which is a multidimensional analogue of the trigonometric formula for the cosine of the sum of two angles. Now we use two classical theorems on perturbation of singular values with respect to addition:

$$s_k(T+S) \leq s_k(T) + \|S\|, \tag{5.3}$$

and with respect to multiplication:

$$s_k(TS^T) \leq s_k(T)\|S^T\|, \tag{5.4}$$

where T and S are matrices of corresponding sizes. We first need to take $T = A^{1/2}Q_FQ_F^TAQ_GQ_G^TAQ_{\tilde{G}}$ and $S = A^{1/2}Q_FQ_F^TA(I - Q_GQ_G^TA)Q_{\tilde{G}}$ in (5.3) to get

$$\begin{aligned}\hat{\sigma}_k = s_k(A^{1/2}Q_FQ_F^TAQ_{\tilde{G}}) \leq{}& s_k(A^{1/2}Q_FQ_F^TAQ_GQ_G^TAQ_{\tilde{G}})\\ &+ \|A^{1/2}Q_FQ_F^TA(I - Q_GQ_G^TA)Q_{\tilde{G}}\|,\end{aligned}$$

where the first equality follows from Theorem 4.5. In the second term in the sum on the right, we need to estimate a product, similar to a product of *three* orthoprojectors. We notice that column vectors of $(I - Q_GQ_G^TA)Q_{\tilde{G}}$ belong to the subspace $(\mathcal{G}+\tilde{\mathcal{G}})\ominus\mathcal{G}$. Let $P_{(\mathcal{G}+\tilde{\mathcal{G}})\ominus\mathcal{G}}$ be an A-orthogonal projector on the subspace. Then the second term can be rewritten, also using the projector $Q_FQ_F^TA = P_F$, as

$$\begin{aligned}A^{1/2}Q_FQ_F^TA(I - Q_GQ_G^TA)Q_{\tilde{G}} &= A^{1/2}Q_FQ_F^TAP_{(\mathcal{G}+\tilde{\mathcal{G}})\ominus\mathcal{G}}(I - Q_GQ_G^TA)Q_{\tilde{G}}\\ &= \left(A^{1/2}P_FP_{(\mathcal{G}+\tilde{\mathcal{G}})\ominus\mathcal{G}}A^{-1/2}\right)A^{1/2}(I - Q_GQ_G^TA)Q_{\tilde{G}};\end{aligned}$$

therefore, it can be estimated as

$$\|A^{1/2}Q_FQ_F^TA(I-Q_GQ_G^TA)Q_{\tilde{G}}\| \leq \|A^{1/2}P_FP_{(\mathcal{G}+\tilde{\mathcal{G}})\ominus\mathcal{G}}A^{-1/2}\|\|A^{1/2}(I-Q_GQ_G^TA)Q_{\tilde{G}}\|.$$

The first multiplier in the last product equals

$$\|A^{1/2}P_FP_{(\mathcal{G}+\tilde{\mathcal{G}})\ominus\mathcal{G}}A^{-1/2}\| = \|P_FP_{(\mathcal{G}+\tilde{\mathcal{G}})\ominus\mathcal{G}}\|_A = \cos(\theta_{\min}\{(\mathcal{G}+\tilde{\mathcal{G}})\ominus\mathcal{G},\, \mathcal{F}\}),$$

similar to (4.6) and using Theorem 4.6 for subspaces $(\mathcal{G}+\tilde{\mathcal{G}})\ominus\mathcal{G}$ and $\mathcal{F}$; while the second multiplier is $\operatorname{gap}_A(\mathcal{G},\tilde{\mathcal{G}})$, because of our assumption $\dim\mathcal{F} = \dim\mathcal{G} = \dim\tilde{\mathcal{G}}$.

To estimate the first term in the sum, we apply (5.4) with $T = A^{1/2}Q_F Q_F^T A Q_G$ and $S^T = Q_G^T A Q_{\tilde{G}}$:

$$\begin{aligned} s_k(A^{1/2}Q_F Q_F^T A Q_G Q_G^T A Q_{\tilde{G}}) &\leq s_k(A^{1/2}Q_F Q_F^T A Q_G)\|Q_G^T A Q_{\tilde{G}}\| \\ &\leq s_k(A^{1/2}Q_F Q_F^T A Q_G) = \sigma_k, \end{aligned}$$

simply because the second multiplier here is the cosine of an angle between $\mathcal{G}$ and $\tilde{\mathcal{G}}$ in the A-based scalar product, which is, of course, bounded by one from above. Thus, we just proved

$$\hat{\sigma}_k \leq \sigma_k + \cos(\theta_{\min}\{(\mathcal{G} + \tilde{\mathcal{G}}) \ominus \mathcal{G}, \mathcal{F}\})\mathrm{gap}_A(\mathcal{G}, \tilde{\mathcal{G}}).$$

Changing places of $Q_{\tilde{G}}$ and Q_G, we obtain

$$\sigma_k \leq \hat{\sigma}_k + \cos(\theta_{\min}\{(\mathcal{G} + \tilde{\mathcal{G}}) \ominus \tilde{\mathcal{G}}, \mathcal{F}\})\mathrm{gap}_A(\mathcal{G}, \tilde{\mathcal{G}})$$

and come to the statement of the lemma. □

REMARK 5.1. *Let us try to clarify the meaning of constants appearing in the statement of Lemma* 5.1. *Let us consider, e.g.,* $\cos(\theta_{\min}\{(\mathcal{G} + \tilde{\mathcal{G}}) \ominus \mathcal{G}, \mathcal{F}\})$. *The cosine takes its maximal value, one, when at least one direction of the perturbation of* $\mathcal{G}$ *is A-orthogonal to* $\mathcal{G}$ *and parallel to* $\mathcal{F}$ *at the same time. It is small, on the contrary, when a part of the perturbation, A-orthogonal to* $\mathcal{G}$*, is also A-orthogonal to* $\mathcal{F}$*. As* $(\mathcal{G} + \tilde{\mathcal{G}}) \ominus \mathcal{G} \subseteq \mathcal{G}^\perp$*, we have*

$$\cos(\theta_{\min}\{(\mathcal{G} + \tilde{\mathcal{G}}) \ominus \mathcal{G}, \mathcal{F}\}) \leq \cos\left(\theta_{\min}\{\mathcal{G}^\perp, \mathcal{F}\}\right) = \sin\left(\theta_{\max}\{\mathcal{G}, \mathcal{F}\}\right) = \mathrm{gap}_A(\mathcal{G}, \mathcal{F}),$$

which is the constant of the asymptotic perturbation estimate of [3] *(where* $A = I$*). The latter constant is small if subspaces* $\mathcal{G}$ *and* $\mathcal{F}$ *are close to each other, which can be considered more as a cancellation prize as in this case cosine of all principal angles is almost one, and a perturbation estimate for the cosine does not help much because of the cancellation effect.*

REMARK 5.2. *A natural approach similar to that of* [15] *with* $A = I$ *involves a simpler identity:*

$$Q_F^T A Q_{\tilde{G}} = Q_F^T A Q_G + Q_F^T A(Q_{\tilde{G}} - Q_G),$$

where a norm of the second term is then estimated. Then (5.3) *gives an estimate of singular values using* $\|A^{1/2}(Q_{\tilde{G}} - Q_G)\|$*. As singular values are invariant with respect to particular choices of matrices* $Q_{\tilde{G}}$ *and* Q_G *with A-orthonormal columns, as far as they provide ranges* $\tilde{\mathcal{G}}$ *and* $\mathcal{G}$*, respectively, we can choose them to minimize the norm of the difference, which gives*

$$\inf_Q \|A^{1/2}(Q_G - Q_{\tilde{G}}Q)\|, \tag{5.5}$$

where Q is an arbitrary q-by-q orthogonal matrix. This quantity appears in [15] *with* $A = I$ *as a special type of the Procrustes problem. In* [15]*, it is estimated in terms of the gap between subspaces* $\tilde{\mathcal{G}}$ *and* $\mathcal{G}$ *(using an extra assumption that* $2q \leq n$*). Repeating similar arguments, we derive*

$$|\sigma_k - \hat{\sigma}_k| \leq \inf_Q \|A^{1/2}(Q_G - Q_{\tilde{G}}Q)\|_A \leq \sqrt{2}\,\mathrm{gap}_A(\mathcal{G}, \tilde{\mathcal{G}}), \qquad k = 1, \ldots, q. \tag{5.6}$$

Lemma 5.1 *furnishes estimates of the perturbation of singular values in terms of the gap directly, which gives a much better constant, consistent with that of the asymptotic estimate of* [3] *for $A = I$; see the previous remark.*

Now we prove a separate estimate for sine.

LEMMA 5.2. *Let $\mu_1 \leq \mu_2 \leq \cdots \leq \mu_q$ and $\hat{\mu}_1 \leq \hat{\mu}_2 \leq \cdots \leq \hat{\mu}_q$ be sine of principal angles between subspaces $\mathcal{F}$, $\mathcal{G}$, and $\mathcal{F}$, $\tilde{\mathcal{G}}$, respectively, computed in the A-based scalar product. Then, for $k = 1, \ldots, q$,*

$$|\mu_k - \hat{\mu}_k| \leq \max\{\sin(\theta_{\max}\{(\mathcal{G}+\tilde{\mathcal{G}}) \ominus \mathcal{G},\, \mathcal{F}\});\, \sin(\theta_{\max}\{(\mathcal{G}+\tilde{\mathcal{G}}) \ominus \tilde{\mathcal{G}},\, \mathcal{F}\})\}\mathrm{gap}_A(\mathcal{G}, \tilde{\mathcal{G}}), \tag{5.7}$$

where $\theta_{\max}$ is the largest angle between corresponding subspaces, measured in the A-based scalar product.

Proof. The proof is based on the following identity:

$$\begin{aligned} A^{1/2}(I - Q_F Q_F^T A)Q_{\tilde{G}} = {} & A^{1/2}(I - Q_F Q_F^T A)Q_G Q_G^T A Q_{\tilde{G}} \\ & + A^{1/2}(I - Q_F Q_F^T A)(I - Q_G Q_G^T A)Q_{\tilde{G}}, \end{aligned}$$

which is a multidimensional analogue of the trigonometric formula for the sine of the sum of two angles. The rest of the proof is similar to that of Lemma 5.1.

We first need to take $T = A^{1/2}(I - Q_F Q_F^T A)Q_G Q_G^T A Q_{\tilde{G}}$ and $S = A^{1/2}(I - Q_F Q_F^T A)(I - Q_G Q_G^T A)Q_{\tilde{G}}$ and use (5.3) to get

$$\begin{aligned} s_k(A^{1/2}(I - Q_F Q_F^T A)Q_{\tilde{G}}) \leq {} & s_k(A^{1/2}(I - Q_F Q_F^T A)Q_G Q_G^T A Q_{\tilde{G}}) \\ & + \|A^{1/2}(I - Q_F Q_F^T A)(I - Q_G Q_G^T A)Q_{\tilde{G}}\|. \end{aligned}$$

In the second term in the sum on the right, $Q_F Q_F^T A = P_F$ and we deduce

$$\begin{aligned} A^{1/2}(I - Q_F Q_F^T A)(I - Q_G Q_G^T A)Q_{\tilde{G}} &= A^{1/2}(I - P_F)P_{(\mathcal{G}+\tilde{\mathcal{G}})\ominus\mathcal{G}}(I - Q_G Q_G^T A)Q_{\tilde{G}} \\ &= \left(A^{1/2}(I - P_F)P_{(\mathcal{G}+\tilde{\mathcal{G}})\ominus\mathcal{G}}A^{-1/2}\right)\left(A^{1/2}(I - Q_G Q_G^T A)Q_{\tilde{G}}\right), \end{aligned}$$

using notation $P_{(\mathcal{G}+\tilde{\mathcal{G}})\ominus\mathcal{G}}$ for the A-orthogonal projector on the subspace $(\mathcal{G}+\tilde{\mathcal{G}}) \ominus \mathcal{G}$, introduced in the proof of Lemma 5.1. Therefore, the second term can be estimated as

$$\begin{aligned} & \|A^{1/2}(I - Q_F Q_F^T A)(I - Q_G Q_G^T A)Q_{\tilde{G}}\| \\ & \qquad \leq \|A^{1/2}(I - P_F)P_{(\mathcal{G}+\tilde{\mathcal{G}})\ominus\mathcal{G}}A^{-1/2}\|\|A^{1/2}(I - Q_G Q_G^T A)Q_{\tilde{G}}\|. \end{aligned}$$

The first multiplier is

$$\|A^{1/2}(I - P_F)P_{(\mathcal{G}+\tilde{\mathcal{G}})\ominus\mathcal{G}}A^{-1/2}\| = \|(I - P_F)P_{(\mathcal{G}+\tilde{\mathcal{G}})\ominus\mathcal{G}}\|_A = \sin(\theta_{\max}\{(\mathcal{G}+\tilde{\mathcal{G}}) \ominus \mathcal{G},\, \mathcal{F}\})$$

by Theorem 4.7 as $\dim\mathcal{F} \geq \dim((\mathcal{G}+\tilde{\mathcal{G}}) \ominus \mathcal{G})$, while the second multiplier is simply $\mathrm{gap}_A(\mathcal{G}, \tilde{\mathcal{G}})$ because of our assumption $\dim\mathcal{G} = \dim\tilde{\mathcal{G}}$.

To estimate the first term in the sum, we take with $T = A^{1/2}(I - Q_F Q_F^T A)Q_G$ and $S^T = Q_G^T A Q_{\tilde{G}}$ and apply (5.4):

$$\begin{aligned} s_k(A^{1/2}(I - Q_F Q_F^T A)Q_G Q_G^T A Q_{\tilde{G}}) &\leq s_k(A^{1/2}(I - Q_F Q_F^T A)Q_G)\|Q_G^T A Q_{\tilde{G}}\| \\ &\leq s_k(A^{1/2}(I - Q_F Q_F^T A)Q_G), \end{aligned}$$

using exactly the same arguments as in the proof of Lemma 5.1.

Thus, we have proved

$$\hat{\mu}_k \leq \mu_k + \sin(\theta_{\max}\{(\mathcal{G}+\tilde{\mathcal{G}}) \ominus \mathcal{G},\, \mathcal{F}\})\mathrm{gap}_A(\mathcal{G},\tilde{\mathcal{G}}).$$

Changing the places of $Q_{\tilde{G}}$ and Q_G, we get

$$\mu_k \leq \hat{\mu}_k + \sin(\theta_{\max}\{(\mathcal{G}+\tilde{\mathcal{G}}) \ominus \tilde{\mathcal{G}},\, \mathcal{F}\})\mathrm{gap}_A(\mathcal{G},\tilde{\mathcal{G}}).$$

The statement of the lemma follows. □

REMARK 5.3. *Let us also highlight that simpler estimates,*

$$|\mu_k - \hat{\mu}_k| \leq \mathrm{gap}_A(\mathcal{G},\tilde{\mathcal{G}}), \quad |\sigma_k - \hat{\sigma}_k| \leq \mathrm{gap}_A(\mathcal{G},\tilde{\mathcal{G}}), \quad k = 1,\dots,q,$$

which are not as sharp as those we prove in Lemmas 5.1 *and* 5.2*, can be derived almost trivially using orthoprojectors (see* [29, 27, 14]*), where this approach is used for the case $A = I$. Indeed, we start with identities*

$$A^{1/2}P_FP_{\tilde{G}}A^{-1/2} = A^{1/2}P_FP_GA^{-1/2} + \left(A^{1/2}P_FA^{-1/2}\right)\left(A^{1/2}(P_{\tilde{G}} - P_G)A^{-1/2}\right)$$

for the cosine and

$$A^{1/2}(I-P_F)P_{\tilde{G}}A^{-1/2} = A^{1/2}(I-P_F)P_GA^{-1/2} + \left(A^{1/2}(I-P_F)A^{-1/2}\right)\left(A^{1/2}(P_{\tilde{G}} - P_G)A^{-1/2}\right)$$

for the sine, and use (5.3) *and Theorems* 4.6 *and* 4.7*. A norm of the second term is then estimated from above by* $\mathrm{gap}_A(\mathcal{G},\tilde{\mathcal{G}})$*, using the fact that for an A-orthoprojector P_F we have $\|P_F\|_A = \|I - P_F\|_A = 1$.*

Instead of the latter, we can use a bit more sophisticated approach, as in [14]*, if we introduce the A-orthogonal projector $P_{\mathcal{G}+\tilde{\mathcal{G}}}$ on the subspace $\mathcal{G}+\tilde{\mathcal{G}}$. Then the norm of second term is bounded by* $\mathrm{gap}_A(\mathcal{G},\tilde{\mathcal{G}})$ *times $\|P_FP_{\mathcal{G}+\tilde{\mathcal{G}}}\|_A$ for the cosine and times $\|(I-P_F)P_{\mathcal{G}+\tilde{\mathcal{G}}}\|_A$ for the sine, where we can now use Theorem* 4.6 *to provide a geometric interpretation of these two constants. This leads to estimates similar to those of* [14] *for $A = I$:*

$$|\sigma_k - \hat{\sigma}_k| \leq \cos(\theta_{\min}\{\mathcal{F},\, \mathcal{G}+\tilde{\mathcal{G}}\})\mathrm{gap}_A(\mathcal{G},\tilde{\mathcal{G}}), \qquad k = 1,\dots,q,$$

and

$$|\mu_k - \hat{\mu}_k| \leq \cos(\theta_{\min}\{\mathcal{F}^{\perp},\, \mathcal{G}+\tilde{\mathcal{G}}\})\mathrm{gap}_A(\mathcal{G},\tilde{\mathcal{G}}), \qquad k = 1,\dots,q.$$

However, the apparent constant "improvement" in the second estimate, for the sine, is truly misleading as

$$\cos(\theta_{\min}\{\mathcal{F}^{\perp},\, \mathcal{G}+\tilde{\mathcal{G}}\}) = 1$$

simply because $\dim\mathcal{F} < \dim(\mathcal{G}+\tilde{\mathcal{G}})$ *in all cases except for the trivial possibility $\mathcal{G} = \tilde{\mathcal{G}}$, so subspaces $\mathcal{F}^{\perp}$ and $\mathcal{G}+\tilde{\mathcal{G}}$ must have a nontrivial intersection.*

The first estimate, for the cosine, does give a better constant (compare to one), but our constant is sharper; e.g.,

$$\cos(\theta_{\min}\{\mathcal{F},\, (\mathcal{G}+\tilde{\mathcal{G}}) \ominus G\}) \leq \cos(\theta_{\min}\{\mathcal{F},\, \mathcal{G}+\tilde{\mathcal{G}}\}).$$

Our more complex identities used to derive perturbation bounds provide an extra projector in the error term, which allows us to obtain better constants.

We can now establish an estimate of absolute sensitivity of cosine and sine of principal angles with respect to absolute *perturbations of subspaces.*

THEOREM 5.3. *Let $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_q$ and $\tilde{\sigma}_1 \geq \tilde{\sigma}_2 \geq \cdots \geq \tilde{\sigma}_q$ be cosine of principal angles between subspaces $\mathcal{F}$, $\mathcal{G}$, and $\tilde{\mathcal{F}}$, $\tilde{\mathcal{G}}$, respectively, computed in the A-based scalar product. Then*

$$|\sigma_k - \tilde{\sigma}_k| \leq c_1 \mathrm{gap}_A(\mathcal{F}, \tilde{\mathcal{F}}) + c_2 \mathrm{gap}_A(\mathcal{G}, \tilde{\mathcal{G}}), \qquad k = 1, \ldots, q, \tag{5.8}$$

where

$$c_1 = \max\{\cos(\theta_{\min}\{(\mathcal{G}+\tilde{\mathcal{G}}) \ominus \mathcal{G},\, \mathcal{F}\}); \cos(\theta_{\min}\{(\mathcal{G}+\tilde{\mathcal{G}}) \ominus \tilde{\mathcal{G}},\, \mathcal{F}\})\},$$

$$c_2 = \max\{\cos(\theta_{\min}\{(\mathcal{F}+\tilde{\mathcal{F}}) \ominus \mathcal{F},\, \tilde{\mathcal{G}}\}); \cos(\theta_{\min}\{(\mathcal{F}+\tilde{\mathcal{F}}) \ominus \tilde{\mathcal{F}},\, \tilde{\mathcal{G}}\})\},$$

where $\theta_{\min}$ is the smallest angle between corresponding subspaces in the A-based scalar product.

Proof. First, by Lemma 5.1, for $k = 1, \ldots, q$,

$$|\sigma_k - \hat{\sigma}_k| \leq \max\{\cos(\theta_{\min}\{(\mathcal{G}+\tilde{\mathcal{G}}) \ominus \mathcal{G},\, \mathcal{F}\}); \cos(\theta_{\min}\{(\mathcal{G}+\tilde{\mathcal{G}}) \ominus \tilde{\mathcal{G}},\, \mathcal{F}\})\}\mathrm{gap}_A(\mathcal{G}, \tilde{\mathcal{G}}).$$

Second, we apply a similar statement to cosine of principal angles between subspaces $\mathcal{F}$, $\tilde{\mathcal{G}}$ and $\tilde{\mathcal{F}}$, $\tilde{\mathcal{G}}$, respectively, computed in the A-based scalar product:

$$|\tilde{\sigma}_k - \hat{\sigma}_k| \leq \max\{\cos(\theta_{\min}\{(\mathcal{F}+\tilde{\mathcal{F}}) \ominus \mathcal{F},\, \tilde{\mathcal{G}}\}); \cos(\theta_{\min}\{(\mathcal{F}+\tilde{\mathcal{F}}) \ominus \tilde{\mathcal{F}},\, \tilde{\mathcal{G}}\})\}\mathrm{gap}_A(\mathcal{F}, \tilde{\mathcal{F}}).$$

The statement of the theorem now follows from the triangle inequality. □

THEOREM 5.4. *Let $\mu_1 \leq \mu_2 \leq \cdots \leq \mu_q$ and $\tilde{\mu}_1 \leq \tilde{\mu}_2 \leq \cdots \leq \tilde{\mu}_q$ be sine of principal angles between subspaces $\mathcal{F}$, $\mathcal{G}$, and $\tilde{\mathcal{F}}$, $\tilde{\mathcal{G}}$, respectively, computed in the A-based scalar product. Then*

$$|\mu_k - \tilde{\mu}_k| \leq c_3 \mathrm{gap}_A(\mathcal{F}, \tilde{\mathcal{F}}) + c_4 \mathrm{gap}_A(\mathcal{G}, \tilde{\mathcal{G}}), \qquad k = 1, \ldots, q, \tag{5.9}$$

where

$$c_3 = \max\{\sin(\theta_{\max}\{(\mathcal{G}+\tilde{\mathcal{G}}) \ominus \mathcal{G},\, \mathcal{F}\}); \sin(\theta_{\max}\{(\mathcal{G}+\tilde{\mathcal{G}}) \ominus \tilde{\mathcal{G}},\, \mathcal{F}\})\},$$

$$c_4 = \max\{\sin(\theta_{\max}\{(\mathcal{F}+\tilde{\mathcal{F}}) \ominus \mathcal{F},\, \tilde{\mathcal{G}}\}); \sin(\theta_{\max}\{(\mathcal{F}+\tilde{\mathcal{F}}) \ominus \tilde{\mathcal{F}},\, \tilde{\mathcal{G}}\})\},$$

where $\theta_{\max}$ is the largest angle between corresponding subspaces in the A-based scalar product.

Proof. First, by Lemma 5.2, for $k = 1, \ldots, q$,

$$|\mu_k - \hat{\mu}_k| \leq \max\{\sin(\theta_{\max}\{(\mathcal{G}+\tilde{\mathcal{G}}) \ominus \mathcal{G},\, \mathcal{F}\}); \sin(\theta_{\max}\{(\mathcal{G}+\tilde{\mathcal{G}}) \ominus \tilde{\mathcal{G}},\, \mathcal{F}\})\}\mathrm{gap}_A(\mathcal{G}, \tilde{\mathcal{G}}).$$

Second, we apply a similar statement to sine of principal angles between subspaces $\mathcal{F}$, $\tilde{\mathcal{G}}$ and $\tilde{\mathcal{F}}$, $\tilde{\mathcal{G}}$, respectively, computed in the A-based scalar product:

$$|\tilde{\mu}_k - \hat{\mu}_k| \leq \max\{\sin(\theta_{\max}\{(\mathcal{F}+\tilde{\mathcal{F}}) \ominus \mathcal{F},\, \tilde{\mathcal{G}}\}); \sin(\theta_{\max}\{(\mathcal{F}+\tilde{\mathcal{F}}) \ominus \tilde{\mathcal{F}},\, \tilde{\mathcal{G}}\})\}\mathrm{gap}_A(\mathcal{F}, \tilde{\mathcal{F}}).$$

The statement of the theorem now follows from the triangle inequality. □

Finally, we want a perturbation analysis in terms of matrices F and G that generate subspaces $\mathcal{F}$ and $\mathcal{G}$. For that, we have to estimate the sensitivity of a column-space of a matrix, for example, matrix G.

LEMMA 5.5. *Let*

$$\kappa_A(G) = \frac{s_{\max}(A^{1/2}G)}{s_{\min}(A^{1/2}G)}$$

denote the corresponding A-based condition number of G, where $s_{\max}$ and $s_{\min}$ are, respectively, largest and smallest singular values of the matrix $A^{1/2}G$. Let $\mathcal{G}$ and $\tilde{\mathcal{G}}$ be column-spaces of matrices G and $\tilde{G}$, respectively. Then

$$\operatorname{gap}_A(\mathcal{G}, \tilde{\mathcal{G}}) \le \kappa_A(G) \frac{\|A^{1/2}(G - \tilde{G})\|}{\|A^{1/2}G\|}. \tag{5.10}$$

Proof. Here, we essentially just adopt the corresponding proof of [29] for the A-based scalar product using the same approach as in Theorem 4.2.

Let us consider the polar decompositions

$$A^{1/2}G = A^{1/2}Q_G T_G \text{ and } A^{1/2}\tilde{G} = A^{1/2}Q_{\tilde{G}} T_{\tilde{G}},$$

where matrices $A^{1/2}Q_G$ and $A^{1/2}Q_{\tilde{G}}$ have orthonormal columns and matrices T_G and $T_{\tilde{G}}$ are q-by-q symmetric positive definite; e.g., $T_G = (Q_G Q_G^T)^{1/2}$. Singular values of T_G and $T_{\tilde{G}}$ are, therefore, the same as singular values of $A^{1/2}G$ and $A^{1/2}\tilde{G}$, respectively. Then,

$$(I - P_{\tilde{G}})(G - \tilde{G}) = (I - P_{\tilde{G}})Q_G T_G.$$

Therefore,

$$A^{1/2}(I - P_{\tilde{G}})Q_G = \left(A^{1/2}(I - P_{\tilde{G}})A^{-1/2}\right) A^{1/2}(G - \tilde{G})T_G^{-1},$$

and

$$\operatorname{gap}_A(\mathcal{G}, \tilde{\mathcal{G}}) \le \|A^{1/2}(G - \tilde{G})\| \|T_G^{-1}\| = \frac{\|A^{1/2}(G - \tilde{G})\|}{s_{\min}(A^{1/2}G)},$$

as $\|A^{1/2}(I - P_{\tilde{G}})A^{-1/2}\| = \|I - P_{\tilde{G}}\|_A \le 1$. The statement of the lemma follows. □

REMARK 5.4. *Some matrices allow improvement of their condition numbers by column scaling, which trivially does not change the column range. Our simple Lemma* 5.5 *does not capture this property. A more sophisticated variant can be easily obtained using a technique developed in* [15, 14].

Our cosine theorem follows next. It generalizes results of [27, 14, 15] to A-based scalar products and somewhat improves the constant.

THEOREM 5.6. *Under assumptions of Theorem* 5.3,

$$|\sigma_k - \tilde{\sigma}_k| \le c_1 \kappa_A(F) \frac{\|A^{1/2}(F - \tilde{F})\|}{\|A^{1/2}F\|} + c_2 \kappa_A(G) \frac{\|A^{1/2}(G - \tilde{G})\|}{\|A^{1/2}G\|}, \qquad k = 1, \ldots, q. \tag{5.11}$$

The theorem above does not provide an accurate estimate for small angles. To fill the gap, we suggest the following perturbation theorem in terms of sine of principal angles; cf. [27, 14] for $A = I$.

Theorem 5.7. *Under assumptions of Theorem* 5.4,

$$|\mu - \tilde{\mu}_k| \leq c_3 \kappa_A(F) \frac{\|A^{1/2}(F - \tilde{F})\|}{\|A^{1/2}F\|} + c_4 \kappa_A(G) \frac{\|A^{1/2}(G - \tilde{G})\|}{\|A^{1/2}G\|}, \qquad k = 1, \dots, q. \tag{5.12}$$

Finally, let us underline that all sensitivity results of the present paper are for *absolute* errors. Golub and Zha in [14] observe that relative errors of sine and cosine of principal angles are not, in general, bounded by the perturbation.

We consider algorithms of computing principal angles with respect to an A-based scalar product in the next section.

6. Algorithm implementation. In this section, we provide a detailed description of our code SUBSPACEA.m and discuss the algorithm implementation.

Theorem 4.4 is our main theoretical foundation for a sine-based algorithm for computing principal angles with respect to an A-based scalar product. However, a naive implementation, using the SVD of the matrix S^TAS, may not produce small angles accurately in computer arithmetic. We now try to explain informally this fact, which is actually observed in numerical tests.

Let, for simplicity of notation, all principal angles be small.

Let us consider a particular case, where columns of matrices Q_F and Q_G are already principal vectors in exact arithmetic. In reality, this is the situation we will face in Algorithm 6.2. Then, in exact arithmetic, columns of S are A-orthogonal and their A-norms are exactly the sine of principal angles. Thus, if there are several small angles different in orders of magnitude, the columns of S are badly scaled. When we take the norms squared, by explicitly computing the product S^TAS, we make the scaling even worse, as the diagonal entries of this diagonal matrix are now the sine of the principal angles squared, in exact arithmetic. In the presence of round-off errors, the matrix S^TAS is usually not diagonal; thus, principal angles smaller than 10^{-8} will lead to an underflow effect in double precision, which cannot be cured by taking square roots of its singular values later in the algorithm.

To resolve this, we want to be able to compute the SVD of the matrix S^TAS without using either S^TAS itself or $A^{1/2}S$. One possibility is suggested in the following lemma.

Lemma 6.1. *The SVD of the matrix $A^{1/2}S$ coincides with the SVD of the matrix Q_S^TAS, where Q_S is a matrix with A-orthonormal columns, which span the same column-space as columns of matrix S.*

Proof. We have

$$(Q_S^TAS)^TQ_S^TAS = S^TAQ_SQ_S^TAS = S^TAP_SS = S^TAS,$$

where $P_S = Q_SQ_S^TA$ is the A-orthogonal projector on the column-space of Q_S, which is the same, by definition, as the column-space of S, so that $P_SS = S$. □

This contributes to the accuracy of our next Algorithm 6.1, based on Lemma 6.1, to be more reliable in the presence of round-off errors, when several principal angles are small.

By analogy with Algorithms 3.1 and 3.2, we can remove the restriction that matrix Σ is invertible and somewhat improve the costs in Algorithm 6.1 by reducing the size of the matrix S in step 4, which leads to Algorithm 6.2.

```
ALGORITHM 6.1: SUBSPACEA.m.
Input: real matrices $F$ and $G$ with the same number of rows, and a symmetric positive
    definite matrix $A$ for the scalar product, or a device to compute $Ax$ for a given
    vector $x$.
 1. Compute $A$-orthonormal bases $Q_F = \mathrm{ortha}(F)$, $Q_G = \mathrm{ortha}(G)$ of column-spaces of
    $F$ and $G$.
 2. Compute SVD for cosine $[Y, \Sigma, Z] = \mathrm{svd}(Q_F^T \, A \, Q_G)$, $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_q)$.
 3. Compute matrices of left $U_{\cos} = Q_F Y$ and right $V_{\cos} = Q_G Z$ principal vectors.
 4. Compute the matrix $S = \begin{cases} Q_G - Q_F(Q_F^T \, A \, Q_G) & \text{if } \mathrm{rank}(Q_F) \geq \mathrm{rank}(Q_G), \\ Q_F - Q_G(Q_G^T \, A \, Q_F) & \text{otherwise.} \end{cases}$
 5. Compute $A$-orthonormal basis $Q_S = \mathrm{ortha}(S)$ of the column-space of $S$.
 6. Compute SVD for sine: $[Y, \mathrm{diag}(\mu_1, \ldots, \mu_q), Z] = \mathrm{svd}(Q_S^T A S)$.
 7. Compute matrices $U_{\sin}$ and $V_{\sin}$ of left and right principal vectors:
        $V_{\sin} = Q_G Z$, $U_{\sin} = Q_F(Q_F^T A V_{\sin})\Sigma^{-1}$   if $\mathrm{rank}(Q_F) \geq \mathrm{rank}(Q_G)$;
        $U_{\sin} = Q_F Z$, $V_{\sin} = Q_G(Q_G^T A U_{\sin})\Sigma^{-1}$   otherwise.
 8. Compute the principal angles, for $k = 1, \ldots, q$:
        $\theta_k = \begin{cases} \arccos(\sigma_k) & \text{if } \sigma_k^2 < 1/2, \\ \arcsin(\mu_k) & \text{if } \mu_k^2 \leq 1/2. \end{cases}$
 9. Form matrices $U$ and $V$ by picking up corresponding columns of $U_{\sin}$, $V_{\sin}$ and
    $U_{\cos}$, $V_{\cos}$, according to the choice for $\theta_k$ above.
Output: Principal angles $\theta_1, \ldots, \theta_q$ between column-spaces of matrices $F$ and $G$ in the
    $A$-based scalar product, and corresponding matrices of left, $U$, and right, $V$, principal
    vectors.
```

```
ALGORITHM 6.2: Improved SUBSPACEA.m.
Input: real matrices $F$ and $G$ with the same number of rows, and a symmetric positive
    definite matrix $A$ for the scalar product, or a device to compute $Ax$ for a given
    vector $x$.
 1. Compute $A$-orthogonal bases $Q_F = \mathrm{ortha}(F)$, $Q_G = \mathrm{ortha}(G)$ of column-spaces of
    $F$ and $G$.
 2. Compute SVD for cosine $[Y, \Sigma, Z] = \mathrm{svd}(Q_F^T \, A \, Q_G)$, $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_q)$.
 3. Compute matrices of left $U_{\cos} = Q_F Y$ and right $V_{\cos} = Q_G Z$ principal vectors.
 4. Compute large principal angles for $k = 1, \ldots, q$:
        $\theta_k = \arccos(\sigma_k)$ if $\sigma_k^2 < 1/2$.
 5. Form parts of matrices $U$ and $V$ by picking up corresponding columns of $U_{\cos}$, $V_{\cos}$,
    according to the choice for $\theta_k$ above. Put columns of $U_{\cos}$, $V_{\cos}$, which are left,
    in matrices $R_F$ and $R_G$. Collect the corresponding $\sigma$'s in a diagonal matrix $\Sigma_R$.
 6. Compute the matrix $S = R_G - Q_F(Q_F^T A R_G)$.
 7. Compute $A$-orthogonal basis $Q_S = \mathrm{ortha}(S)$ of the column-space of $S$.
 8. Compute SVD for sine: $[Y, \mathrm{diag}(\mu_1, \ldots, \mu_q), Z] = \mathrm{svd}(Q_S^T A S)$.
 9. Compute matrices $U_{\sin}$ and $V_{\sin}$ of left and right principal vectors:
        $V_{\sin} = R_G Z$, $U_{\sin} = R_F(R_F^T A V_{\sin})\Sigma_R^{-1}$.
10. Compute the missing principal angles, for $k = 1, \ldots, q$:
        $\theta_k = \arcsin(\mu_k)$ if $\mu_k^2 \leq 1/2$.
11. Complete matrices $U$ and $V$ by adding columns of $U_{\sin}$, $V_{\sin}$.
Output: Principal angles $\theta_1, \ldots, \theta_q$ between column-spaces of matrices $F$ and $G$, and
    corresponding matrices $U$ and $V$ of left and right principal vectors, respectively.
```

Previous remarks of section 3 for the algorithms with $A = I$ are applicable to the present algorithms with self-evident changes. A few additional A-specific remarks follow.

REMARK 6.1. *In step* 1 *of Algorithms* 6.1 *and* 6.2*, we use our SVD-based function ORTHA.m for the A-orthogonalization. Specifically, computing an A-orthogonal basis Q of the column-space of a matrix X is done in three steps in ORTHA.m. First, we orthonormalize X, using SVD-based built-in MATLAB code ORTH.m with a preceding explicit column scaling; see Remark* 3.1 *on whether the scaling is actually needed. Second, we compute $[U, S, V] = \mathrm{svd}(X^T AX)$, using MATLAB's built-in SVD code. Finally, we take $Q = XUS^{-1/2}$. If A is ill-conditioned, an extra cycle may be performed to improve the accuracy. While formal stability and accuracy analysis of this method is yet to be done, ORTHA.m demonstrates practical robustness in our numerical tests. A detailed description and investigation of the algorithm used in ORTHA.m will be reported elsewhere.*

REMARK 6.2. *We note that, when $n \gg p$, the computational costs of SVDs of p-by-p matrices are negligible; it is multiplication by A, which may be very computationally expensive. Therefore, we want to minimize the number of multiplications by A. In the present version* 4.0 *of our code SUBSPACEA.m, based on Algorithm* 6.2*, we multiply matrix A by a vector $2p+q$ times in the worst-case scenario of all angles being small, in steps* 1 *and* 7*. We can avoid multiplying by A on steps* 2*,* 8*, and* 9 *by using appropriate linear combinations of earlier computed vectors instead.*

REMARK 6.3. *Our actual code is written for a more general complex case, where we require matrix A to be Hermitian.*

Let us finally underline that in a situation when a matrix K from the factorization $A = K^T K$ is given rather than the matrix A itself, we do not advise using Algorithms 6.1 and 6.2. Instead, we recommend multiplying matrices F and G by K on the right and using simpler Algorithms 3.1 and 3.2, according to Theorem 4.2.

7. Numerical examples. Numerical tests in the present section were performed using version 4.0 of our SUBSPACEA.m code, based on Algorithms 3.2 and 6.2. Numerical results presented were obtained, unless indicated otherwise, on Red Hat 6.1 LINUX Dual Pentium-III 500, running MATLAB release 12, version 6.0.0.18. Tests were also made on Compaq Dual Alpha server DS 20, running MATLAB release 12, version 6.0.0.18 and on several Microsoft Windows Pentium-III systems, running MATLAB version 5.1–5.3. In our experience, Intel PIII-based LINUX systems typically provided more accurate results, apparently utilizing the extended 80 bit precision of FPU registers of PIII; see the discussion at the end of the section.

The main goal of our numerical tests is to check a practical robustness of our code, using the following argument. According to our analysis of section 5 and similar results of [3, 27, 14, 15], an absolute change in cosine and sine of principal angles is bounded by perturbations in matrices F and G, with the constant, proportional to their condition numbers taken after a proper column scaling; see Theorems 5.6 and 5.12 and Remark 5.4. Assuming a perturbation of entries of matrices F and G at the level of double precision, $EPS \approx 10^{-16}$, we expect a similar perturbation in cosine and sine of principal angles, when matrices F and G are well-conditioned after column scaling. We want to check if our code achieves this accuracy in practice.

We concentrate here on testing our sine-based algorithms, i.e., for principal angles smaller than $\pi/4$. The cosine-based algorithm with $A = I$ is recently studied in [10].

Our first example is taken from [3] with $p = 13$ and $m = 26$. Matrices F and G were called A and B in [3]. F was orthogonal, while G was an m-by-p Vandermonde matrix with $\operatorname{cond}(G) \approx 10^4$. Matrix G was generated in double precision and then rounded to single precision.

According to our theory and a perturbation analysis of [3, 27, 14, 15], in this example an absolute change in principal angles is bounded by a perturbation in matrix G times its condition number. Thus, we should expect sine and cosine of principal angles computed in [3] to be accurate with approximately four decimal digits.

In our code, all computations are performed in double precision; therefore, answers in Table 7.1 should be accurate up to twelve decimal digits. We observe, as expected, that our results are consistent with those of [3] within four digits.

TABLE 7.1
Computed sine and cosine of principal angles of the example of [3].

k	$\sin(\theta_k)$	$\cos(\theta_k)$
1	0.00000000000	1.00000000000
2	0.05942261363	0.99823291519
3	0.06089682091	0.99814406635
4	0.13875176720	0.99032719194
5	0.14184708183	0.98988858230
6	0.21569434797	0.97646093022
7	0.27005046021	0.96284617096
8	0.33704307148	0.94148922881
9	0.39753678833	0.91758623677
10	0.49280942462	0.87013727135
11	0.64562133627	0.76365770483
12	0.99815068733	0.06078820101
13	0.99987854229	0.01558527040

In our next series of tests, we assume n to be even and $p = q \leq n/2$. Let D be a diagonal matrix of the size p:

$$D = \operatorname{diag}(d_1, \ldots, d_p), \quad d_k > 0, \quad k = 1, \ldots, p.$$

We first define n-by-p matrices

$$F_1 = [I\ 0]^T, \quad G_1 = [I\ D\ 0]^T, \tag{7.1}$$

where I is the identity matrix of the size p and 0 are zero matrices of appropriate sizes. We notice that condition numbers of F_1 and G_1 are, respectively, one and

$$\operatorname{cond} G_1 = \sqrt{\frac{1 + (\max\{\operatorname{diag}(D)\})^2}{1 + (\min\{\operatorname{diag}(D)\})^2}}.$$

Thus, the condition number may be large only when large diagonal entries in D are present. Yet in this case, the condition number of G_1 can be significantly reduced by column scaling; see Remark 5.4.

The exact values of sine and cosine of principal angles between column-spaces of matrices F_1 and G_1 are obviously given by

$$\mu_k = \frac{d_k}{\sqrt{1 + d_k^2}}, \quad \sigma_k = \frac{1}{\sqrt{1 + d_k^2}}, \qquad k = 1, \ldots, p, \tag{7.2}$$

respectively, assuming that d_k's are sorted in the increasing order. The collective error in principal angles is measured as the following sum:

$$\sqrt{(\mu_1 - \tilde{\mu}_1)^2 + \cdots + (\mu_p - \tilde{\mu}_p)^2} + \sqrt{(\sigma_1 - \tilde{\sigma}_1)^2 + \cdots + (\sigma_p - \tilde{\sigma}_p)^2}, \tag{7.3}$$

where μ's are the sine and σ's are the cosine of principal angles, and the tilde sign $\tilde{}$ is used for actual computed values.

We multiply matrices F_1 and G_1 by a random orthogonal matrix U of the size n on the left to get

$$F_2 = U * F_1, \quad G_2 = U * G_1. \tag{7.4}$$

This transformation does not change angles and condition numbers. It still allows for improving the condition number of G_2 by column scaling.

Finally, we multiply matrices by random orthogonal p-by-p matrices T_F and T_G, respectively, on the right:

$$F_3 = F_2 * T_F, \quad G_3 = G_2 * T_G. \tag{7.5}$$

This transformation does not change angles or condition numbers. It is likely, however, to remove the possibility of improving the condition number of G_3 by column scaling. Thus, if G_3 is ill-conditioned, we could expect a loss of accuracy; see Theorems 5.6 and 5.12 and Remark 5.4.

We start by checking scalability of the code for well-conditioned cases. We increase the size of the problem n and plot the collective error, given by (7.3), for the principal angles between F_3 and G_3, against the value $n/2$. We solve the same problem two times, using Algorithm 3.2 and Algorithm 6.2 with $A = I$.

In the first two tests, diagonal entries of D are chosen as uniformly distributed random numbers $rand$ on the interval $(0, 1)$. On Figure 7.1 (top) we fix $p = q = 20$. We observe that the average error grows approximately two times with a ten times increase in the problem size. On Figure 7.1 (bottom), we also raise $p = q = n/2$. This time, the error grows with the same pace as the problem size. Please note different scales used for errors on Figure 7.1.

To test our methods for very small angles with $p = q = 20$, we first choose

$$p = 20, \quad d_k = 10^{-k}, \quad k = 1, \dots, p.$$

We observe a similar pattern of the absolute error as that of Figure 7.1 (top) and do not reproduce this figure here because of the space limitations.

In our second test for small angles, we set $p = q = n/2$ as on Figure 7.1 (bottom) and select every diagonal element of D in the form $10^{-17 \cdot rand}$, where $rand$ is again a uniformly distributed random number on the interval $(0, 1)$. The corresponding figure, not shown here for the sake of brevity, looks the same as Figure 7.1 (bottom), except that the error grows a bit faster and reaches the level $\approx 4 \cdot 10^{-14}$ (compare to $\approx 3 \cdot 10^{-14}$ value on Figure 7.1 (bottom)).

In all these and our other analogous tests, Algorithm 3.2 and Algorithm 6.2 with $A = I$ behave very similarly, so that Figure 7.1 provides a typical example.

In our next series of experiments, we fix a small $p = q$ and $n = 100$, and compute angles between F_2 and G_2 and between F_3 and G_3 500 times, changing only the random matrices used in the construction of our F_i and G_i. Instead of the collective error, given by (7.3), we now compute errors for individual principal angles as

$$|\mu_k - \tilde{\mu}_k| + |\sigma_k - \tilde{\sigma}_k|, \qquad k = 1, \dots, p.$$

We tested several different combinations of angles less than $\pi/4$. In most cases, the error was only insignificantly different from $EPS \approx 10^{-16}$. The worst-case scenario found numerically corresponds to

$$D = \mathrm{diag}\{1, 0.5, 10^{-11}, 10^{-12}, 10^{-13}, 5 \cdot 10^{-15}, 2 \cdot 10^{-15}, 10^{-15}, 10^{-16}, 0\}$$

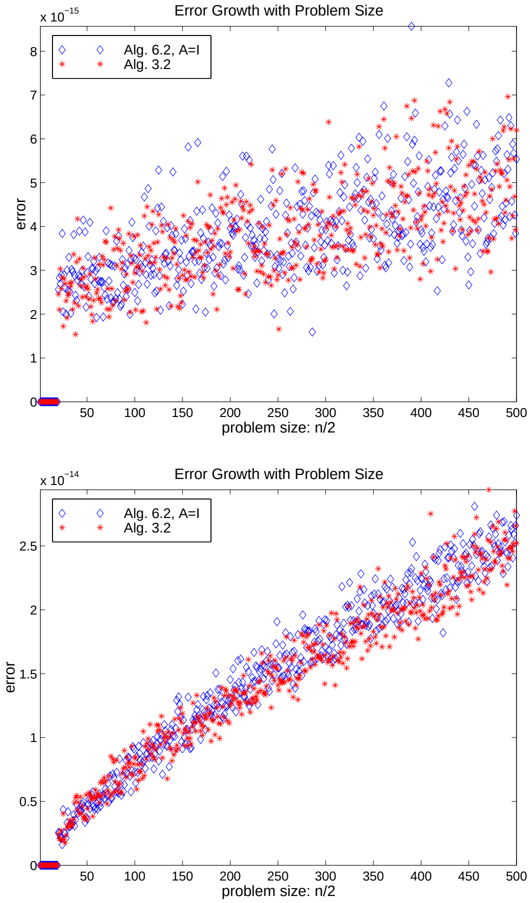


FIG. 7.1. *Errors in principal angles as functions of* $n/2$*:* $p = 20$ *(top) and* $p = n/2$ *(bottom).*

and is presented on Figure 7.2. Figure 7.2 (top) shows the distribution of the error for individual angles between F_3 and G_3 in Algorithm 3.2 in 500 runs. Figure 7.2 (bottom) demonstrates the performance of Algorithm 6.2 with $A = I$, for the same problem. The numeration of angles is reversed for technical reasons; i.e., smaller angles are further away from the viewer. Also the computed distribution of the error for individual angles between F_2 and G_2 are very similar and, because of that, are not shown here.

We detect that for such small values of p and n most of the angles are computed essentially within the double precision accuracy. Only multiple small angles present a slight challenge, more noticeable on Figure 7.2 (bottom), which uses a somewhat different scale for the error to accommodate a larger error. Nevertheless, all observed errors in all 500 runs are bounded from above by $\approx 6\cdot 10^{-15}$ on Figure 7.2, which seems to be a reasonable level of accuracy for accumulation of round-off errors appearing in computations with 200-by-10 matrices in double precision.

To test our code for an ill-conditioned case, we add two large values to the previous choice of D to obtain

$$D = \operatorname{diag}\{10^{10}, 10^{8}, 1, 0.5, 10^{-11}, 10^{-12}, 10^{-13}, 5\cdot 10^{-15}, 2\cdot 10^{-15}, 10^{-15}, 10^{-16}, 0\},$$

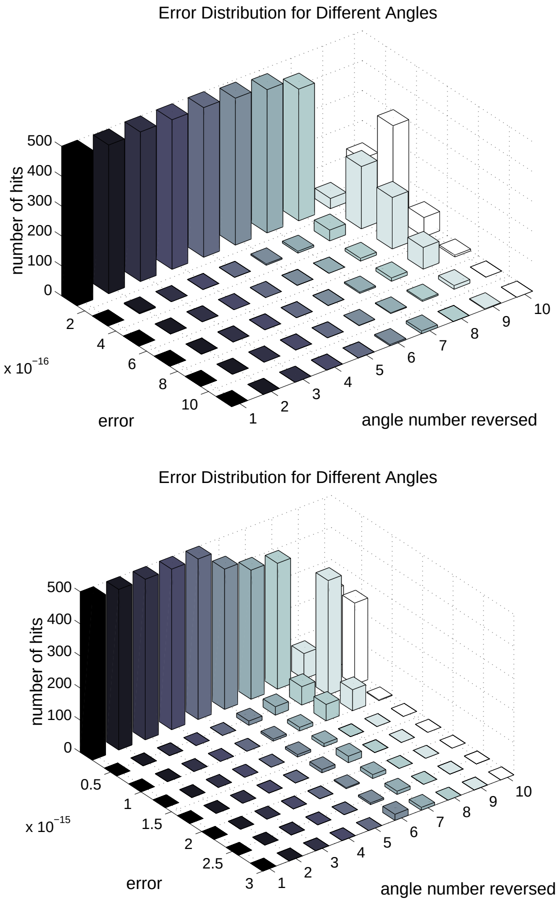
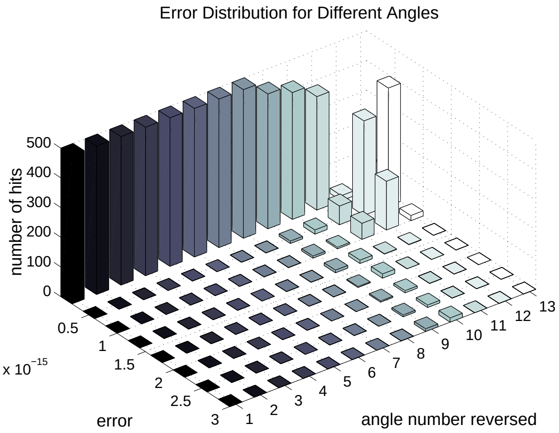


FIG. 7.2. *Errors in individual angles in Algorithms* 3.2 *(top) and* 6.2 *with* $A = I$ *(bottom).*

which leads to $\operatorname{cond} G_1 \approx 10^{10}$.

Figure 7.3 (top) shows the distribution of the error for individual angles between F_2 and G_2 in Algorithm 6.2 with $A = I$ after 500 runs. The numeration of angles is again reversed; i.e., smaller angles are further away from the viewer. There is no visible difference between the new Figure 7.3 (top) and the previous Figure 7.2 for the well-conditioned case, which confirms results of [15, 14] on column scaling of ill-conditioned matrices; see Remark 5.4. Namely, our ill-conditioned matrix G_2 can be made well-conditioned by column scaling. Thus, perturbations in the angles should be small. Figure 7.3 (top) therefore demonstrates that our code SUBSPACEA.m is able to take advantage of this.

As we move to computing angles between F_3 and G_3 in this example (see Figure 7.3 (bottom)), where the distribution of the error for individual angles in Algorithm 6.2 with $A = I$ after 500 runs is shown, the situation changes dramatically. In general, it is not possible to improve the condition number $\operatorname{cond}(G_3) \approx 10^{10}$ by column scaling. Thus, according to [3, 15] and our perturbation analysis of Theorems 5.6 and 5.12 and Remark 5.4, we should expect the absolute errors in angles to grow $\approx 10^{10}$ times (compare to the errors on Figure 7.3 (bottom)), i.e., up to the level 10^{-5}. On Figure



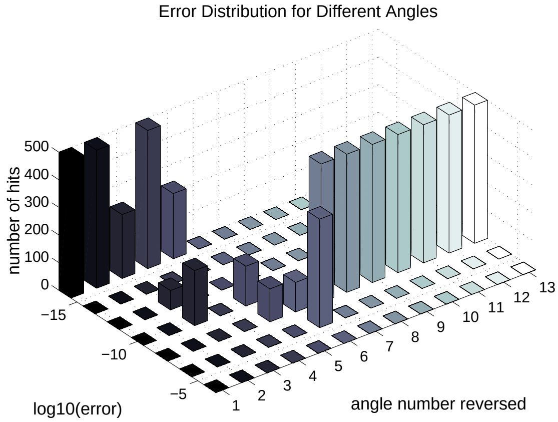


FIG. 7.3. *Errors in individual angles between F_2 and G_2 (top) and F_3 and G_3 (bottom).*

7.3 (bottom), which shows actual errors obtained during 500 runs of the code, we indeed observe absolute errors in angles at the level up to 10^{-5}, as just predicted. Surprisingly, the absolute error for angles with small cosines is much better. We do not have an explanation of such good behavior, as the present theory does not provide individual perturbation bounds for different angles.

Our concluding numerical results illustrate performance of our code for ill-conditioned scalar products.

We take G to be the first ten columns of the identity matrix of size twenty, and F to be the last ten columns of the Vandermonde matrix of size twenty with elements $v_{i,j} = i^{20-j}$, $i, j = 1, \ldots, 20$. Matrix F is ill-conditioned, cond$F \approx 10^{13}$. We compute principal angles and vectors between F and G in an A-based scalar product for the following family of matrices:

$$A = A_l = 10^{-l} I + H, \qquad l = 1, \ldots, 16,$$

where I is identity and H is the Hilbert matrix of the order twenty, whose elements are given by $h_{i,j} = 1/(i + j - 1)$, $i, j = 1, \ldots, 20$. Our subspaces $\mathcal{F}$ and $\mathcal{G}$ do not change with l; only the scalar product that describes the geometry of the space changes.

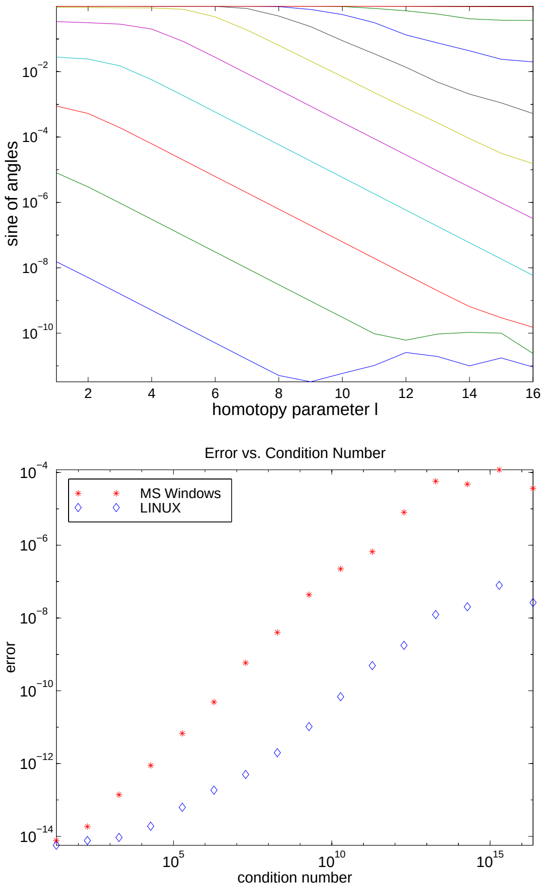


FIG. 7.4. *μ_k as functions of l (top). Errors as functions of condition number (bottom).*

When $l = 1$, we observe three angles with cosine less than 10^{-3} and three angles with sine less than 10^{-3}. When l increases, we are getting closer to the Hilbert matrix, which emphasizes first rows in matrices F and G, effectively ignoring last rows. By construction of F, its large elements, which make subspace $\mathcal{F}$ to be further away from subspace $\mathcal{G}$, are all in last rows. Thus, we should expect large principal angles between $\mathcal{F}$ and $\mathcal{G}$ to decrease monotonically as l grows. We observe this in our numerical tests (see Figure 7.4 (top)), which plots in logarithmic scale sine of all ten principal angles as functions of l.

Of course, such change in geometry that makes sine of an angle to decrease 10^4 times, means that matrix A_l, describing the scalar product, gets more and more ill-conditioned, as it gets closer to Hilbert matrix H, namely, $\text{cond}(A) \approx 10^l$ in our case. It is known that ill-conditioned problems usually lead to a significant increase of the resulting error, as ill-conditioning amplifies round-off errors. To investigate this effect for our code, we introduce the error as the following sum:

$$\text{error} = \|V^T AV - I\| + \|U^T AU - I\| + \|\Sigma - U^T AV\|,$$

where the first two terms control orthogonality of principal vectors and the last term measures the accuracy of cosine of principal angles. We observe in our experiments that different terms in the sum are close to each other, and none dominates. The accuracy of sine of principal angles is not crucial in this example as angles are not small enough to cause concerns. As U and V are constructed directly from columns of F and G, they span the same subspaces with high accuracy independently of condition number of A, as we observe in the tests.

We plot the error on the y-axis of Figure 7.4 (bottom) for a Pentium III 500 PC running two different operating systems: Microsoft Windows NT 4.0 SP6 (red stars) and RedHat LINUX 6.1 (blue diamonds), where the x-axis presents condition number of A; both axes are in logarithmic scale. The MATLAB Version 5.3.1.29215a (R11.1) is the same on both operating systems.

We see, as expected, that the error grows, apparently linearly, with the condition number. We also observe, now with quite a surprise, that the error on LINUX is much smaller than the error on MS Windows!

As the same MATLAB's version and the same code SUBSPACEA.m are run on the same hardware, this fact deserves an explanation. As a result of a discussion with Nabeel and Lawrence Kirby at the News Group *sci.math.num-analysis*, it has been found that MATLAB was apparently compiled on LINUX to take advantage of extended 80 bit precision of FPU registers of PIII, while Microsoft compilers apparently set the FPU to 64 bit operations. To demonstrate this, Nabeel suggested the following elegant example: compute scalar product

$$(1\ 10^{-19}\ \ -1)^T\ (1\ 1\ 1).$$

On MS Windows, the result is zero, as it should be in double precision, while on LINUX the result is 1.084210^{-19}.

Figure 7.4 (bottom) shows that our algorithm turns this difference into a significant error improvement for an ill-conditioned problem.

Finally, our code SUBSPACEA.m has been used since 1999 in the code LOBPCG.m (see [20, 21]) to control accuracy of invariant subspaces of large symmetric generalized eigenvalue problems, and thus has been tested for a variety of large-scale practical problems.

8. Availability of the software.

See http://www.mathworks.com/support/ftp/linalgv5.shtml for our code SUBSPACEA.m and the function ORTHA.m it uses, as well as our fix for SUBSPACE.m, as submitted to MathWorks. They are also publicly available at the Web page maintained by the first author: http://www-math.cudenver.edu/˜aknyazev/software.

9. Conclusion.

Let us formulate here the main points of the present paper:

- A bug in the cosine-based algorithm for computing principal angles between subspaces, which prevents one from computing small angles accurately in computer arithmetic, is illustrated.
- An algorithm is presented that computes all principal angles accurately in computer arithmetic and is proved to be equivalent to the standard algorithm in exact arithmetic.
- A generalization of the algorithm to an arbitrary scalar product given by a symmetric positive definite matrix is suggested and justified theoretically.
- Perturbation estimates for absolute errors in cosine and sine of principal angles, with improved constants and for an arbitrary scalar product, are derived.

- A description of the code is given as well as results of numerical tests. The code is robust in practice and provides accurate angles for large-scale and ill-conditioned cases we tested numerically. It is also reasonably efficient for large-scale applications with $n \gg p$.
- Our algorithms are "matrix-free"; i.e., they do not require storing in memory any matrices of the size n and are capable of dealing with A, which may be specified only as a function that multiplies A by a given vector.

Acknowledgments. The authors thank CU-Denver graduate students Sean Jenson, who helped to test our code SUBSPACEA.m, and Chan-Chai Aniwathananon and Saulo Oliveira, who participated in testing the code ORTHA.m.

The authors are also indebted to anonymous referees for their creative and constructive comments.

REFERENCES

[1] N. I. Akhiezer and I. M. Glazman, *Theory of Linear Operators in Hilbert Space*, Dover, New York, 1993.

[2] J. Barlow and J. Demmel, *Computing accurate eigensystems of scaled diagonally dominant matrices*, SIAM J. Numer. Anal., 27 (1990), pp. 762–791.

[3] Å. Björck and G. H. Golub, *Numerical methods for computing angles between linear subspaces*, Math. Comp., 27 (1973), pp. 579–594.

[4] F. Chatelin, *Eigenvalues of Matrices*, John Wiley and Sons, Chichester, UK, 1993.

[5] J. Dauxois and G. M. Nkiet, *Canonical analysis of two Euclidean subspaces and its applications*, Linear Algebra Appl., 264 (1997), pp. 355–388.

[6] C. Davis and W. M. Kahan, *The rotation of eigenvectors by a perturbation.* III, SIAM J. Numer. Anal., 7 (1970), pp. 1–46.

[7] J. Demmel, M. Gu, S. Eisenstat, I. Slapničar, K. Veselić, and Z. Drmač, *Computing the singular value decomposition with high relative accuracy*, Linear Algebra Appl., 299 (1999), pp. 21–80.

[8] J. Demmel and K. Veselić, *Jacobi's method is more accurate than QR*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 1204–1245.

[9] F. Deutsch, *The angle between subspaces of a Hilbert space*, in Approximation Theory, Wavelets and Applications, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1995, pp. 107–130.

[10] Z. Drmač, *On principal angles between subspaces of Euclidean space*, SIAM J. Matrix Anal. Appl., 22 (2000), pp. 173–194.

[11] D. A. Flanders, *Angles between flat subspaces of a real n-dimensional Euclidean space*, in Studies and Essays Presented to R. Courant on his 60th Birthday, January 8, 1948, Interscience Publishers, New York, 1948, pp. 129–138.

[12] I. C. Gohberg and M. G. Kreĭn, *Introduction to the Theory of Linear Nonselfadjoint Operators*, Transl. Math. Monogr. 18, AMS, Providence, RI, 1969.

[13] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed., Johns Hopkins University Press, Baltimore, MD, 1996.

[14] G. H. Golub and H. Y. Zha, *Perturbation analysis of the canonical correlations of matrix pairs*, Linear Algebra Appl., 210 (1994), pp. 3–28.

[15] G. H. Golub and H. Y. Zha, *The canonical correlations of matrix pairs and their numerical computation*, in Linear Algebra for Signal Processing, Springer-Verlag, New York, 1995, pp. 27–49.

[16] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia, PA, 1996.

[17] H. Hotelling, *Relation between two sets of variables*, Biometrica, 28 (1936), pp. 322–377.

[18] T. Kato, *Perturbation Theory for Linear Operators*, Springer-Verlag, New York, 1976.

[19] A. V. Knyazev, *New estimates for Ritz vectors*, Math. Comp., 66 (1997), pp. 985–995.

[20] A. V. Knyazev, *Preconditioned eigensolvers—an oxymoron?*, Electron. Trans. Numer. Anal., 7 (1998), pp. 104–123.

[21] A. V. Knyazev, *Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method*, SIAM J. Sci. Comput., 23 (2001), pp. 517–541.

[22] R.-C. Li, *Relative perturbation theory:* II. *Eigenspace and singular subspace variations*, SIAM J. Matrix Anal. Appl., 20 (1998), pp. 471–492.

[23] C. C. Paige and M. Wei, *History and generality of the CS decomposition*, Linear Algebra Appl., 208/209 (1994), pp. 303–326.

[24] B. N. Parlett, *The Symmetric Eigenvalue Problem*, SIAM, Philadelphia, PA, 1997.

[25] G. W. Stewart, *Computing the CS decomposition of a partitioned orthonormal matrix*, Numer. Math., 40 (1982), pp. 297–306.

[26] G. W. Stewart and J. G. Sun, *Matrix Perturbation Theory*, Academic Press, Boston, MA, 1990.

[27] J. G. Sun, *Perturbation of angles between linear subspaces*, J. Comput. Math., 5 (1987), pp. 58–61.

[28] C. Van Loan, *Computing the CS and the generalized singular value decompositions*, Numer. Math., 46 (1985), pp. 479–491.

[29] P. A. Wedin, *On angles between subspaces of a finite-dimensional inner product space*, in Matrix Pencils, Bo Kågström and Axel Ruhe, eds., Springer-Verlag, Berlin, 1983, pp. 263–285.

[30] H. K. Wimmer, *Canonical angles of unitary spaces and perturbations of direct complements*, Linear Algebra Appl., 287 (1999), pp. 373–379.

SIAM J. SCI. COMPUT.
Vol. 23, No. 6, pp. 2041–2054

FOURTH ORDER CHEBYSHEV METHODS WITH RECURRENCE RELATION[*]

ASSYR ABDULLE[†]

Dedicated to Professor Gerhard Wanner on the occasion of his 60th birthday

Abstract. In this paper, a new family of fourth order Chebyshev methods (also called stabilized methods) is constructed. These methods possess nearly optimal stability regions along the negative real axis and a three-term recurrence relation. The stability properties and the high order make them suitable for large stiff problems, often space discretization of parabolic PDEs. A new code ROCK4 is proposed, illustrated at several examples, and compared to existing programs.

Key words. stiff ordinary differential equations, explicit Runge–Kutta methods, orthogonal polynomials, parabolic partial differential equations

AMS subject classifications. 65L20, 65M20

PII. S1064827500379549

1. Introduction.

Chebyshev methods are a class of explicit Runge–Kutta methods with extended stability domains along the negative real axis. The stability properties of these methods make them suitable for stiff problems which possess a Jacobian matrix with (possibly large) eigenvalues close to the real negative axis. Since they are explicit, Chebyshev methods avoid linear algebra difficulties and can be applied to very large problems. The main applications are parabolic PDEs when discretized by finite difference. It usually gives a large system of ODEs with a symmetric and negative definite Jacobian matrix. Thus, the eigenvalues of the discretized parabolic PDEs are real negative and, furthermore, become larger while refining the space discretization.

Recently, a new strategy to construct second order Chebyshev methods has been proposed by Abdulle and Medovikov [2]. It combines the advantages of the methods introduced by Lebedev [11], [12] and van der Houwen and Sommeijer [9] (see also [14] for the latest implementation of these methods). An algorithm to construct nearly optimal stability functions along the real negative axis based on orthogonal polynomials was proposed in [2]. The advantage of using orthogonal polynomials is the three-term recurrence relation which can be used to construct the numerical methods. At the same time, choosing an appropriate weight function for these polynomials leads to a nearly optimal stability domain (see [2]).

For order more than 2, the only known Chebyshev methods are those of Medovikov [10]. They are constructed upon the strategy of Lebedev-type methods: the zeros of the optimal stability polynomials are computed, and the numerical methods are based on a suitable ordering of these zeros. The drawback is that the ordering, crucial for the internal stability of the methods, depends on the degree of the polynomials and needs some art. There are also no recurrence relations. The methods of order 4 proposed in this paper are based on a three-term recurrence relation and avoid the preceding problems.

[*] Received by the editors October 16, 2000; accepted for publication (in revised form) October 18, 2001; published electronically February 27, 2002.
http://www.siam.org/journals/sisc/23-6/37954.html

[†] Section de Mathématiques, Université de Genève, CH-1211 Genève 24, Switzerland (Assyr.Abdulle@math.unige.ch).

The paper is organized as follows. In section 2 we explain how to compute nearly optimal fourth order stability polynomials. Section 3 is devoted to the construction of a family of numerical methods. These methods have been implemented in a new code called ROCK4 which is briefly described in section 4. Finally, in section 5 we present some numerical experiments and comparisons with other codes.

2. Fourth order stability polynomials. The aim is to construct a family of polynomials of order 4 depending on the degree s,[1]

$$\widetilde{R}_s(z) = 1 + z + \frac{z^2}{2!} + \frac{z^3}{3!} + \frac{z^4}{4!} + \mathcal{O}(z^5), \tag{2.1}$$

which remain bounded by 1 as long as possible along the real negative axis, i.e.,

$$|\widetilde{R}_s(z)| \leq 1 \quad \text{for } z \in [-\widetilde{l}_s, 0], \tag{2.2}$$

with $\widetilde{l}_s$ as large as possible. It is proved in [1] that fourth order optimal stability polynomials (which exist and are unique see [13]) possess exactly four complex zeros. Thus, they can be written as

$$\widetilde{R}_s(z) = \widetilde{w}_4(z)\widetilde{P}_{s-4}(z), \tag{2.3}$$

where $\widetilde{w}_4(z) = (1-/t_1)(1-/\bar{t}_1)(1-/t_2)(1-/\bar{t}_2)$, t_i are complex (conjugate) numbers, and $\widetilde{P}_{s-4}(z)$ possesses only real zeros. The idea developed in [2] for second order is to approximate these polynomials by

$$R_s(z) = w_4(z)P_{s-4}(z), \tag{2.4}$$

where $w_4(z) = (1-/z_1)(1-/\bar{z}_1)(1-/z_2)(1-/\bar{z}_2)$ and $P_{s-4}(z)$ is an orthogonal polynomial associated with the weight function $w_4(z)^2/\sqrt{1-z^2}$. We want to find such a decomposition which satisfies (2.2) for $z \in [-l_s, 0]$ with l_s close to $\widetilde{l}_s$. At the same time, we want that the product satisfies the fourth order conditions (2.1).

The motivation for considering such polynomials can be found in [2]. Notice that for first order optimal polynomials we have a formula similar to (2.3), with $\widetilde{w}(z) = 1$ and $\widetilde{R}_s(z) = T_s(1 + \frac{z}{s^2})$, where $T_s(z)$ are the Chebyshev polynomials. These optimal polynomials are at the same time orthogonal with respect to the weight function $1/\sqrt{1-z^2}$. The arguments developed in [2] show that for order 2 the polynomials (2.3) and (2.4) are very close. These arguments can be generalized for even orders higher than 2. Figure 2.1 shows for $s = 9$ the difference between a fourth order optimal stability polynomial and its approximation. We see that they can hardly be distinguished.

For second order, the algorithm for computing the orthogonal polynomials and the zeros of the function $w(z)$ of (2.4) was given in [2]. We adapt here this algorithm for order 4.

In the following we will work in the normalized interval $[-1, 1]$ instead of $[-l_s, 0]$ by setting $x = 1 + \frac{2z}{l_s}$ (l_s is the length of the stability domain along $\mathbb{R}^-$ we want to optimize), and we take the same notation for the shifted polynomials except for the fact that we use the variable x instead of z. Thus, we are searching for order 4 at $x = 1$ (see below). If we shift the polynomials defined by (2.4) and normalize them

[1] We use the notation $\widetilde{R}_s$ because we reserve R_s for the stability polynomials we will construct.

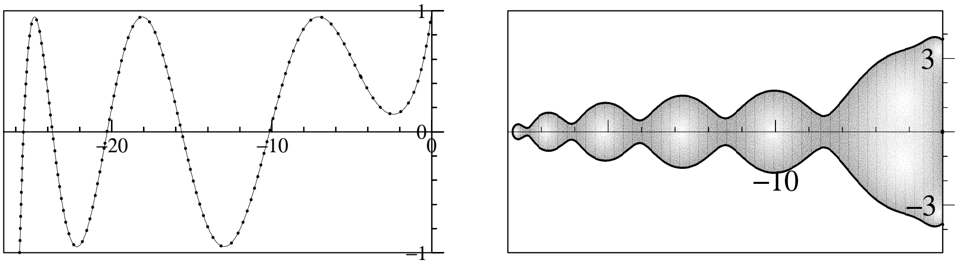


FIG. 2.1. $R_9(x)$ *and its stability domain, with* $\widetilde{R}_9(x)$ *in the dotted line,* $\eta = 0.95$ *(damping).*

such that $|w_4(x)P_{s-4}(x)| \le 1$ for $x \in [-1, 1]$, then $w_4(1)P_{s-4}(1)$ is usually not equal to 1. As in [2], we therefore introduce a parameter a close to 1, set

$$R_s(x) = \frac{w_4(x)P_{s-4}(x)}{w_4(a)P_{s-4}(a)}, \tag{2.5}$$

and we search fourth order conditions at the point a. We denote the complex zeros of $w_4(x)$ by $x_1 = \alpha + i\beta, \bar{x}_1 = \alpha - i\beta, x_2 = \gamma + i\delta, \bar{x}_2 = \gamma - i\delta$, and to emphasize the dependence of $R_s(x)$ on $\alpha, \beta, \gamma, \delta$ we will write $R_s(x, \alpha, \beta, \gamma, \delta)$. We have now an optimization problem.

Problem. Find $a, d, \alpha, \beta, \gamma, \delta$ (depending on s) such that

$$R_s'(a, \alpha, \beta, \gamma, \delta) = d, \qquad R_s''(a, \alpha, \beta, \gamma, \delta) = d^2, \tag{2.6}$$

$$R_s'''(a, \alpha, \beta, \gamma, \delta) = d^3, \qquad R_s^{(4)}(a, \alpha, \beta, \gamma, \delta) = d^4, \tag{2.7}$$

$$|R_s(x)| \le 1 \quad \text{for} \quad x \in [-1, a] \tag{2.8}$$

$$\text{with} \quad l_s = (1 + a)d \quad \text{as large as possible.} \tag{2.9}$$

If we then set $z = (x - a)/d$, we will have

$$R_s(0) = 1, \quad R_s'(0) = 1, \quad R_s''(0) = 1, \quad R_s'''(0) = 1, \quad R_s^{(4)}(0) = 1, \tag{2.10}$$

$$|R_s(z)| \le 1 \quad \text{for} \quad z \in [-l_s, 0]. \tag{2.11}$$

Here again we used the same notation for the shifted polynomials. For more details about this algorithm we refer the reader to [2].

We have computed the parameters $l_s, \alpha, \beta, \gamma, \delta, a$ (depending on s) for degree 5 up to degree more than 1000. In practice the bound in (2.8) should be replaced by a value $\eta < 1$ so that the stability domain is at a safe distance from the real axis (see Figure 2.1). We chose $\eta = 0.95$. In Table 2.1 we give the values of l_s and $c(s) = l_s/s^2$ for some degrees s. Several authors observed that the optimal values $\widetilde{l}_s$ satisfy $\widetilde{l}_s = c(s)s^2$, with $c(s)$ rapidly approaching a limit value. The value $c(s) \simeq 0.34$ is given in [15] (see also [8]) and $c(s) \simeq 0.35$ in [10]. We observe in Table 2.1 that $l_s \simeq 0.35 \cdot s^2$, which means that our stability regions are nearly optimal.

Table 2.1
The stability parameters of $R_s(x)$.

Degree s	Stability region l_s	Value $c(s) = l_s/s^2$	Degree s	Stability region l_s	Value $c(s) = l_s/s^2$
5	5.9983	0.239931	100	3538.1276	0.353813
10	32.4470	0.324470	250	22184.4995	0.354952
20	138.3586	0.345897	500	88746.9995	0.354988
50	879.8864	0.351955	750	199684.4999	0.354995

3. Fourth order Chebyshev methods.

In the preceding section we have explained how to compute our stability polynomials. Assume now that we have such a family of fourth order polynomials (depending on the degree s)

$$R_s(z) = w_4(z)P_{s-4}(z), \tag{3.1}$$

which satisfy

$$|R_s(z)| \leq 1 \quad \text{for} \quad z \in [-l_s, 0], \tag{3.2}$$

where $w_4(z) = (1 - /z_1)(1 - /\bar{z}_1)(1 - /z_2)(1 - /\bar{z}_2)$ and $P_{s-4}(z)$ is an orthogonal polynomial associated with the weight function

$$w_4(z)^2/\sqrt{1 - z^2}, \tag{3.3}$$

and normalized such that $P_{s-4}(0) = 1$. For a given degree s we will further use the family of orthogonal polynomials $(P_j)_{j=0}^{s-4}$ associated with the weight function (3.3), normalized such that $P_j(0) = 1$. These polynomials possess a three-term recurrence relation

$$P_j(z) = (\mu_j z - \nu_j)P_{j-1}(z) - \kappa_j P_{j-2}(z).$$

In [2] it was explained how to compute explicitly these polynomials, given the zeros of the function (3.3). The same procedure can be applied here, and the recurrence coefficients can be computed simply by solving a linear system.

For second order methods, it is sufficient to construct a numerical method which is of order 2 for linear problems, since the order conditions are the same for both linear and nonlinear problems. This is not true for orders larger than 2, and it is therefore not sufficient to consider only the linear case. Here we have to give a realization of our weight function $w_4(z)$ so that the order conditions up to order 4 (8 conditions) are satisfied. As in [10] we will use the theory of composition of methods (the "Butcher group") to realize fourth order Runge–Kutta–Chebyshev methods.

Suppose that we have two Runge–Kutta methods. The idea of composition of methods is to apply one method after the other to an initial value y_0 with the same step size. The result of this process can be interpreted as a large Runge–Kutta method: a composition of the two latter methods. For the theory of composition of Runge–Kutta methods, we refer to [3],[6, pp. 264–273],[5].

To construct a fourth order Runge–Kutta method with the polynomials (3.1) we proceed as follows:

- We construct a first method, denoted by P, which possesses $P_{s-4}(z)$ as stability polynomial.
- We then determine a second method, denoted by W, which possesses $w_4(z)$ as stability polynomial, to achieve fourth order for the "composite" method denoted by WP.

$$\begin{array}{c|cccc}
0 & & & & \\
\widetilde{c}_2 & \widetilde{a}_{21} & & & \\
\vdots & \vdots & \ddots & & \\
\widetilde{c}_{s-4} & \widetilde{a}_{s-4,1} & \dots & \widetilde{a}_{s-4,s-5} & \\
\hline
 & \widetilde{b}_1 & \dots & \widetilde{b}_{s-5} & \widetilde{b}_{s-4}
\end{array}
\qquad\qquad
\begin{array}{c|cccc}
0 & & & & \\
\widehat{c}_2 & \widehat{a}_{21} & & & \\
\widehat{c}_3 & \widehat{a}_{31} & \widehat{a}_{32} & & \\
\widehat{c}_4 & \widehat{a}_{41} & \widehat{a}_{42} & \widehat{a}_{43} & \\
\hline
 & \widehat{b}_1 & \widehat{b}_2 & \widehat{b}_3 & \widehat{b}_4
\end{array}$$

FIG. 3.1. *Tableau of the method P (left) and W (right).*

The resulting method will be of order 4 and will possess $R_s(z) = w_4(z)P_{s-4}(z)$ as stability polynomial.

The first method. To construct the method P we apply a procedure similar to that used in [2]. That is, the three-term recurrence relation of the orthogonal polynomials $(P_j)_{j=1}^{s-4}$ is used to define the internal stages of the method as follows:

$$\begin{aligned}
&g_0 := y_0, \\
&g_1 := y_0 + h\mu_1 f(g_0), \\
&g_j := h\mu_j f(g_{j-1}) - \nu_j g_{j-1} - \kappa_j g_{j-2}, \qquad\qquad j = 2, \dots, s-4, \\
&y_1 := g_{s-4}.
\end{aligned} \tag{3.4}$$

Applied to $y' = \lambda y$ with $z = h\lambda$ yields

$$g_{s-4} = P_{s-4}(z)g_0. \tag{3.5}$$

This method is given in the left tableau of Figure 3.1. The coefficients of the method P can be expressed recursively in term of the coefficients ν_j, μ_j, κ_j. Indeed, using the notation $k_i = f(y_0 + h\sum_{j=1}^{i-1} \widetilde{a}_{ij}k_j)$ (autonomous form), we obtain for the first stage

$$g_1 = y_0 + h\mu_1 f(y_0) = y_0 + h\widetilde{a}_{21}k_1; \tag{3.6}$$

this yields $a_{21} = \mu_1$. For the second stage we have

$$\begin{aligned}
g_2 &= h\mu_2 f(y_0 + h\widetilde{a}_{21}k_1) - \nu_2(y_0 + h\widetilde{a}_{21}k_1) - \kappa_2 y_0 \\
&= y_0(-\nu_2 - \kappa_2) + h(-\nu_2\widetilde{a}_{21})k_1 + h\mu_2 k_2;
\end{aligned} \tag{3.7}$$

hence $\widetilde{a}_{31} = -\nu_2\widetilde{a}_{21}$ and $\widetilde{a}_{32} = \mu_2$. (Notice that $-\nu_2 - \kappa_2 = 1$ because of the normalization $P_j(0) = 1$.) The third stage is then given by

$$\begin{aligned}
g_3 &= h\mu_3 f(y_0 + h(\tilde{a}_{31}k_1 + \tilde{a}_{32}k_2)) - \nu_3(y_0 + h(\tilde{a}_{31}k_1 + \tilde{a}_{32}k_2)) - \kappa_3(y_0 + h\tilde{a}_{21}k_1) \\
&= y_0(-\nu_3 - \kappa_3) + h(-\nu_3\tilde{a}_{31} - \kappa_3\tilde{a}_{21})k_1 + h(-\nu_3\tilde{a}_{32})k_2 + h\mu_3 k_3,
\end{aligned}$$

and thus $\tilde{a}_{41} = -\nu_3\tilde{a}_{31} - \kappa_3\tilde{a}_{21}$, $\tilde{a}_{42} = -\nu_3\tilde{a}_{32}$ and $\tilde{a}_{43} = \mu_3$. By induction we obtain the following lemma.

LEMMA 3.1. *For the method P given by* (3.4) *the coefficients $\widetilde{a}_{ij}, \widetilde{b}_i, \widetilde{c}_i$ of the corresponding Runge–Kutta method are given by*

$$\begin{array}{lcll}
\widetilde{a}_{i,j} & = & -\nu_{i-1}\widetilde{a}_{i-1,j} - \kappa_{i-1}\widetilde{a}_{i-2,j}, & j \le i-2, \qquad i \le s-3 \ (\widetilde{a}_{jj} := 0), \\
\widetilde{a}_{i,i-1} & = & \mu_{i-1}, & i \le s-3, \\
\widetilde{b}_j & = & \widetilde{a}_{s-3,j}, & 1 \le j \le s-4,
\end{array} \tag{3.8}$$

and the $\widetilde{c}_i$ satisfy the usual relation $\widetilde{c}_i = \sum_{j=1}^{i-1} \widetilde{a}_{ij}$. □

We emphasize that the coefficients $\tilde{a}_{ij}, \widetilde{b}_i$ will be used only to compute the method W. For the implementation of the method, formulas (3.4) will be used.

$$
\begin{array}{c|cccccccc}
0 & & & & & & & & \\
\widetilde{c}_2 & \widetilde{a}_{21} & & & & & & & \\
\vdots & \vdots & \ddots & & & & & & \\
\widetilde{c}_{s-4} & \widetilde{a}_{s-4,1} & \dots & \widetilde{a}_{s-4,s-5} & & & & & \\
\sum \widetilde{b}_i & \widetilde{b}_1 & \dots & \widetilde{b}_{s-3} & \widetilde{b}_{s-4} & & & & \\
\sum \widetilde{b}_i + \widehat{c}_2 & \widetilde{b}_1 & \dots & \widetilde{b}_{s-3} & \widetilde{b}_{s-4} & \widehat{a}_{21} & & & \\
\vdots & \vdots & & & & & \ddots & & \\
\sum \widetilde{b}_i + \widehat{c}_4 & \widetilde{b}_1 & \dots & \widetilde{b}_{s-3} & \widetilde{b}_{s-4} & \widehat{a}_{41} & \dots & \widehat{a}_{43} & \\
\hline
 & \widetilde{b}_1 & \dots & \widetilde{b}_{s-3} & \widetilde{b}_{s-4} & \widehat{b}_1 & \dots & \widehat{b}_3 & \widehat{b}_4
\end{array}
$$

FIG. 3.2. *Tableau of method WP.*

The second and the "composite" methods. For the method W we take a fourth stage method (right tableau of Figure 3.1) so that the composite method WP is given by Figure 3.2. In the tableau of method WP, we will denote by c_i the elements of the first column, by a_{ij} the elements of the "triangle," and by b_i the elements of the last row. The order conditions of the method WP are the usual ones for order 4:

$$
\begin{array}{lclcl}
wp(\bullet) & = & \sum b_i & = & 1, \\
wp(/) & = & 2\sum b_i a_{ij} & = & 1, \\
wp(\vee) & = & 3\sum b_i a_{ij} a_{ik} & = & 1, \\
wp(\rangle) & = & 6\sum b_i a_{ij} a_{jk} & = & 1, \\
wp(\Psi) & = & 4\sum b_i a_{ij} a_{ik} a_{il} & = & 1, \\
wp(\dot{\vee}) & = & 8\sum b_i a_{ij} a_{jk} a_{il} & = & 1, \\
wp(Y) & = & 12\sum b_i a_{ij} a_{jk} a_{jl} & = & 1, \\
wp(\wr) & = & 24\sum b_i a_{ij} a_{jk} a_{kl} & = & 1.
\end{array}
\tag{3.9}
$$

Here we used the trees notation (connected graphs without cycles and a distinguished vertex) for the elementary weights ($wp(\dots)$) involved in the order conditions (see [6, pp. 145–154] or [3]).

Theorem 12.6 of [6, p. 267] can be used to express the order conditions of the method WP in function of the two submethods, W and P. See also [5] for a new simple proof of this latter theorem (with another normalization for the elementary weights). We obtain

$$
\begin{array}{lcl}
wp(\bullet) & = & w(\bullet) + p(\bullet), \\
wp(/) & = & w(/) + 2w(\bullet)p(\bullet) + p(/), \\
wp(\vee) & = & w(\vee) + 3w(/)p(\bullet) + 3w(\bullet)p(\bullet)^2 + p(\vee), \\
wp(\rangle) & = & w(\rangle) + 3w(/)p(\bullet) + 3w(\bullet)p(/) + p(\rangle), \\
wp(\Psi) & = & w(\Psi) + 4w(\vee)p(\bullet) + 6w(/)p(\bullet)^2 + 4w(\bullet)p(\bullet)^3 + p(\Psi),
\end{array}
$$

$$\begin{aligned}
wp(\overset{\bullet}{\ddot{\vee}}) &= w(\overset{\bullet}{\ddot{\vee}}) + 4(\tfrac{1}{3}w(\ddot{\rangle})p(\bullet) + \tfrac{2}{3}w(\ddot{\vee})p(\bullet)) + 6(\tfrac{1}{3}w(\diagup)p(\diagup) + \tfrac{2}{3}w(\diagup)p(\bullet)^2) \\
&\quad + 4w(\bullet)p(\bullet)p(\diagup) + p(\overset{\bullet}{\ddot{\vee}}), \\
wp(\ddot{Y}) &= w(\ddot{Y}) + 4w(\ddot{\rangle})p(\bullet) + 6w(\diagup)p(\bullet)^2 + 4w(\bullet)p(\ddot{\vee}) + p(\ddot{Y}), \\
wp(\overset{\bullet}{\ddot{\rangle}}) &= w(\overset{\bullet}{\ddot{\rangle}}) + 4w(\ddot{\rangle})p(\bullet) + 6w(\diagup)p(\diagup) + 4w(\bullet)p(\ddot{\rangle}) + p(\overset{\bullet}{\ddot{\rangle}}),
\end{aligned}$$

where $w(\dots)$ and $p(\dots)$ are the expressions (3.9) for the methods W and P, respectively. These formulas allow us to compute recursively the expressions $w(\bullet), w(\diagup), \dots$ for the method W, since the expressions $p(\bullet), p(\diagup), \dots$ can be computed with the coefficients of the method P given by (3.8). This leads to the following equations for the method W:

$$\begin{aligned}
&\widehat{b}_1 + \widehat{b}_2 + \widehat{b}_3 + \widehat{b}_4 &&= w(\bullet), \\
&\widehat{b}_2\widehat{c}_2 + \widehat{b}_3\widehat{c}_3 + \widehat{b}_4\widehat{c}_4 &&= \tfrac{w(\diagup)}{2}, \\
&\widehat{b}_2\widehat{c}_2^2 + \widehat{b}_3\widehat{c}_3^2 + \widehat{b}_4\widehat{c}_4^2 &&= \tfrac{w(\ddot{\vee})}{3}, \\
(3.10)\qquad &\widehat{b}_3\widehat{a}_{32}\widehat{c}_2 + \widehat{b}_4(\widehat{a}_{42}\widehat{c}_2 + \widehat{a}_{43}\widehat{c}_3) &&= \tfrac{w(\ddot{\rangle})}{6}, \\
&\widehat{b}_2\widehat{c}_2^3 + \widehat{b}_3\widehat{c}_3^3 + \widehat{b}_4\widehat{c}_4^3 &&= \tfrac{w(\dddot{\vee})}{4}, \\
&\widehat{b}_3\widehat{c}_3\widehat{a}_{32}\widehat{c}_2 + \widehat{b}_4\widehat{c}_4(\widehat{a}_{42}\widehat{c}_2 + \widehat{a}_{43}\widehat{c}_3) &&= \tfrac{w(\overset{\bullet}{\ddot{\vee}})}{8}, \\
&\widehat{b}_3\widehat{a}_{32}\widehat{c}_2^2 + \widehat{b}_4(\widehat{a}_{42}\widehat{c}_2^2 + \widehat{a}_{43}\widehat{c}_3^2) &&= \tfrac{w(\ddot{Y})}{12}, \\
&\widehat{b}_4\widehat{a}_{43}\widehat{a}_{32}\widehat{c}_2 &&= \tfrac{w(\overset{\bullet}{\ddot{\rangle}})}{24}.
\end{aligned}$$

This is a system of 8 equations for 10 unknowns $\widehat{b}_i, \widehat{a}_{ij}$ ($\widehat{c}_i$ are determined by $\widehat{c}_i = \sum_{j=1}^{i-1} \widehat{a}_{ij}$.) These equations are similar to the equations of usual fourth order methods (i.e., when $w(\bullet) = 1, w(\diagup) = 1, \dots$; see [6, pp. 135–136]). We have two degrees of freedom, and we choose $\widehat{c}_3 = \frac{w(\bullet)}{3}$ and $\widehat{c}_4 = \frac{2w(\bullet)}{3}$ after some experimentation. This choice keeps the absolute value of the coefficients $\widehat{b}_i, \widehat{a}_{ij}$ less than 1, which is suitable for a numerical method.

The solution of equations (3.10) gives us the coefficients of the method W. Thus, we have obtained a family of methods (depending on the degree of the stability polynomial) of order 4 with recurrence formulas and with the polynomial (3.1) as a stability function.

The embedded method. For the estimation of the local error of the constructed numerical method

$$(3.11)\qquad y_1 = y_0 + h\sum_{i=1}^{s} b_i f\left(y_0 + h\sum_{j=1}^{i-1} a_{ij}k_j\right) = y_0 + h\sum_{i=1}^{s} b_i k_i,$$

we use an embedded method of order 3.

Since for fourth order methods there is no embedded method of order 3 using the same function values $k_i = f(y_0 + h\sum a_{ij}k_j)$ (see [6, p. 167]), we search for a lower

$$
\begin{array}{c|ccccc}
0 & & & & & \\
\widehat{c}_2 & \widehat{a}_{21} & & & & \\
\widehat{c}_3 & \widehat{a}_{31} & \widehat{a}_{32} & & & \\
\widehat{c}_4 & \widehat{a}_{41} & \widehat{a}_{42} & \widehat{a}_{43} & & \\
\hline
\widehat{c}_5 & \widehat{b}_1 & \widehat{b}_2 & \widehat{b}_3 & \widehat{b}_4 & \\
\hline
 & \bar{b}_1 & \bar{b}_2 & \bar{b}_3 & \bar{b}_4 & \bar{b}_5
\end{array}
$$

FIG. 3.3. *Tableau of the method $\overline{W}$.*

order method of the form

$$\overline{y}_1 = y_0 + h\left(\sum_{i=1}^{s}\beta_i f\left(y_0 + h\sum_{j=1}^{i-1}a_{ij}k_j\right) + \beta_{s+1}f(y_1)\right) = y_0 + h\left(\sum_{i=1}^{s}\beta_i k_i + \beta_{s+1}k_{s+1}\right). \tag{3.12}$$

This is no extra work (if the step is accepted) because $f(y_1)$ has to be computed anyway for the next stage. As a measure of the error after one step we will take

$$err = \|y_1 - \overline{y}_1\|. \tag{3.13}$$

We want to keep the recurrence formulas for the embedded method. Therefore, the embedded method will be a composition of the method P, defined in (3.4), with a new method W denoted by $\overline{W}$. For that, we add a fifth stage to the method W as shown in Figure 3.3. Similarly to (3.10) we derive third order conditions for the method $\overline{W}$:

$$
\begin{array}{lcl}
\bar{b}_1 + \bar{b}_2 + \bar{b}_3 + \bar{b}_4 + \bar{b}_5 & = & w(\,\bullet\,), \\
\bar{b}_2\widehat{c}_2 + \bar{b}_3\widehat{c}_3 + \bar{b}_4\widehat{c}_4 + \bar{b}_4\widehat{c}_5 & = & \frac{w(\overset{\bullet}{/})}{2}, \\
\bar{b}_2\widehat{c}_2^2 + \bar{b}_3\widehat{c}_3^2 + \bar{b}_4\widehat{c}_4^2 + \bar{b}_5\widehat{c}_5^2 & = & \frac{w(\overset{\bullet\,\bullet}{\vee})}{3}, \\
\bar{b}_3\widehat{a}_{32}\widehat{c}_2 + \bar{b}_4(\widehat{a}_{42}\widehat{c}_2 + \widehat{a}_{43}\widehat{c}_3) + \bar{b}_5(\widehat{b}_2\widehat{c}_2 + \widehat{b}_3\widehat{c}_3 + \widehat{b}_4\widehat{c}_4) & = & \frac{w(\overset{\bullet}{\underset{\bullet}{\rangle}})}{6}.
\end{array} \tag{3.14}
$$

Notice that $\widehat{c}_5 = \sum\widehat{b}_i = w(\bullet)$. The last equation of (3.14) can be simplified since $\widehat{b}_2\widehat{c}_2 + \widehat{b}_3\widehat{c}_3 + \widehat{b}_4\widehat{c}_4 = \frac{w(\overset{\bullet}{/})}{2}$ (see (3.10)). We obtain a system of 4 equations for 5 unknowns $\bar{b}_i$. We require the following additional condition:

$$\bar{w}_5(-l_s) = 0, \tag{3.15}$$

where $\bar{w}_5(z)$ is the stability polynomial of the method $\overline{W}$ and l_s the length of the stability domain along the negative real axis (see (3.2)).

Solving (3.14) and (3.15), numerical computations show that the stability polynomials of the embedded methods are bounded (by $\eta = 0.95$) on the same interval as the stability polynomials of the numerical methods (see Figure 3.4). There is a simple criterion to check if $\bar{R}_{s+1}(z)$, the stability polynomials of the embedded method, is bounded by η on the same interval as $R_s(z)$. Recall that $R_s(z) = w_4(z)P_{s-4}(z)$

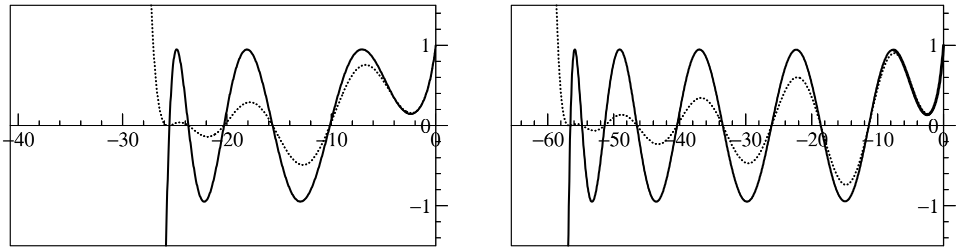


FIG. 3.4. *Stability polynomials of the method WP and the embedded method $\overline{W}P$ (dotted line) for $s = 9$ (left) and $s = 13$ (right).*

and $\bar{R}_{s+1}(z) = \bar{w}_5(z)P_{s-4}(z)$. Denote by z_i the zeros of $P_{s-4}(z)$. Because $P_{s-4}(z)$ is an orthogonal polynomial its zeros are simple and all in the stability interval, say $-l_s < z_1 < \cdots < z_{s-4} < 0$.

LEMMA 3.2. *With the above notations, suppose that $\bar{R}_{s+1}(-l_s) = 0$. If there exist $\delta > 0$ such that $\bar{R}_{s+1}(z) < R_s(z)$ for $z \in (z_{s-4}, z_{s-4} + \delta)$, then*

$$|\bar{R}_{s+1}(z)| \le |R_s(z)| \quad \textit{for } z \in (z_1, z_{s-4} + \delta). \tag{3.16}$$

Proof. Define $d(z) = \bar{R}_{s+1}(z) - R_s(z)$; we have

$$d(z) = P_{s-4}(z)z^4(\beta_0 + \beta_1 z),$$

since $\bar{R}_{s+1}(z) - e^z = \mathcal{O}(z^4)$ and $R_s(z) - e^z = \mathcal{O}(z^5)$.

Suppose that there exist $z \in (z_1, z_{s-4})$ such that $|\bar{R}_{s+1}(z)| > |R_s(z)|$. Then, using the second hypothesis, either there exist $\widehat{z} \neq z_i$ such that $d(\widehat{z}) = 0$ or $d(z)$ has a double zero at some z_i. In both cases, it must then exist $\epsilon > 0$ such that $|\bar{R}_{s+1}(z)| > |R_s(z)|$ for $z \in (z_1, z_1 + \epsilon)$. Since $\bar{R}_{s+1}(-l_s) = 0$, $d(z)$ must have a double zero at z_1 or vanish at least once in $(-l_s, z_1)$. In both cases, counting the zeros of $d(z)$ outside of the origin leads to a contradiction. □

4. Description of ROCK4.

We implemented the numerical method described in section 3 in a code called ROCK4 for Orthogonal-Runge–Kutta–Chebyshev (appropriately permuted). This is the fourth order version of the code ROCK2 introduced in [2]. In this section we briefly describe the code.

Step size estimation. As in ROCK2, we implemented the "step size strategy with memory" of Watts [16] and Gustafsson [4],

$$h_{\text{new}} = \text{fac} \cdot h_n \left(\frac{1}{\text{err}_{n+1}}\right)^{\frac{1}{4}} \frac{h_n}{h_{n-1}} \left(\frac{\text{err}_n}{\text{err}_{n+1}}\right)^{\frac{1}{4}}, \tag{4.1}$$

in order to allow the step size to decrease reasonably without rejection (see also [7, p. 124]).

Stage number selection. While most stiff codes have a fixed number of stages, we used, as usual in Chebyshev codes, a family of fourth order methods. At each step we first select a step size in order to control the local error; then we select a stage order so that the stability property (see section 2 and Table 2.1)

$$h\rho\left(\frac{\partial f}{\partial y}(y)\right) \le 0.35s^2 \tag{4.2}$$

is satisfied, where $\rho(\dots)$ denotes the spectral radius of the Jacobian matrix of the ODEs. This is possible because for practical purposes, the error constants of the family of fourth order methods are found to be almost the same. They are close to the error constants of optimal stability polynomials which have been described in [1].

Spectral radius estimate. The user can supply a function which estimates a bound for the spectral radius (for example, by using Gershgorin theorem; see [6, p. 89]). By specifying that the Jacobian is constant, this function will be called only once. If it is not possible to get an estimate of the spectral radius easily, the code ROCK4 can also compute it internally. For that, we have implemented with a slight modification a nonlinear power method proposed by Sommeijer, Shampine, and Verwer (see [14]).

Storage. Due to the three-term recurrence formula, the method requires only a few storage vectors. The number of storage vectors does not depend on the number of stages used. For the computation of an integration step and the error estimation, ROCK4 uses three vectors for the recurrence formula and five additional vectors for the finishing four-stage method and the embedded method. Notice that the low memory demand of these methods is suitable since we want to apply them to large problems.

5. Numerical experiments. We conclude this paper with several stiff problems taken from the test set of stiff problems proposed in [7] (first and second editions). All the parameters chosen for the examples are taken from [7]. We compare the following codes:

ROCK4: the fourth order code described in this paper.

ROCK2: the second order code based on orthogonal polynomials and described in [2].

RKC: the second order Chebyshev code of Sommeijer, Shampine, and Verwer (see [14]).

RADAU5: the well-known implicit code by Hairer and Wanner of order 5 based on a Radau IIA collocation method (see [7]).

For all examples which follow, we compared the obtained numerical results for the different codes with a reference solution for the given ODEs. The computing time is then displayed as a function of the error (in an Euclidian norm). For each problem the codes have been applied with different tolerances, say

$$tol = 10^{-2}, 10^{-2-\frac{1}{4}}, 10^{-2-\frac{1}{2}}, \dots . \tag{5.1}$$

The integer-exponent tolerances are displayed as enlarged symbols. The results were computed with scalar tolerances $atol = rtol = tol$ for all problems. The symbol for $tol = 10^{-5}$ is distinguished by its gray color.

The following examples are parabolic PDEs, discretized by the method of line into a system of ODEs. We replace the second order spatial derivatives by the finite difference scheme

$$\frac{\partial^2 u(x_i, y_j, t)}{\partial x^2} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{(\Delta x)^2} + \mathcal{O}\big((\Delta x)^2\big),$$

where u_{ij} are functions depending on time.

Example 1. The first example is the Burgers' equation

$$u_t + \left(\frac{u^2}{2}\right)_x = \mu u_{xx}, \tag{5.2}$$

with initial condition

$$u(x,0) = 1.5x(1-x)^2, \tag{5.3}$$

and boundary conditions

$$u(0,t) = u(1,t) = 0 \tag{5.4}$$

for $0 \le x \le 1$ and $0 \le t \le 2.5$. We discretize the space variable of (5.2) by the method of lines with $\Delta x = \frac{1}{501}$, and we choose $\mu = 0.0003$.

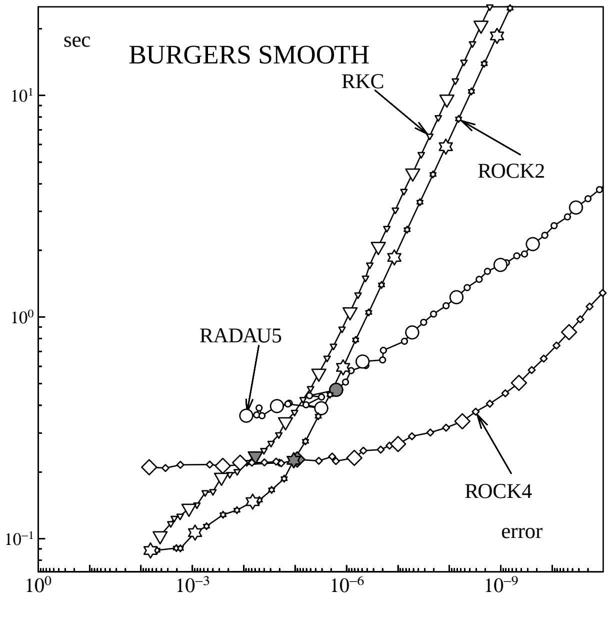


FIG. 5.1. *Work-precision diagram for Burgers' equations.*

Thus, we obtain an ODE (in time) of dimension 500. It is then solved by the different codes for $0 \le t \le 2.5$. For RADAU5 we used the banded algebra option, and for the Chebyshev codes we provide an estimation of the spectral radius by applying the Gershgorin theorem.

We see in Figure 5.1 that the two lower order methods, RKC and ROCK2, are better for lower tolerances. ROCK2 is slightly more efficient and better at delivering an accuracy close to the tolerance. For higher tolerances the high order codes RADAU5 and ROCK4 are better, with an advantage for ROCK4. This latter code also nicely preserves the tolerance proportionality.

Example 2. The second example is the two-dimensional Brusselator reaction-diffusion problem

$$\begin{aligned} \frac{\partial u}{\partial t} &= 1 + u^2 v - 4.4u + \alpha\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + f(x,y,t), \\ \frac{\partial v}{\partial t} &= 3.4u - u^2 v + \alpha\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right), \end{aligned} \tag{5.5}$$

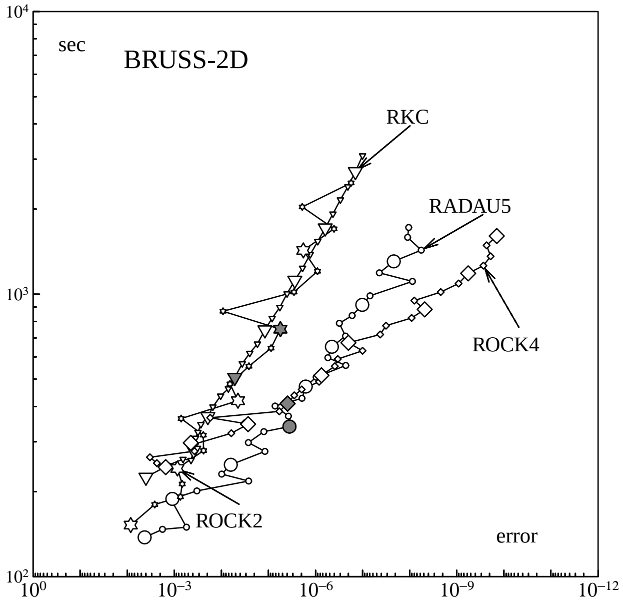


FIG. 5.2. *Work-precision diagram for the two-dimensional Brusselator problem.*

with initial conditions

$$u(x,y,0) = 22 \cdot y(1-y)^{3/2}, \qquad v(x,y,0) = 27 \cdot x(1-x)^{3/2},$$

and periodic boundary conditions

$$u(x+1,y,t) = u(x,y,t), \qquad u(x,y+1,t) = u(x,y,t)$$

for $0 \leq x \leq 1,\ 0 \leq y \leq 1,\ t \geq 0$. The function f is defined by

$$f(x,y,t) = \begin{cases} 5 & \text{if } (x-0.3)^2 + (y-0.6)^2 \leq 0.1^2 \text{ and } t \geq 1.1, \\ 0 & \text{else.} \end{cases}$$

We discretize the space variables of equations (5.5) with $x_i = \frac{i}{N+1}$, $y_i = \frac{i}{N+1}$, $i = 1,2,\dots,N$ and choose $N = 128$ and $\alpha = 0.1$. Thus, we obtain a system of $2N^2 = 32768$ equations. We chose the output points $t_{\text{out}} = 1.5$ and 11.5. The spectral radius of the Jacobian $\rho \simeq 13200$ can be estimated with the Gershgorin theorem; thus as in the previous example, we provide a bound for it when using Chebyshev methods. As advised in [7, p. 157] the linear equations in the code RADAU5 are solved by FFT methods so that the code is optimized for this problem. (Otherwise it will certainly not be competitive with Chebyshev methods.)

We see in Figure 5.2 that ROCK2 and RKC behaves similarly. For higher order methods, RADAU5 behaves better for low tolerances, while ROCK4 is better for higher tolerances. Between Chebyshev codes, except for very low tolerances ROCK4 gives the best results and nicely preserves the tolerance proportionality (as do ROCK2 and RADAU5).

Example 3. The third example is the FitzHugh and Nagumo model for explaining

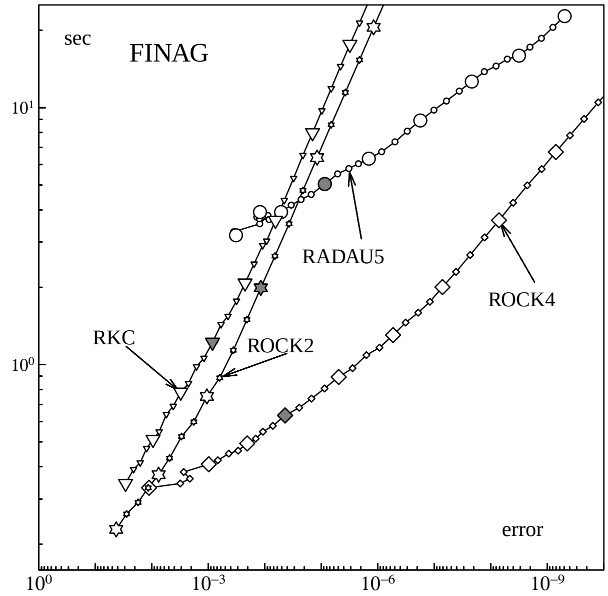


FIG. 5.3. *Work-precision diagram for FitzHugh and Nagumo equations.*

the nerve conduction as a traveling wave:

$$\begin{aligned} \frac{\partial u}{\partial t} &= \frac{\partial^2 u}{\partial x^2} - f(u) - v, \\ \frac{\partial v}{\partial t} &= \eta(u - \beta v), \end{aligned} \tag{5.6}$$

where

$$f(u) = u(u-\alpha)(u-1),$$

with initial conditions

$$u(x,0) = v(x,0) = 0,$$

and boundary conditions

$$\frac{\partial u}{\partial x}(0,t) = -0.3, \qquad \frac{\partial u}{\partial x}(100,t) = 0$$

for $0 \leq x \leq 100$ and $0 \leq t \leq 400$.

We chose $\alpha = 0.139$, $\eta = 0.008$, and $\beta = 2.54$ and discretize the space variable in 200 equidistant steps $x_i = \frac{2i+1}{4}$, $i = 0, \ldots, 199$. We compute numerically a bound for the spectral radius of the Jacobian which was used for the Chebyshev methods. This is a mildly stiff problem, and we see in Figure 5.3 the advantage of the Chebyshev methods compared to an implicit one. Again, ROCK2 works slightly better than RKC. ROCK4 behaves well and is better compared to other Chebyshev methods even for low tolerances.

Conclusion. We have presented a new fourth order Chebyshev method and implemented it in a code called ROCK4. The numerical examples show a good behavior of this code for problems it is intended for. We emphasize that this code, as does other Chebyshev codes, is very simple to use. In fact, it is as simple to use as the forward Euler method. The first version of this code (as well as ROCK2) and some examples are available on the Internet at the address http://www.unige.ch/math/folks/hairer/software.html. Experiences with this code are welcome.

Acknowledgments. The author is grateful to Ernst Hairer and Gerhard Wanner for helpful discussions. He also thanks Alexei Medovikov for his constant interest in this subject.

REFERENCES

[1] A. Abdulle, *On roots and error constant of optimal stability polynomials*, BIT, 40 (2000), pp. 177–182.

[2] A. Abdulle and A.A. Medovikov, *Second order Chebyshev methods based on orthogonal polynomials*, Numer. Math., 90 (2001), pp. 1–18.

[3] J.C. Butcher, *The Numerial Analysis of Ordinary Differential Equations. Runge-Kutta and General Linear Methods*, John Wiley and Sons, Chichester, UK, 1987.

[4] K. Gustafsson, *Control-theoretic techniques for stepsize selection in implicit Runge-Kutta methods*, ACM Trans. Math. Software, 20 (1994), pp. 496–517.

[5] E. Hairer, C. Lubich, and G. Wanner, *Geometric Numerical Integration*, Springer-Verlag, Berlin, Heidelberg, New York, 2002.

[6] E. Hairer, S.P. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations* I. *Nonstiff Problems*, 2nd ed., Springer Ser. Comput. Math. 8, Springer-Verlag, Berlin, 1993.

[7] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations* II. *Stiff and Differential-Algebraic Problems*, 2nd ed., Springer Ser. Comput. Math. 14, Springer-Verlag, Berlin, 1996.

[8] P.J. van der Houwen, *Construction of Integration Formulas for Initial Value Problems*, North-Holland, Amsterdam, 1977.

[9] P.J van der Houwen and B.P. Sommeijer, *On the internal stability of explicit m-stage Runge-Kutta methods for large m-values*, Z. Angew. Math. Mech., 60 (1980), pp. 479–485.

[10] A.A. Medovikov, *High order explicit methods for parabolic equations*, BIT, 38 (1998), pp. 372–390.

[11] V.I. Lebedev, *A new method for determining the zeros of polynomials of least deviation on a segment with weight and subject to additional conditions. Part* I, Russian J. Numer. Anal. Math. Modelling., 8 (1993), pp. 195–222.

[12] V.I. Lebedev, *A new method for determining the zeros of polynomials of least deviation on a segment with weight and subject to additional conditions. Part* II, Russian J. Numer. Anal. Math. Modelling., 8 (1993), pp.397–426.

[13] W. Riha, *Optimal stability polynomials,* Computing, 9 (1972), pp. 37–43.

[14] B.P. Sommeijer, L.F. Shampine, and J.G. Verwer, *RKC: An explicit solver for parabolic PDEs*, J. Comput. Appl. Math., 88 (1998) pp. 316–326.

[15] J.W. Verwer, *Explicit Runge-Kutta methods for parabolic partial differential equations*, Appl. Numer. Math., 22 (1996), pp. 359–379.

[16] H.A. Watts, *Step size control in ordinary differential equation solvers*, Trans. Soc. Computer Simulation, 1 (1984) pp. 15–25.

SIAM J. Sci. Comput.
Vol. 23, No. 6, pp. 2055–2073

HAMILTON–JACOBI EQUATIONS ON A MANIFOLD AND APPLICATIONS TO GRID GENERATION OR REFINEMENT*

PH. HOCH[†] AND M. RASCLE[‡]

Abstract. In this paper, we consider Hamilton–Jacobi equations on a manifold, typically on the graph of some previously computed function $z(x, y)$, and we show how the corresponding level set method allows us to generate and/or to refine a mesh in regions where this function z has large derivatives. Such as it is, the method needs to be strongly improved and accelerated, but the principle is awfully natural, and in principle the method is fully automatic. Similar ideas are also useful in image processing, in particular for the active contours method.

Key words. Hamilton–Jacobi equations, viscosity solutions, level set method, anisotropic mesh generation, high order numerical two dimensional schemes

AMS subject classifications. 65M50, 70H20, 53C22, 49L25

PII. S1064827599360182

1. Introduction.

The method we are presenting here is fairly general. However, for concreteness, we restrict ourselves to the following prototype: we consider in $\mathbb{R}^3$ a manifold Σ, which we assume to be the graph of a smooth function z: $\Omega \to \mathbb{R}$:, $\Sigma := \{(x, z(x)); x = (x_1, x_2) \in \bar{\Omega}\}$, where Ω is a bounded domain in $\mathbb{R}^2$. The function z is assumed to be known and smooth, but its derivatives can be very large in some regions of $\bar{\Omega}$.

The purpose of this paper is to show—at least in principle—how to generate a mesh which is automatically refined in regions where the graph of z—in other words, the manifold Σ—is "steep" (in a sense to be precised later). Practically, z is the (previously computed) numerical solution of some unspecified PDE on the domain Ω. If no such underlying function z is available, then we can always choose $z \equiv 0$, in which case we will just deal with the classical eikonal equation.

In this paper, we assume that z and its derivatives are known analytically at every point of $\bar{\Omega}$. Of course, in realistic applications, z is interpolated from its values at points of the mesh on which it has been computed, say by a finite element or a finite volume method.

There exists much literature on grid generation and refinement algorithms. Among many others, let us mention, for instance, [25, 13, 29, 18, 22, 45] and [11, 40] in the case of finite differences methods.

The main idea that we propose here is to generate such an adapted grid by solving the Hamilton–Jacobi equation on the graph of z, i.e., on the manifold Σ, equipped with a suitable Riemannian metric. Indeed, it is well known that the classical eikonal equation

$$\partial_t \phi(x, t) + |\nabla \phi(x, t)| = 0 \tag{1.1}$$

*Received by the editors August 9, 1999; accepted for publication (in revised form) October 2, 2001; published electronically February 27, 2002. The authors acknowledge support from the EC TMR network HCL under contract ERB FMRX CT 96 0033.

http://www.siam.org/journals/sisc/23-6/36018.html

[†]INRIA Rocquencourt, Projet "ondes", Domaine Voluceau 78153 Le Chesnay Cedex, France (Philippe.Hoch@inria.fr, hoch@math.unice.fr).

[‡]Laboratoire de Mathématiques, U.M.R. C.N.R.S. n6621, Université de Nice, Parc Valrose B.P. 71, F06108 Nice Cedex 2, France (rascle@math.unice.fr).

is a powerful tool to move curves at unit speed in the (Euclidean) normal direction.

In order to generate a mesh which is refined in regions where z has large derivatives, it is perfectly natural to slow down the motion of curves in these regions. The most natural way to do that is to move curves on the graph of z, at a given speed, in the (Riemannian) normal direction with a suitable metric. In other words, the curves move with constant speed on a stiff mountain, but their horizontal projections move very slowly so that the resulting mesh is automatically refined in these regions.

Although the principle is very general, the two natural metrics that we essentially consider here are based on the positive definite quadratic forms associated to either the matrix

(i)

$$G_1(x) = \begin{pmatrix} 1+(\frac{\partial z}{\partial x_1})^2 & \frac{\partial z}{\partial x_1}\frac{\partial z}{\partial x_2} \\ \frac{\partial z}{\partial x_1}\frac{\partial z}{\partial x_2} & 1+(\frac{\partial z}{\partial x_2})^2 \end{pmatrix} \tag{1.2}$$

(ii) or to

$$\begin{aligned} &G_2(x) = I +^t H(z)(x)H(z)(x), \\ &\text{where } H(z)(x) := \left(\frac{\partial^2 z}{\partial x_i \partial x_j}\right)(x)\text{: Hessian matrix of } z. \end{aligned} \tag{1.3}$$

Note that G_1 is nothing but the restriction of the ambient Euclidean metric in $\mathbb{R}^3$ to the manifold Σ.

Let us briefly comment on these two particular (strongly anisotropic) choices.

With the first choice, in regions where $|\nabla z| \gg 1$, the Riemannian distance between two neighbor points A and B (see Figure 1) is much bigger than their Euclidean distance, except if vectors $\vec{AB}$ and ∇z are orthogonal (in other words if A and B are essentially on the same level curve of z). Indeed, the eigenvector of $G_1(x)$ associated to the larger eigenvalue $\lambda_2(x) = 1+|\nabla z(x)|^2$ is $\nabla z(x)$, whereas the eigenvector associated to $\lambda_1(x) = 1$ is $(\nabla z)^\perp(x)$.

Consequently, with this first choice, the motion will be much slower

1. in regions where the (Euclidean) norm $|\nabla z|$ is large, and
2. inside these regions in the direction of ∇z. For instance, locally, we can have as in Figure 1 $d^{G_1}(A,B) = d^{G_1}(A,C)$ if $\nabla z(A) \gg 1$. In contrast, an isotropic metric defined, for instance, by $G_3(x) = F(|\nabla z(x)|)I$ (see, e.g., [45]) would lead to $d^{G_3}(A,B) \gg d^{G_3}(A,C) \simeq d^{G_3}(A,B')$ which is in principle inappropriate in this context.

With the choice (1.3), the motion is slowed down in regions where the second order derivatives of z are large, regardless of the size of the gradient. The motivation for that is very natural (see, e.g., the above-mentioned references [25, 18, 13]). Indeed, for example, in the context of P1 finite element approximations, there is no need to refine a mesh in regions where the gradients are large, if the solution is essentially linear, like in region BC in Figure 2 (in the one dimensional case).

In such a case, the mesh should be refined only in intervals AB and CD. In the multidimensional case this simple remark is reflected in the error estimates for the finite element method (see again the above references). In particular, in two dimensions, these error estimates are optimal when the triangles are almost equilateral with respect to a metric like G_2.

Remark 1.1. The approach described here in the case of the scalar-valued function z of two variables is in principle completely general: z could be any vector or (tensor-)

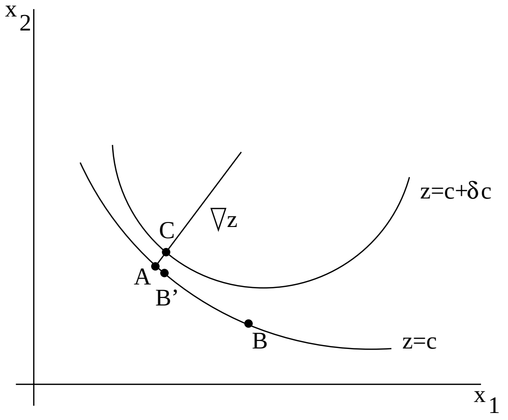


Fig. 1.

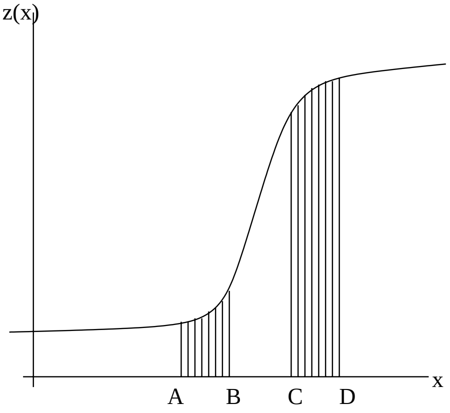


Fig. 2.

valued function defined on a domain in $\mathbb{R}^3$. This function z could even be defined on a two dimensional manifold $M \in \mathbb{R}^3$, say the wing of an airplane, and the goal is then to build a "good" mesh, automatically refined near singularities of the flow (transonic shocks, geometric singularities of the wing, etc.). In this view, the purpose would be to generate the mesh on the boundary and then to march in time in order to generate the fully three dimensional mesh.

With these ideas, our method turns out to be a variant of the classical level set methods [38, 42, 30, 36]. Naturally, quite similar problems arise in image processing [3], in particular in active contours (snakes) [17], etc. In the above-mentioned references, several authors deal with very similar problems (see, e.g., [30]), but in general the motivation is different. As far as we know, this very simple idea of generating a refined mesh by solving the Hamilton–Jacobi (HJ) equation on a manifold has never been explicitly expressed.

A last comment is that the results presented here are far from being optimal. As we will see later, there are still many imperfections in generating the grid. Our purpose here is to show only that this strategy is natural, and in principle tractable, even though the first version of the algorithm described here needs to be (strongly) improved.

The outline of the paper is as follows. In section 2, we recall a few basic facts on HJ equations. In sections 3 and 4, we briefly describe the numerical schemes that we have used, and we show in a few examples the level curves of the corresponding viscosity solution. In sections 5 and 6, we describe the principle of the construction of a triangular mesh based on these level curves, and we show a few numerical results. Finally, we recall the definition of viscosity solutions to the HJ equation in section 7.

2. The HJ equations on a manifold.

2.1. Basic facts. Although the method is fairly general, for simplicity we restrict ourselves to the typical situation described in section 1: Ω is a bounded domain in $\mathbb{R}^2$, and Σ is the graph of a smooth function z: $\Omega \to \mathbb{R}$, equipped with a Riemannian metric: at every point $(x, z(x))$ of Σ, we are given a positive definite symmetric bilinear form $g(x;.,.)$: in the coordinates system $x = (x^1, x^2)$, the length of dx is defined by

$$L(x, dx) := |dx|_{G(x)} = (dx, G(x)dx)^{\frac{1}{2}} = (g_{ij}(x)dx^i dx^j)^{\frac{1}{2}} = |G(x)^{\frac{1}{2}}dx| \tag{2.1}$$

with the Einstein summation convention. Here, $G(x) = (g_{ij}(x))_{1\leq i,j\leq 2}$, $(u,v) := \delta_{ij}u^iv^j$ is the Euclidean scalar product in $\mathbb{R}^2$, and $|u| = (u,u)^{\frac{1}{2}}$. The corresponding length of a curve $\Gamma := \{(\gamma(t), z(\gamma(t))), t \in [0,T]\}$ on Σ is then

$$L(\Gamma) = \int_0^T L(\gamma(t),\dot\gamma(t))dt := \int_0^T (\dot\gamma(t), G(\gamma(t))\dot\gamma(t))^{\frac{1}{2}}dt \tag{2.2}$$

Now, let $X(x) := (x, z(x))$ and $X(y)$ be given points on Σ. A (minimal) geodesic curve on Σ is a curve Γ for which $L(\Gamma)$ is extremal (minimal) among all curves on Σ with the same endpoints. Under standard assumptions, there exists a (unique) minimal C^1 geodesic curve between two arbitrary points $X(x)$ and $X(y)$ on Σ if these points are close enough (cf., e.g., [48]). Moreover, the "speed" $|G(\gamma(t))^{1/2}\dot\gamma(t)|$ is constant along a geodesic, and if the parameter t is the arc length, then minimizing $L(\Gamma)$ is equivalent to minimizing the energy

$$\mathrm{E}(\Gamma) := \int_0^T (L(\gamma(t),\dot\gamma(t)))^2dt \tag{2.3}$$

with the same boundary conditions (see, e.g., [21]). Here and often in the paper, we allow ourselves the confusion between curves on Σ and their horizontal projections in Ω.

2.2. The variational problem. Now, let $C_0 := \{(x, z(x)), x \in c_0\}$ be a curve on Σ, whose horizontal projection c_0 is a given (smooth) curve in $\bar\Omega$, parametrized, e.g., by its arc length s (practically, c_0 is the boundary of the domain Ω), let x be a nearby point in Ω, and let us consider the following problem:

$$\left\{\begin{array}{lcc} \text{Find } y \in c_0 \text{ such that} \quad d^G(x,y) & = & \text{Min }\left\{d^G(x,y'); y' \in c_0\right\} \\ & = & d^G(x,c_0), \end{array}\right. \tag{2.4}$$

where $d^G(x,y) := \text{Inf}\left\{\int_0^T L(\gamma(t),\dot\gamma(t))dt; \gamma(0) = y, \gamma(T) = x, \gamma(t) \in \bar\Omega\ \forall t \in [0,T]\right\}$.

Clearly (cf., e.g., [48]), for the case of geodesics with fixed endpoints, if c_0 is smooth and if $d(x,c_0)$ is small enough, then there exists a unique solution to (2.4), characterized by the Euler–Lagrange equations

$$-\frac{d}{dt}\left(\frac{\partial L}{\partial\dot\gamma}\right) + \frac{\partial L}{\partial\gamma} = 0, \tag{2.5}$$

$$\left(\frac{\partial L}{\partial\dot\gamma}(\gamma(0),\dot\gamma(0)), \tau\right) = 0, \tag{2.6}$$

where $\tau = \tau(y)$ is the tangent vector to c_0 at $y = \gamma(0)$. Again, we can restrict ourselves to considering only the horizontal projection

$$\frac{G(\gamma(0))\dot\gamma(0)}{|G(\gamma(0))\dot\gamma(0)|} = \pm\mathbf{n}, \tag{2.7}$$

where $\mathbf{n} = \mathbf{n}(\gamma(0))$ is one of the two possible unit normal vectors in $\Omega \subset \mathbb{R}^2$. Note that for the Euclidean metric in $\mathbb{R}^2$, the vector $\dot\gamma(0)$ is not normal to c_0, but in the particular case where $G = G_1$, this vector is the horizontal projection of the *normal geodesic* (cf., e.g., [42, 30])

$$\dot\Gamma(0) = \pm|\dot\Gamma(0)|\mathbf{T} \wedge \mathbf{N}. \tag{2.8}$$

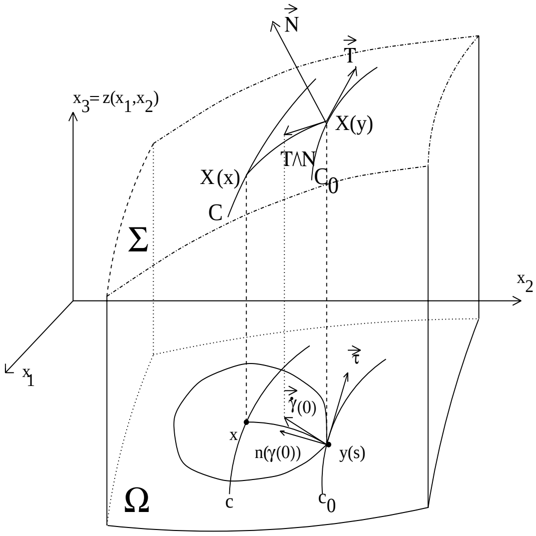


FIG. 3.

Naturally, in practical situations, the horizontal projection $\Omega \subset \mathbb{R}^2$ could be replaced by a two dimensional manifold $M \subset \mathbb{R}^3$, e.g., the wing of an airplane.

Geometrically, (2.6) says that the "ball" of center x and radius $d^G(x, y)$ is tangent to the curve c_0 at y; see Figure 3.

2.3. The HJ equations. Now, the purpose is to move the curve C_0 on Σ, or, equivalently, the curve c_0 in $\Omega \in \mathbb{R}^2$ at a given Riemannian speed V, by the level set method (see, e.g., [8, 38, 42, 26]). Dividing if necessary the matrix $G(x)$ by $(V(x))^2$, we restrict ourselves to the case $V \equiv 1$.

In other words, we define c_0 as the horizontal projection of the zero level curve of a function $\Phi(x, 0) = \Phi_0(x) : \Omega \to \mathbb{R}$, and we look for a function $(x, t) \mapsto \Phi(x, t)$ such that, for any $t \geq 0$,

$$c_t := \{x \in \Omega; \Phi(x,t) = 0\}\,. \tag{2.9}$$

A formal first order Taylor expansion gives

$$0 = \Phi(x,t) = \Phi(y,0) + t\partial_t\Phi(y,0) + \nabla\Phi(y,0).(x-y) + o(|t| + |x-y|). \tag{2.10}$$

Now, multiply both sides of (2.7) by $G^{-1}(y)$. Since $|\dot\gamma(0)|_G = 1$ and $V = 1$, we obtain at first order, after some manipulations,

$$x - y \;=\; t\,\dot\gamma(0) + o(t) = \pm\, t\,\frac{G^{-1}(y)\mathbf{n}}{|G^{-1}(y)\mathbf{n}|_G} + o(t) \;=\pm\, t\,\frac{G^{-1}(y)\nabla\Phi}{|G^{-1/2}(y)\nabla\Phi|} + o(t), \tag{2.11}$$

since the vector $\nabla\Phi$ is parallel to $\mathbf{n}$. Finally,

$$0 = \Phi(x,t) = 0 + t(\partial_t\Phi \pm (G^{-1}(y)\nabla\Phi, \nabla\Phi)^{\frac{1}{2}}) + o(t).$$

Therefore, the function Φ must satisfy

$$\partial_t\Phi \pm (G^{-1}(y)\nabla\Phi|\nabla\Phi)^{\frac{1}{2}} = \partial_t\Phi \pm |G^{-1/2}(y)\nabla\Phi| = 0. \tag{2.12}$$

Choosing the sign of the initial datum Φ_0 on each side of c_0 imposes the direction of propagation and therefore suppresses the above ambiguity. From now on, we choose the positive sign in (2.12). Therefore the Hamiltonian

$$H(x,p) := |G^{-1/2}(x)p| \tag{2.13}$$

is convex with respect to p.

2.4. The stationary problem. Now, Ω is a bounded domain in $\mathbb{R}^2$ with a (theoretically) smooth boundary $c_0 := \partial\Omega$. In fact, what we would like to compute is the family of level curves

$$\tilde{c_t} := \left\{x \in \Omega; d^G(x,\partial\Omega) = t, t \geq 0\right\} \tag{2.14}$$

(we will see below that curves $\tilde{c_t}$ and c_t are the same), or, equivalently, the family of curves

$$\tilde{C_t} := \{(x, z(x)); x \in \tilde{c_t}\}\,. \tag{2.15}$$

In the case of the Euclidean metric it is a classical result (see, e.g., [6, 32, 20]) that $\chi(x) := d(x,\partial\Omega)$ is a *viscosity solution* to the stationary equation

$$|\nabla\chi(x)| = 1 \qquad \text{in } \Omega \tag{2.16}$$

and satisfies in the classical sense the Dirichlet boundary condition

$$\chi(x) = 0 \qquad \text{on } \partial\Omega. \tag{2.17}$$

In fact, χ is the unique *viscosity solution* to (2.16) which satisfies (2.17). The definition of viscosity solutions are recalled below; see section 7. Here, the matrix $G(x)$ depends smoothly on x, and it is easy to adapt the above result to show the following proposition.

PROPOSITION 2.1. $\Psi(x) := d^G(x,\partial\Omega) := \inf_{y\in\partial\Omega} d^G(x,y)$ *is the unique viscosity solution to problem*

$$|G^{-1/2}(x)\nabla\Psi(x)| = 1 \qquad \text{in } \Omega, \tag{2.18}$$

which satisfies in the classical sense the boundary condition

$$\Psi(x) = 0 \qquad \text{on } \partial\Omega. \tag{2.19}$$

2.5. The initial boundary-value problem. Naturally, the stationary solution to (2.18), (2.19) is unknown (see, e.g., [41] for computational aspects). The principle of the *level set method* [10, 8, 38] is to compute the family of curves (c_t) defined by (2.14) with suitable initial data and boundary conditions. Indeed, the family of curves (c_t) depends on the initial curve $c_0 = \partial\Omega = \left\{x \in \bar{\Omega}; \Phi_0(x) = 0\right\}$ and *not* on the particular solution Φ to

$$\partial_t\Phi + |G^{-1/2}(x)\nabla\Phi| = 0 \qquad \text{in } Q = \Omega \times (0,T). \tag{2.20}$$

More precisely, we choose an initial datum Φ_0 such that

$$c_0 := \partial\Omega := \left\{x \in \bar{\Omega}; \Phi_0(x) = 0\right\}, \tag{2.21}$$

$$\Omega_0 := \Omega := \left\{x \in \bar{\Omega}; \Phi_0(x) > 0\right\}, \tag{2.22}$$

and we impose

$$|G^{-1/2}(x)\nabla\Phi_0(x)| = 1 \qquad \text{on } \partial\Omega. \tag{2.23}$$

The choice of "optimal" boundary data is a subtle question, which in particular raises in general the issue of discontinuous viscosity solutions (see again [6, 1, 12]). We recall that the family (c_t) is called "regular" if relations similar to (2.21), (2.22) between c_t and Ω_t hold for all $t > 0$; cf. [10, 8]. Here, using the independence of level curves c_t with respect to the particular solution Φ to (2.20), we follow the simplest choice:

$$\Phi(x,t) = -t \qquad \text{on } \partial\Omega \times (0,t). \tag{2.24}$$

The motivation for this choice is the compatibility with the obvious solution

$$\tilde{\Phi}(x,t) = \Psi(x) - t = d^G(x,\partial\Omega) - t \tag{2.25}$$

to problem (2.20), (2.24) with the initial data

$$\tilde{\Phi}(x,0) = \Psi(x), \tag{2.26}$$

which, of course, satisfies conditions (2.21), (2.22), (2.23). With this choice it is clear that curves c_t and $\tilde{c_t}$ are the same. Of course, solving the problem (2.20), (2.24), (2.26) is purely theoretical, since we do not know explicitly the function Ψ.

Therefore, we in fact solve problem (2.20), (2.24), (2.27), with

$$\Phi(x,0) = \Phi_0(x). \tag{2.27}$$

Adapting [32, Prop. 11.1] and standard results of uniqueness [6, 32], we obtain the following proposition.

Proposition 2.2. *For any Lipschitz function Φ_0 satisfying conditions* (2.21), (2.22), (2.23)*, the initial boundary value problem* (2.20), (2.24), (2.27) *has a unique viscosity solution Φ, whose level curves $\{c_t, t \geq 0\}$ do not depend on the particular choice of boundary conditions and initial data.*

Moreover, consider $\Omega_0 := \Omega$ and $\Omega_t := \{x \in \Omega; d^G(x,\partial\Omega) > t\}$; then Φ is in principle given by the Oleinik–Lax [35, 33] or Hopf [28, 5, 33, 7] type of formula:

$$\Phi(x,t) = \begin{cases} \operatorname{Inf}\{\Phi_0(y); d^G(y,x) \leq t\} & \text{if } x \in \bar{\Omega}_t, \\ d^G(x,\partial\Omega_0) - t & \text{if } x \in \bar{\Omega}\backslash\bar{\Omega}_t. \end{cases} \tag{2.28}$$

For related results with the classical eikonal equation, see, e.g., [7, 5, 33] and the references therein. We recall (see, e.g., [24]) that (2.23) would be necessary to define locally the *bicharacteristics* if the "initial" condition (2.19) for the stationary problem (2.18) is (only) given on curve $\partial\Omega$.

Remark 2.1. We remark here that, since Ω is bounded, the function Φ given by (2.28) converges in finite time to the stationary solution Ψ defined in Proposition 2.1 and obviously has the same level curves $\{x \in \Omega; d^G(x,\partial\Omega) = t\}$. This remark is the basis of the fast marching method developed in [49, 44], etc.

In principle, an adaptation of this fast marching method to this Riemannian anisotropic situation would be the fastest and most natural way to compute the successive positions of the front. However, this method has been classically developed in the case where the Hamiltonian H is a function of u^2 and v^2, where $(u,v) := \nabla\Phi$. This property, combined with the convexity of H, implies, e.g., $u\partial H/\partial u \geq 0$ [37], and a similar inequality for v, which is essential to build the monotone upwind approximation scheme described, e.g., in [46, 47]. In contrast, in the strongly anisotropic case considered here, this property is far from being satisfied, and the switching between "backward" and "forward" approximation has to be carefully modified.

Therefore, since our purpose in this paper was to introduce the *principle* of our Riemannian approach for grid generation or refinement, we did not consider this fast marching method in detail. For more recent references in this direction, we refer, e.g., to [37] or [43].

3. Numerical schemes. In view of the above remark, we want to approximate the initial boundary-value problem (2.20), (2.24), (2.27). We recall that (2.20) is hyperbolic in $\nabla\Phi = U = (u, v)$. We adopt the Osher–Shu approach [39]; in other words, we use a Cartesian grid $(x_i, y_j) = (i\Delta x, j\Delta y)$, and we first semidiscretize (2.20) to obtain

$$\frac{\partial}{\partial t}\Phi_{i,j}(t) = -h^{num}(x_i, y_j, u^-_{ij}, u^+_{ij}, v^-_{ij}, v^+_{ij}) \simeq -H(x_i, y_j, \nabla\Phi(x_i, y_j)), \tag{3.1}$$

where $U^{\pm}_{ij} := (u^{\pm}_{ij}, v^{\pm}_{ij})$ are approximations to $\nabla\Phi(x_i, y_j)$, and h^{num} is simply a two dimensional continuous, monotone numerical Hamiltonian. Let us describe more precisely the three classical steps in the algorithm.

3.1. The interpolation. At point (x_i, y_j), we first construct either an ENO or a TVD polynomial interpolation of Φ. For instance, a TVD interpolation of first order is

$$\begin{aligned} u^- &= D^{-x}_{ij} = \frac{\Phi_{i,j} - \Phi_{i-1,j}}{\Delta x}, & u^+ &= D^{+x}_{ij} = \frac{\Phi_{i+1,j} - \Phi_{i,j}}{\Delta x}, \\ v^- &= D^{-y}_{ij} = \frac{\Phi_{i,j} - \Phi_{i,j-1}}{\Delta y}, & v^+ &= D^{+y}_{ij} = \frac{\Phi_{i,j+1} - \Phi_{i,j}}{\Delta y}, \end{aligned}$$

or [38] a second order TVD interpolation with a minmod limiter is

$$\begin{aligned} u^- &= D^{-x}_{ij} + \frac{\Delta x}{2}\, m(D^{-x-x}_{ij}, D^{+x-x}_{ij}), & u^+ &= D^{+x}_{ij} - \frac{\Delta x}{2}\, m(D^{+x+x}_{ij}, D^{+x-x}_{ij}), \\ v^- &= D^{-y}_{ij} + \frac{\Delta y}{2}\, m(D^{-y-y}_{ij}, D^{+y-y}_{ij}), & v^+ &= D^{+y}_{ij} - \frac{\Delta y}{2}\, m(D^{+y+y}_{ij}, D^{+y-y}_{ij}). \end{aligned}$$

We have also used a second order ENO interpolation (see [39]) for the level curves of Figures 5A–5F.

3.2. Numerical Hamiltonian. Knowing the $(u, v)^{\pm}_{ij} = \nabla\Phi(x_i, y_j) + O(\Delta x^r)$ in regions where the solution is smooth, we want to approximate the ODE

$$\frac{\partial}{\partial t}\Phi_{i,j}(t) = -h^{num}(x_i, y_j, u^-, u^+, v^-, v^+),$$

where the numerical Hamiltonian h^{num} is a robust approximation of $H(x_i, y_j, \nabla\Phi_{ij})$. We choose the Lax–Friedrichs numerical Hamiltonian

$$\begin{aligned} h^{num}_{LF}(x, y, u^-, u^+, v^-, v^+) := H\left(x, y, \frac{u^- + u^+}{2}, \frac{v^- + v^+}{2}\right) \\ - \frac{\alpha}{2}(u^+ - u^-) - \frac{\beta}{2}(v^+ - v^-) \end{aligned} \tag{3.2}$$

with

$$\alpha = \sup_{u,v} \left| \frac{\partial H(x,y,u,v)}{\partial u} \right|, \qquad \beta = \sup_{u,v} \left| \frac{\partial H(x,y,u,v)}{\partial v} \right|.$$

We note that, e.g., in (1.2) or (1.3) the eigenvalues of G(x) are larger than 1. Therefore in (2.13), the eigenvalues of $G^{-1/2}$ are smaller than 1, so that it is easy in (3.2) to guarantee the CFL condition ($\Delta t(\frac{1}{\Delta x} + \frac{1}{\Delta y}) \leq 1$) uniformly in (x, y). However (see Hoch [27]), one can explicitly compute the optimized coefficients, $\alpha(x, y) = \sqrt{G^{-1}_{11}(x, y)}$ and $\beta(x, y) = \sqrt{G^{-1}_{22}(x, y)}$, based on the metric at point (x, y), for which this numerical Hamiltonian has the minimal diffusion and remains monotone.

3.3. Time discretization. We now discretize the ODE $\frac{\partial}{\partial t} U = L(U)$. Here, we have used the TVD Runge–Kutta second order method of Osher and Shu [39].

$$\tag{3.3} \begin{cases} U^{(0)} = U^n, \\ \\ U^{(i)} = \displaystyle\sum_{j=0}^{i-1} \alpha_{ij}\, U^{(j)} + \beta_{ij}\, \Delta t\, L(U^{(j)}), \qquad i = 1 \ldots s = 2, \\ \\ U^{n+1} = U^{(s)}. \end{cases}$$

We recall the principle of this approximation: if any spatial stability is satisfied for the Euler time approximation scheme, e.g., $TV(U^n + \Delta t L^x(U^n)) \leq TV(U^n)$, then the same stability holds for (3.3), modulo $C\Delta t$. An easy way to obtain such a scheme is to impose $\forall(i, j)$: $(\alpha_{ij}, \beta_{ij}) \geq 0$, since then (3.3) is a convex combination of Euler steps with Δt replaced by

$$\left(\min_{i,j} (\beta_{ij} \alpha^{-1}_{ij}) \right) \Delta t.$$

3.4. The boundary conditions. As in section 2, we impose the boundary condition (3.5), and for second order schemes we add near $\partial\Omega$ one layer of ghost points (x_i, y_j) out of Ω, at a distance of order Δx, and we define the values $\Phi^n_{i,j}$ at these ghost points at time $t_n = n\Delta t$ in order to satisfy a discrete approximation of (2.25). In fact, let $x = x_{i,j}$ be such a ghost point. We first define the signed distance

$$\tilde{\Psi}(x) := \begin{cases} \Psi(x) = d^G(x, \partial\Omega) & \text{in } \Omega, \\ -d^G(x, \partial\Omega) & \text{out of } \Omega. \end{cases}$$

Now, let y be (an approximation of) the normal projection of x on $\partial\Omega$:

$$x - y = -d(x, \partial\Omega)\mathbf{n}(y) = -d\mathbf{n}(y),$$

where d is the Euclidean distance. Since $\mathbf{n} = \mathbf{n}(y) = \frac{\nabla\Psi(y)}{|\nabla\Psi(y)|} = \frac{x-y}{|x-y|}$ is the inward normal vector to $\partial\Omega$, and since the Hamiltonian H in (2.13) is homogenous of order one, we have

$$|\nabla\Psi(y)| = (H(y, \mathbf{n}(y)))^{-1}.$$

We approximate $\Psi(x)$ by

$$\tag{3.4} \Psi_h(x) = \Psi(y) + \nabla\Psi(y).(x - y) = -d\,|\nabla\Psi(y)| = -\frac{d(x, \partial\Omega)}{H(y, \mathbf{n}(y))}.$$

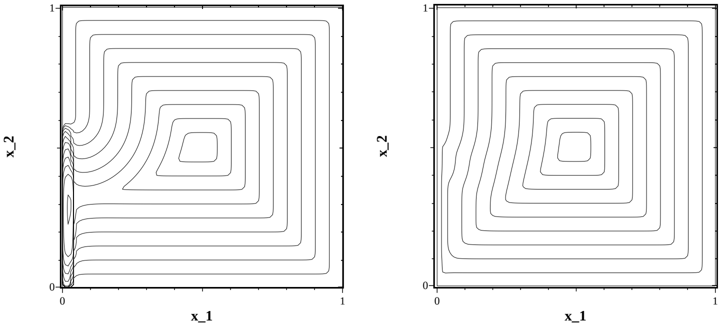


FIG. 4A AND 4B. *The role of the compatibility conditions* (3.6).

In view of (2.25) we define the approximate boundary data

$$\Phi_h(x,t_n) :=\Psi_h(x) - t_n. \tag{3.5}$$

3.5. The initial data. In the same spirit, we define the initial condition $\Phi(x,0)$ for x in Ω by the approximation

$$\Phi_h(x,0) := +\frac{d(x,\partial\Omega)}{H(y,\mathbf{n}(y))}, \tag{3.6}$$

where $d(x,\partial\Omega)$ is again the Euclidean distance of x to $\partial\Omega$, which in our case is easy to calculate. Theoretically, any other choice of the initial data satisfying (2.21), (2.22) would be permitted, but for x near $\partial\Omega$, formula (3.6) is a discretization of the *compatibility condition* (2.23) between the boundary condition (3.5) and the initial data. In order to illustrate the role of the denominator in (3.6), we have considered in Figures 4A and 4B the following example: Ω is the unit square, $G = G_1$ is given by (1.2), and $z(x) = \delta_{\epsilon_1}(x_1 - x_1^0)\delta_{\epsilon_2}(x_2 - x_2^0)$, with δ_ϵ defined in (4.4) below $0 < \epsilon_1 = 0.01 \ll \epsilon_2 = 0.1$, $x_1^0 = 0$, $x_2^0 = 0.25$. In other words, the graph of z mimics a boundary layer on a part of $\{x_1 = 0\}$. We have used the first order TVD (monotone) method described in section 3. We show in Figures 4A and 4B the level curves $\{x; \Phi(x,t_n) = 0\}$ for the same sequence, $t_n = Kn\Delta t$, $K \in \mathbb{N}$ fixed, of the solution when the metric $G(x)$ involves a boundary layer. In Figure 4B, where we have dropped the denominator in (3.6), the level curves almost do not detect the boundary layer.

4. Level curves: Results. Here, we show a few level curves $\{x; \Phi(x,t_n) = 0\}$ for a regular sequence of times $t_n = Kn\Delta t$. We recall that Φ is the unique viscosity solution to (2.20), (2.24), (2.27), computed with the algorithms discussed in section 3, and that our purpose is to *generate a mesh based on these level curves of* $\Phi(.,t_n)$, automatically refined in regions where z is stiff.

We have tested the above-mentioned algorithms on several examples:

(i) Ω is either a square or a circle. In the first case, note the shocks—sometimes smoothed out by the numerical viscosity—due to the corners of the square. Of course this difficulty disappears in the case of a circle.

(ii) The involved Riemannian metric is associated to the matrix: $G(x) = V^{-2}(x)G_i(x)$, $i = 1, 2$, so that the speed is multiplied by $V(x)$. Here, $V(x)$ is a given positive scalar speed, say $V(x) = \mathrm{Max}(h(x), V_{\min})$, adjusted to impose a (very small) minimal speed $V_{\min}$ in very stiff regions, and

$$h(x) = \begin{cases} h1(x) := (\det G(x))^{-1/2} & \text{or} \\ h2(x) := \exp(1 - (\det G(x))^{10}). \end{cases} \tag{4.1}$$

The presence of h still makes the motion slower in stiff regions. On the other hand,

$$G_1(x) = I + \nabla z \otimes \nabla z, \qquad G_2(x) = I + {}^tH(z)H(z). \tag{4.2}$$

(iii) This function z is a given (combination of) smooth real valued function(s), with large first and second order derivatives, which is analytically known. Typically (see Figures 5A–5D),

$$z(x) = z(x_1, x_2) = K_1 \, atan \left(K_2 f(x_1, x_2)\right), \tag{4.3}$$

where K_1 and/or K_2 are constants, adjusted to produce sharp fronts near the curve $\{f(x_1, x_2) = 0\}$.

In other cases (see Figures 5E, 5F), $z(x) = z(x_1, x_2)$ is proportional to a regularized delta-function

$$z(x) = \delta_\epsilon(|x - x_0|) = \frac{C}{\epsilon}\delta_1\left(\frac{|x - x_0|}{\epsilon}\right) \tag{4.4}$$

or a tensor-product $z = z_1 \otimes z_2$,

$$z_i(x) = C_i \delta_{\epsilon_i}(a_i x_1 + b_i x_2 + c_i); \tag{4.5}$$

see Figures 4A, 4B, 5D. In (4.4), $\delta_1(s) \geq 0$, $\int_{\mathbb{R}} \delta_1(s)ds = 1$, and $\delta_1(s) \equiv 0$ for $|s| \geq 1$.

The last two examples are chosen to simulate a sort of boundary layer or some oblique singularity near by the boundary.

(iv) In all cases, except in Figures 4A and 4B, we have used a second order ENO interpolation with a second order TVD time discretization. The CFL condition is 0.5, and $nx = ny = 101$. On the top of each figure, we have noted the corresponding metric G in (ii), the coefficient h in (4.1), and the minimum speed $V_{\min}$ in (ii).

A few comments are in order:

(i) Note the changes of topology of the level curves after having passed an obstacle, e.g., in Figure 4A, and Figures 5D–5F.

(ii) Note that the minimal speed is too small in Figure 5D, so that the level curves do not advance fast enough. This fact is a severe drawback if the purpose is to generate a mesh, as in section 5. On the contrary, it is an advantage if the purpose is to detect contours in image processing (snakes).

(iii) In pictures 5E and 5F, the involved function z is the same, given by (4.4), but Figures 5E and 5F, respectively, correspond to the metrics G_1 and G_2. As noted by one of the referees, here the metric G_1 (and still more the metric G_2) is stiff, so that the corresponding level curves motion is still slower, and therefore the location of the level curves is much better in the latter case. By the way, we have not systematically shown here a comparison between these two typical examples of metrics. Clearly, a numerical simulation of the one dimensional example of Figure 2 would strongly discriminate these two metrics.

5. Applications to grid generation. We want to generate a mesh whose vertices are located on successive level curves of the viscosity solution Φ to problem (2.20), (2.24), (3.5). The level curves of Φ allow us to generate a mesh as soon as

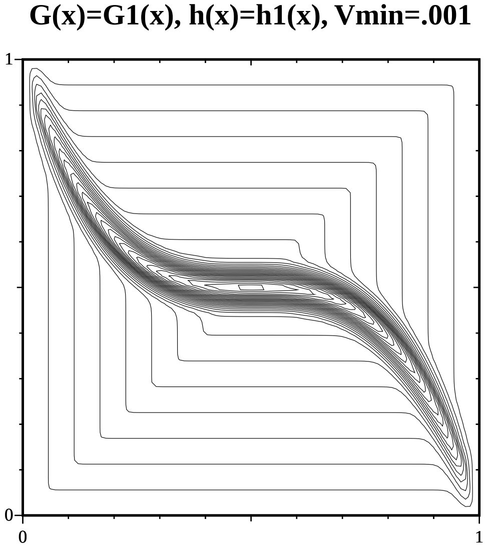


FIG. 5A.

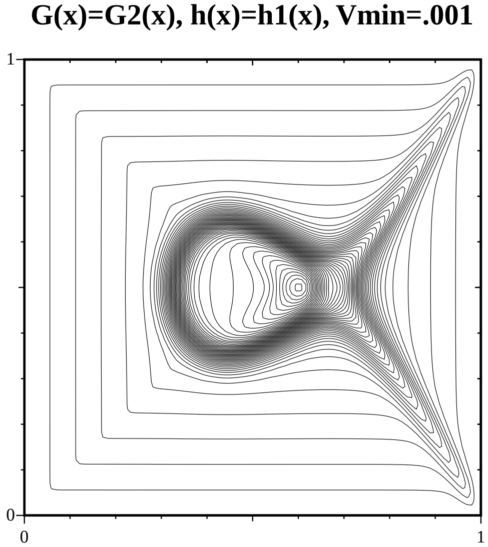


FIG. 5B.

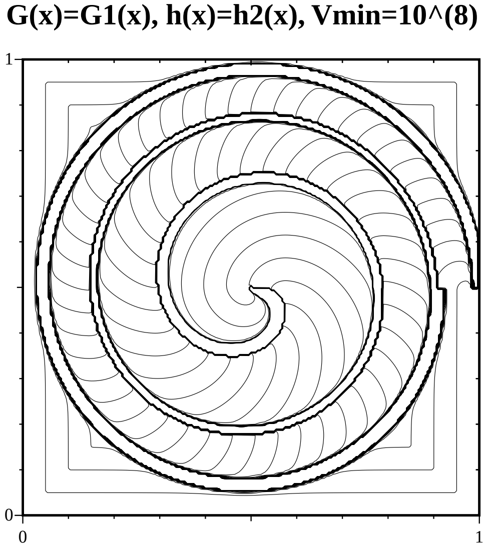


FIG. 5C.

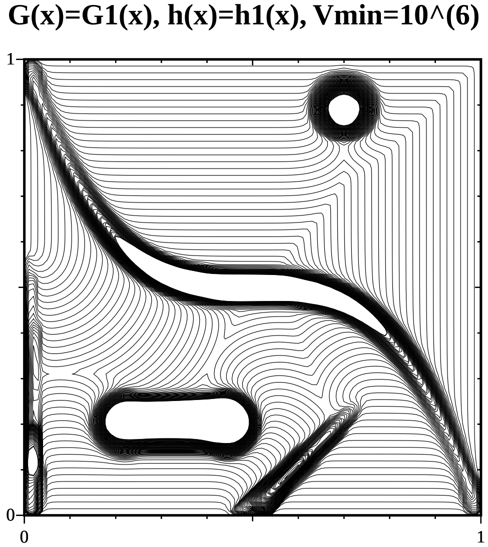


FIG. 5D.

we are able to connect points between two fronts. The elements that we generate are triangles or in some experiments mixed (quadrangles or triangles).

First, we recall that the ODE associated to (2.20) is

$$\frac{dx}{dt} = \frac{G^{-1}(x)\nabla\Phi(x,t)}{|G^{-1/2}(x)\,\nabla\Phi(x,t)|} = \nabla_p H(x,\nabla\Phi). \tag{5.1}$$

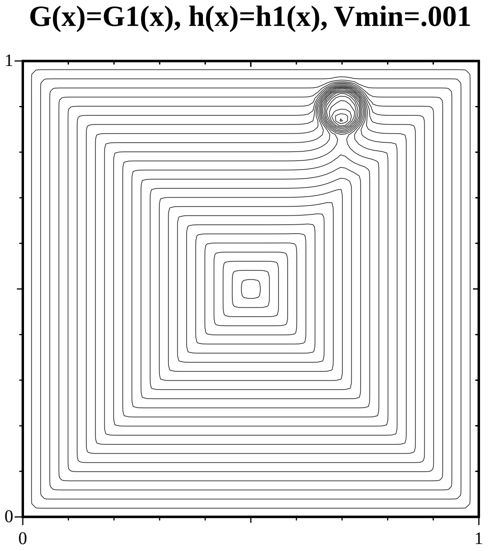

G(x)=G1(x), h(x)=h1(x), Vmin=.001

FIG. 5E.

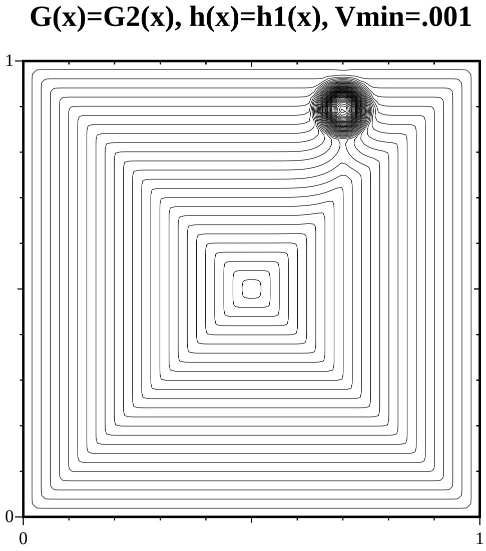

G(x)=G2(x), h(x)=h1(x), Vmin=.001

FIG. 5F.

These curves are the horizontal projections of the bicharacteristics, given by

$$\left\{ \begin{array}{l} \frac{dx}{dt} = \nabla_p H(x,p), \\ \frac{dp}{dt} = -\nabla_x H(x,p), \quad p = \nabla \Phi(x,t). \end{array} \right. \tag{5.2}$$

At each time step, having numerically solved (2.20) provides an approximation of $p = \nabla\Phi(x_i, y_j)$. We then interpolate p to locally solve numerically (5.1). We detect the appearance of *shocks* in step 5 below. We restrict ourselves to the description of the case of triangles.

5.1. Principle of the construction of a triangular mesh. The principle is the following:

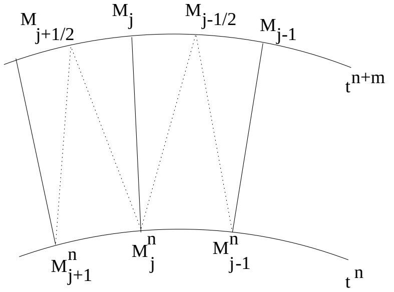
1. Define the domain Ω, and define Φ_0 by (3.6). In our numerical experiments, Ω is either a square or a circle.
2. Define a finite number of points $\{\tilde{M}_j^0, 1 \le j \le \tilde{N}_0\}$, with $\tilde{M}_{N_0}^0 = \tilde{M}_0^0$, compute an approximation (by the trapezoidal rule) of the Riemannian distance $d_j^0 := d^G(\tilde{M}_j^0, \tilde{M}_{j+1}^0)$, and replace the $\tilde{M}_j^0$ by new points M_j^0, which are approximately equidistant with respect to d^G. Their average Riemannian distance $\tilde{d^0}$ is thus near a desired reference length dgr.
3. n:=0.
4. Choose $m = m_n = [Kd^n/\Delta t]$, where $[x]$ is the integer part of x and K is a constant so that the average triangles will be "approximately" equilateral with respect to d^G. In the Euclidean case, we would choose $K = \sqrt{3}/2$.
5. For $k = 1 \ldots m$,
   (i) update Φ from t_{n+k-1} to t_{n+k}, using the algorithm defined in section 3;
   (ii) then update the points M_j according to the ODE (5.1);
   (iii) check the Riemannian distances $d_j^{n+k} := d^G(M_j^{n+k}, M_{j+1}^{n+k})$;
   - if $d_j^{n+k} < C_1 d^n$, then suppress one point: $M_j^{n+k} := (M_j^{n+k} + M_{j+1}^{n+k})/2$, $(C_1 = \frac{1}{\sqrt{2}})$;

- if $d_j^{n+k} > C_2 d^n$, then add one point: $M_{j+1/2}^{n+k}$ between M_j^{n+k} and M_{j+1}^{n+k}; $(C_2 = \sqrt{2})$;

(iv) if necessary, relabel points M_j^{n+k}.

6. At time t_{n+m},

(i) define $M_{j+1/2}^{n+m} \simeq$ the Riemannian midpoint between M_j^{n+m} and M_{j+1}^{n+m}; here we have used the approximation used in [25], $M_{j+1/2}^{n+m} = M_j^{n+m} + \alpha\ \mathbf{u_j^{n+m}}$, where we have noted $\mathbf{u_j^{n+m}} = \mathbf{M_j^{n+m} M_{j+1}^{n+m}}$ and $\alpha = \left(|\mathbf{u_j^{n+m}}|_{\mathbf{G(M_{j+1}^{n+m})}}\right)\left(|\mathbf{u_j^{n+m}}|_{\mathbf{G(M_{j+1}^{n+m})}} + |\mathbf{u_j^{n+m}}|_{\mathbf{G(M_j^{n+m})}}\right)^{-1}$;



(ii) project these new points $M_{j+\frac{1}{2}}^{n+m}$ on the level curves $\{\Phi(x, t^{n+m}) = 0\}$, using, say, two or three steps of the Newton method;

(iii) check whether the triangles $M_j^n M_{j+1}^n M_{j+1/2}^{n+m}$ and $M_{j-1/2}^{n+m} M_j^n M_{j+1/2}^{n+m}$ are approximately equilateral with respect to the distance d^G, by computing for a triangle ABC, $q = \frac{min(d^G(\mathbf{AB}),\mathbf{d^G(AC)},\mathbf{d^G(BC)})}{max(d^G(\mathbf{AB}),\mathbf{d^G(AC)},\mathbf{d^G(BC)})} \in]0,1]$. We also use the *edge flipping* technique (see, e.g., [25]): if necessary, we exchange the diagonals of a quadrangle, depending on the quality of the two resulting triangles.

If necessary, add or remove one point at time t_{n+m};

(iv) in any case, relabel the points $M_j^{n+m} := M_{j+1/2}^{n+m}$; evaluate the average Riemannian distance d^{n+m}.

7. In the case where the above iteration fails for some points M_j^{n+m}, freeze these points, use some "cuisine" to detect the underlying change of topologies in the level curves of $\Phi(., t_{n+m})$ and to generate the corresponding local part of the mesh.
8. Go back to step 4, with $n := n + m$, until all the points M_j^n are frozen.

Remark 5.1. The most difficult parts are steps 6(ii) and 7, which require a more detailed analysis, and will be addressed in a forthcoming work [4]. Note that such changes of topologies essentially appear in two types of situations:

(i) when there is a "pass" in the graph of the stationary viscosity solution $\Psi(x) = d^G(x, \partial\Omega)$ (see, e.g., Figures 5D–5F);

(ii) when this stationary solution Ψ presents a local maximum, i.e., when its level curves of level t are (locally) about to disappear, i.e., when the corresponding part of the mesh is *almost* finished. Not surprisingly, this step is one of the most difficult to handle, and should be handled by a local Delaunay type algorithm, adapted to the Riemannian metric (see, e.g., [25]).

5.2. Here we present some meshes. The reference length dgr and the other parameters are defined below or in section 5.1.

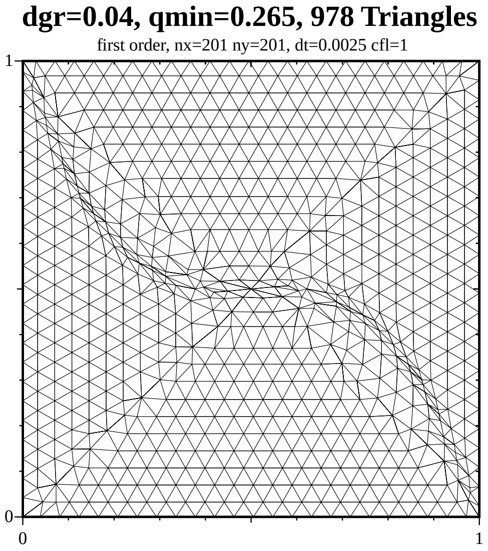

dgr=0.04, qmin=0.265, 978 Triangles
first order, nx=201 ny=201, dt=0.0025 cfl=1
1
0
0
1

Fig. 6A.

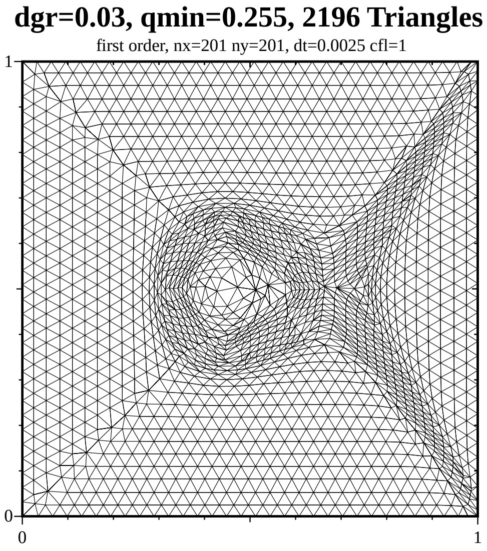

dgr=0.03, qmin=0.255, 2196 Triangles
first order, nx=201 ny=201, dt=0.0025 cfl=1
1
0
0
1

Fig. 6B.

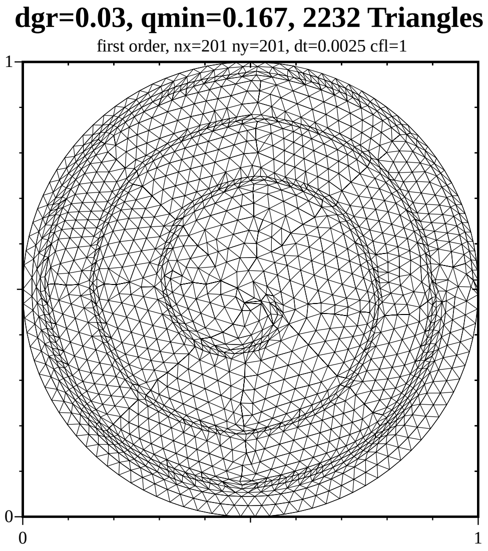

dgr=0.03, qmin=0.167, 2232 Triangles
first order, nx=201 ny=201, dt=0.0025 cfl=1
1
0
0
1

Fig. 6C.

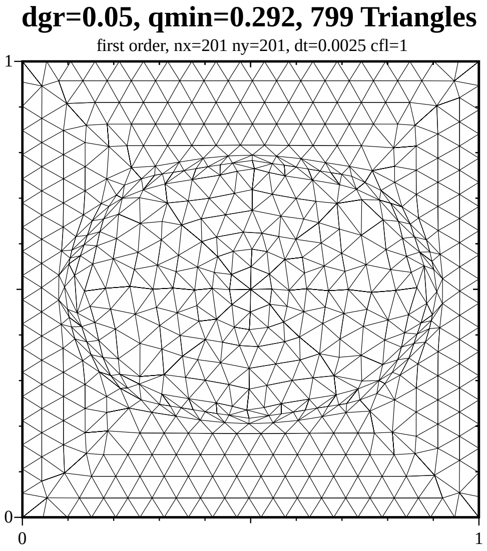

dgr=0.05, qmin=0.292, 799 Triangles
first order, nx=201 ny=201, dt=0.0025 cfl=1
1
0
0
1

Fig. 6D.

Comments and figure captions.

(i) All these simulations have been performed with $G(x) = G1(x) = I + \nabla z \otimes \nabla z$, and $V(x) \equiv 1$; we have also added a limitation on the coefficient of $G(x)$, i.e., on ∇z we have imposed $|z_x| < C$ and $|z_y| < C$, (C=2). In each case, dgr is defined in section 5.1, $qmin$ is the minimal quality of the triangles, nx and ny the number of grid points dt the time step, and cfl the courant number.

(ii) Figures 6A, 6B, and 6C, respectively, correspond to Figures 5A, 5B, and 5C. The latter are motivated, with a slightly different method, by [25]. In the case of Figure 6C, the underlying stiff function z is defined implicitly, which

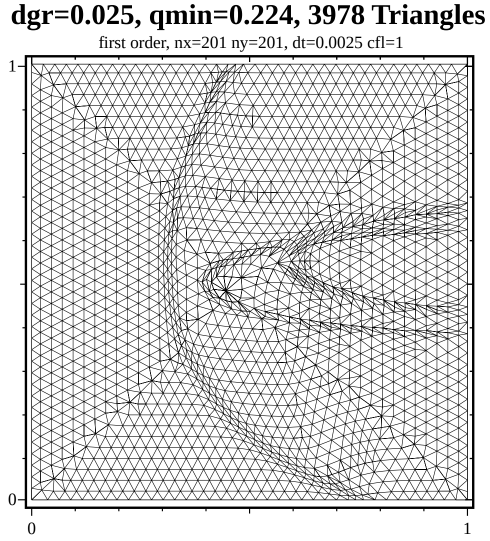

dgr=0.025, qmin=0.224, 3978 Triangles
first order, nx=201 ny=201, dt=0.0025 cfl=1
1
0
0
1

FIG. 6E.

creates some trouble near by $x_2 = 0.5$.

(iii) The "ellipse" in Figure 6D is defined by $f(x_1, x_2) := (x_1 - .5)^2 + 2 * (x_2 - .5)^2 - (0.4)^2 = 0$.

(iv) Figure 6E. Here, the metric is stiff near two ellipses and a parabola. This is a cartoon of a bow shock near a double ellipse. For computational simplicity, we have simply introduced as in section 4, a metric which is stiff near the involved curves without even removing the nodes inside the body! Nevertheless, the triangles look correctly refined.

6. Conclusion. In this paper we have introduced the simple and natural idea of moving curves with respect to the HJ equation on a manifold, or more precisely according to a given Riemannian metric, in order to generate and/or refine an anisotropic mesh. This idea is clearly in the line of the level set method, and several papers have used similar concepts, in a different context and with different motivations (see, e.g., [30, 42]), but as far as we know, this idea had never been explicitly stated in this way, especially in this context. As we said in the introduction, this work is a first attempt in this direction. There are still numerous problems to solve, and our algorithms need to be strongly improved. First, one should deal with the *stationary* problem (see [41, 16]) or more recently [27] with the numerical Hamiltonian of section 3.2 with the optimized coefficient $\alpha(x, y)$ and $\beta(x, y)$. Moreover, one should use an adapted version of the fast marching method [49, 2, 46], and more recently [37, 43]. As we said in Remark 2.1, the difficulty here is the strong anisotropy, so that the upwinding must be carefully performed.

Clearly, one of the main drawbacks of the present version is the *Lagrangian* feature of the mesh generation. Therefore, one should strongly improve steps 5, 6, and 7 in the mesh generation; in particular, one could combine step 7 with a local Delaunay algorithm based on a Riemannian metric, in the spirit of [25], and, still more important, generate the characteristics (5.2) in an *Eulerian* way, e.g., by following a suggestion of J.D. Benamou (Private Comm.) in connection with the correct numerical upwinding.

We address some of these questions in forthcoming works, e.g., in [4]. In principle, our *anisotropic* approach could have natural applications in image processing, in either image enhancement or edge detection. Indeed, some numerical results in [27] look qualitatively good, but not significantly better than other methods, at least on the examples that we have considered.

Finally, and we thank one of the referees for this question, let us mention that the approach developed here is independent from another Riemannian method, based on the theory of harmonic functions, developed for mesh generation in [23] (see also [15, 14, 31]), and independently used in image processing, e.g., in [47]. The latter theory amounts to solving a second order elliptic boundary value problem and is definitely a kind of (beautiful) multi-D generalization of the (one dimensional) theory of geodesics with, however, some severe restrictions of convexity, nonpositive curvature, etc. on the involved manifolds.

In contrast, the method presented here is a variant of the Eikonal equation, classically seen as a "curve mover," in the spirit of the Huygens principle. This equation is definitely a first order (hyperbolic) equation, with only one (upstream) boundary condition on each ray, starting on the boundary of the domain of interest, and moving until the propagating fronts have swept the whole domain. The method is in principle multi-D with no restriction on the shape of the domain. On the other hand, as we have seen, it still needs substantial numerical improvements before being competitive.

7. Appendix.

For convenience, we recall the definition of viscosity solution and refer, e.g., to [6, 19, 20] for more details. Let us consider the evolution equation

$$\partial_t u + H(x, \nabla u) = 0. \tag{7.1}$$

DEFINITION 7.1. *$u \in C(\Omega)$ is called a viscosity solution to* (7.1) *if u is both a subsolution and a supersolution: for all C^1 test function v, at any point (x_0, t_0) in $\Omega \times (0, t)$ where (u-v) has a local maximum (resp., minimum), then*

$$\partial_t v(x_0, t_0) + H(x_0, \nabla v(x_0, t_0)) \leq 0 \quad (resp., \geq 0). \tag{7.2}$$

The definition of viscosity solution in the stationary case is quite similar. As to the initial boundary value problem, in the general case the formulation involves *discontinuous* viscosity solutions, which take into account the possibility that sometimes boundary conditions cannot be satisfied in the classical sense on the whole boundary, e.g., in the case of the problem

$$|\phi'(x)| - 1 = 0 \text{ on } (0, 1), \;\; \phi(0) = 0\,, \; \phi(1) = 2.$$

In this paper, the solution Ψ to the stationary problem (2.18), (2.19), as well as the solution Φ to problem (2.20), (2.24), (3.6), satisfy the boundary condition in the classical sense.

REFERENCES

[1] R. ABGRALL, *Numerical discretization of the first-order Hamilton-Jacobi equation on triangular meshes*, Comm. Pure Appl. Math., 49 (1996), pp. 1339–1373.

[2] D. ADALSTEINSSON AND A. SETHIAN, *A fast level set method for propagating interfaces*, J. Comput. Phys., 118 (1995), pp. 269–277.

[3] L. ALVAREZ, P.L. LIONS, AND J.M. MOREL, *Image selective smoothing and edge detection by nonlinear diffusion* II, SIAM J. Numer. Anal., 29 (1992), pp. 845–866.

[4] P. BAGNERINI, P. HOCH, AND M. RASCLE, *The eikonal equation on a manifold. Applications to grid generation*, in Hyperbolic Problems: Theory, Numerics, Applications, Internat. Ser. Numer. Math. 140, Basel, Boston, Berlin, Birkhäuser, 2001.

[5] M. Bardi and L.C. Evans, *On Hopf's formulas for solutions of Hamilton-Jacobi equations*, Nonlinear Anal., 8 (1984), pp. 1373–1381.

[6] G. Barles, *Solutions de viscosité des équations de Hamilton-Jacobi*, Springer-Verlag, Paris, 1994.

[7] G. Barles, *Uniqueness for first-order Hamilton-Jacobi equations and Hopf formula*, J. Differential Equations, 69 (1987), pp. 346–367.

[8] G. Barles and P.E. Souganidis, *A new approach to front propagation problems: Theory and applications*, Arch. Rational Mech. Anal., 141 (1998), pp. 237–296.

[9] G. Barles and P.E. Souganidis, *Convergence of approximation schemes for fully nonlinear second order equations*, Asymptot. Anal., 4 (1991), pp. 271–283.

[10] G. Barles, H.M. Soner, and P.E. Souganidis, *Front propagation and phase field theory*, SIAM J. Control Optim., 31 (1993), pp. 439–469.

[11] J. Bell, M. Berger, J. Saltzman, and M. Welcome, *Three-dimensional adaptative mesh refinement for hyperbolic conservation laws*, SIAM J. Sci. Comput., 15 (1994), pp. 127–138.

[12] J.D. Benamou, *Multivalued Solution and Viscosity Solutions of the Eikonal Equation*, rapport de recherche 3281, INRIA Rocquencourt, 1997.

[13] H. Borouchaki, P.L. George, F. Hecht, P. Laug, B. Mohammadi, and E. Saltel, *Mailleur bidimensionnel de Delaunay gouverné par une carte de métriques. Partie* II*: Applications*, rapport de recherche 2760, INRIA Rocquencourt, 1995.

[14] J.U. Brackbill, *An adaptative grid with direction control*, J. Comput. Phys., 108 (1993), pp. 38–50.

[15] J.U. Brackbill and J.S. Saltzman, *Adaptative zoning for singular problems in two dimensions*, J. Comput. Phys., 46 (1982), pp. 342–368.

[16] P. Cardaliaguet, P. Saint-Pierre, and M. Quincampoix, *Set-valued numerical analysis for optimal control and differential games*, in Stochastic and Differential Games, M. Bardi, T.E.S. Raghavan, and T. Parthasarathy, eds., Birkhauser, Boston, 1999, pp. 177–247.

[17] V. Caselles, R. Kimmel, and G. Sapiro, *Geodesic active contours*, Intern. J. Comp. Vision, 22 (1997), pp. 61–79.

[18] M.J. Castro Diaz, F. Hecht, B. Mohammadi, and O. Pironneau, *Anisotropic unstructured mesh adaptation for flows simulations*, Internat. J. Numer. Methods Fluids, 25 (1997), pp. 475–491.

[19] M. Crandall, H. Ishii, and P.L. Lions, *User's guide to viscosity solutions of second order partial equations*, Bull. Amer. Math. Soc. (N.S.), 27 (1992), pp. 1–67.

[20] M. Crandall and P.L. Lions, *Viscosity solutions of Hamilton-Jacobi equations*, Trans. Amer. Math. Soc., 277 (1983), pp. 1–42.

[21] M. Demazure, *Calcul différentiel*, Cours de l'École Polytechnique, Palaiseau, France, 1979.

[22] J.A. Désidéri, *Control of hyperbolic grid-generation systems*, Proc. of the Ninth Int. Conf. on Finite Elements in Fluids, New Trends and Applications, Venice, Italy, October 1995, O. Zienkiewicz, ed., Univ. di Padova Publ.

[23] A.S. Dvinsky, *Adaptative grid generation from harmonic maps on Riemannian manifolds*, J. Comput. Phys., 95 (1991), pp. 450–476.

[24] J. Fritz, *Partial Differential Equations*, Springer-Verlag, New York, 1982.

[25] P.L. George and H. Borouchaki, *Triangulation de Delaunay et maillage, applications aux éléments finis*, Hermès, Paris, 1999.

[26] E. Harabetian and S. Osher, *Regularization of ill-posed problems via the level set approach*, SIAM J. Appl. Math, 58 (1998), pp. 1689–1706.

[27] P. Hoch, *Approximation of Non-linear Hyperbolic Problems, Hamilton-Jacobi Equations and Applications*, Ph.D. Thesis, Université Nice-Sophia Antipolis, 2000.

[28] E. Hopf, *Generalized solutions of non-linear equations of first-order*, J. Math. Mech., 14 (1965), pp. 951–973.

[29] C. Johnson and A. Szepessy, *Adaptative finite element methods for conservation laws based on a posteriori error estimates*, Comm. Pure Appl. Math., 98 (1995), pp. 199–234.

[30] R. Kimmel, Curve Evolution on Surfaces, Ph.D. Thesis, University of California, Berkeley, CA, 1995.

[31] R. Li, T. Tang, and P.W. Zhang, *Moving mesh methods in multiple dimensions based on harmonic maps*, J. Comput. Phys., 170 (2001), pp. 562–568.

[32] P.L. Lions, *Generalized Solutions of Hamilton-Jacobi Equations*, Pitman, Boston, 1982.

[33] P.L. Lions and J.C. Rochet, *Hopf formula and multitime Hamilton-Jacobi equations*, Proc. Amer. Math. Soc., 96 (1986), pp. 79–84.

[34] R.B. Milne, *An Adaptative Level Set Method*, Ph.D. Thesis, University of California, Berkeley, CA, 1995.

[35] O.A. Oleinik, *Discontinuous solutions of non-linear differential equations*, Amer. Math. Soc.

Transl., 26 (1963), pp. 95–172.

[36] S. Osher, *A level set formulation for the solution of the Dirichlet problem for Hamilton–Jacobi equations*, SIAM J. Math. Anal., 24 (1993), pp. 1145–1152.

[37] S. Osher and R.P. Fedkiw, *Level set methods: An overview and some recent results*, J. Comput. Phys., 169 (2001), pp. 463–502.

[38] S. Osher and J.A. Sethian, *Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations*, J. Comput. Phys., 79 (1988), pp. 12–49.

[39] S. Osher and C.-W. Shu, *High-order essentially nonoscillatory schemes for Hamilton–Jacobi equations*, SIAM J. Numer. Anal., 28 (1991), pp. 907–922.

[40] R. Pember, J. Bell, Ph. Colella, W. Crutchfield, and M. Welcome, *An adaptative cartesian grid method for unsteady compressible flow in irregular regions*, J. Comput. Phys., 120 (1995), pp. 278–304.

[41] E. Rouy and A. Tourin, *A viscosity solutions approach to shape-from-shading*, SIAM J. Numer. Anal., 29 (1992), pp. 867–884.

[42] J.A. Sethian, *Level Set Methods, Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision, and Materials Science*, Cambridge University Press, Cambridge, UK, 1996.

[43] J.A. Sethian, *Evolution, implementation, and application of level set and fast marching methods for advancing fronts*, J. Comput. Phys., 169 (2001), pp. 503–555.

[44] J.A. Sethian, *Level Set Methods and Fast Marching Methods, Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*, Cambridge University Press, Cambridge, UK, 1999.

[45] J.A. Sethian, *Curvature flow and entropy conditions applied to grid generation*, J. Comput. Phys., 115 (1994), pp. 440–454.

[46] J.A. Sethian, *A fast marching level set method for monotonically advancing fronts*, Proc. Nat. Acad. Sci. U.S.A., 93 (1996), pp. 1591–1595.

[47] N. Sochen, R. Kimmel, and R. Malladi, *A geometrical framework for low level vision*, IEEE Trans. Image Process., 7 (1998), pp. 310–318.

[48] M. Spivak, *A Comprehensive Introduction to Differential Geometry*, Vol. 1, Publish or Perish, Inc., Wilmington, DE, 1975.

[49] J.N. Tsitsiklis, *Efficient algorithms for globally optimal trajectories*, IEEE Trans. Automat. Control, 40 (1995), pp. 1528–1538.

SIAM J. SCI. COMPUT.
Vol. 23, No. 6, pp. 2074–2094

CHEBYSHEV SPECTRAL METHODS FOR RADIATIVE TRANSFER*

ARNOLD D. KIM[†] AND MIGUEL MOSCOSO[‡]

Abstract. We study the performance of Chebyshev spectral methods for time-dependent radiative transfer equations. Starting with a method for one-dimensional problems in homogeneous media, we show that the modifications needed to consider more general problems such as inhomogeneous media, polarization, and higher dimensions are straightforward. In this method, we approximate the spatial dependence of the intensity by an expansion of Chebyshev polynomials. This yields a coupled system of integro-differential equations for the expansion coefficients that depend on angle and time. Next, we approximate the integral operation on the angle variables using a Gaussian quadrature rule resulting in a coupled system of differential equations with respect to time. Using a second-order finite difference approximation, we discretize the time variable. We solve the resultant system of equations with an efficient algorithm that makes Chebyshev spectral methods competitive with other methods for radiative transfer equations.

Key words. Chebyshev spectral method, radiative transfer

AMS subject classifications. 65M70, 35Q60, 78A45, 78A48

PII. S1064827500382312

1. Introduction.

Spectral methods have been widely applied to the Navier–Stokes [1, 2], Schrödinger [3], acoustic [4], and Maxwell [5] equations, among others, with great success. In this paper we are interested in using a Chebyshev spectral method [6, 7] for the vector radiative transfer equation

$$\frac{1}{v}\frac{\partial}{\partial t}\mathbf{I}(\mathbf{r},\widehat{\Omega},t)+\widehat{\Omega}\cdot\nabla_{\mathbf{r}}\mathbf{I}(\mathbf{r},\widehat{\Omega},t)+\mathcal{Q}[\mathbf{I}](\mathbf{r},\widehat{\Omega},t)=\mathbf{F}(\mathbf{r},\widehat{\Omega},t) \tag{1.1}$$

governing wave propagation in a medium $\mathcal{D}\subset\mathbb{R}^n$ ($n=1,2,3$) that scatters, absorbs, depolarizes, and emits radiation. Applications for the vector radiative transfer equation include polarized light propagation in clouds, fog, rain, and biological tissue [8] as well as seismic wave propagation in heterogeneous media [10, 11]. In (1.1) $\mathbf{I}=(I,Q,U,V)$ is the 4×1 Stokes vector needed to describe the polarized radiation field completely. The total intensity is represented by I, the linear polarization state by Q and U, and the circular polarization state by V. The Stokes vector $\mathbf{I}$ depends on position $\mathbf{r}\in\mathbb{R}^n$, direction $\widehat{\Omega}\in\mathbb{S}^2$ ($\mathbb{S}^2$ denotes the surface of the unit sphere), and time $t\in[0,T]$. In (1.1), v is the constant wave speed in the medium, and

$$\mathcal{Q}[\mathbf{I}](\mathbf{r},\widehat{\Omega},t)=\sigma_t(\mathbf{r})\mathbf{I}(\mathbf{r},\widehat{\Omega},t)-\sigma_s(\mathbf{r})\int_{\mathbb{S}^2}\mathbf{S}(\widehat{\Omega},\widehat{\Omega}')\,\mathbf{I}(\mathbf{r},\widehat{\Omega}',t)\,d\widehat{\Omega}' \tag{1.2}$$

is the scattering operator. The total scattering cross-section $\sigma_t(\mathbf{r})$ is the sum of the scattering cross-section $\sigma_s(\mathbf{r})$ and the absorption cross-section $\sigma_a(\mathbf{r})$. All of these cross-sections are real and nonnegative. The scattering matrix $\mathbf{S}(\widehat{\Omega},\widehat{\Omega}')$, which we

*Received by the editors December 8, 2000; accepted for publication (in revised form) November 15, 2001; published electronically February 27, 2002.

http://www.siam.org/journals/sisc/23-6/38231.html

[†] Department of Mathematics, 450 Serra Mall, Building 380, Stanford University, Stanford, CA 94305-2125 (adkim@math.stanford.edu). This author's research was supported by NSF grant DMS-0071578.

[‡] Departamento de Matemáticas, Escuela Politécnica Superior, Universidad Carlos III de Madrid, Avda. de la Universidad 30, 28911 Leganés, Spain (mmoscoso@math.stanford.edu). This author's research was supported by AFOSR grant 49620-98-1-0211 and by NSF grant 9709320.

assume is position independent, describes the directional distribution of energy density that scatters in direction $\widehat{\Omega}$ due to unit energy density incident in direction $\widehat{\Omega}'$. In addition, this matrix describes polarization changes manifested by scattering. Finally, the source term $\mathbf{F}(\mathbf{r}, \widehat{\Omega}, t)$ accounts for any sources contained in the medium.

The radiative transfer equation has to be solved with appropriate initial and boundary conditions. We assume that no radiation other than the source $\mathbf{F}$ enters into the medium so that

$$\mathbf{I}(\mathbf{r}, \widehat{\Omega}, t) = 0 \quad \text{on} \quad \Gamma, \tag{1.3}$$

where $\Gamma = \{(\mathbf{r}, \widehat{\Omega}, t) \in \partial\mathcal{D} \times \mathbb{S}^2 \times [0, T] \ \text{ such that } \ \nu(\mathbf{r}) \cdot \widehat{\Omega} < 0\}$, and $\nu(\mathbf{r})$ denotes the unit outward normal vector to the boundary $\partial\mathcal{D}$. In addition, we assume that no energy is present in the medium at time $t = 0$,

$$\mathbf{I}(\mathbf{r}, \widehat{\Omega}, 0) = 0 \quad \text{in} \quad \mathcal{D} \times \mathbb{S}^2. \tag{1.4}$$

The transport problem (1.1)–(1.4) is well-posed [12]. It models the incoherent or scattered intensity for which its source manifests from coherent intensity incident at the boundary [8, 9].

Equation (1.1) is usually solved using Monte Carlo methods (see [13] and references therein for details). The main advantage of Monte Carlo methods is their relative simplicity and their ability to handle complicated geometries. For large optical depths, they require a large number of particles to obtain statistical accuracy leading to long computational times. Other common numerical methods such as finite differences and finite elements have been applied to the scalar radiative transfer equation with no polarization. However, for vector problems, Monte Carlo methods are preferred.

Despite the high accuracy and competitive cost of spectral methods, there is not, to our knowledge, any attempt to solve the time-dependent, vector radiative transfer equation using spectral methods. However, there are two works that are related to this problem. Ritchie, Dykema, and Braddy [14] solve the scalar, time-dependent problem radiative transfer equation in which polarization is neglected with a Fourier spectral method. Kim and Ishimaru [15] solve the one-dimensional, time-independent, vector radiative transfer equation in homogeneous media with Chebyshev spectral methods. In contrast to Fourier methods, which are restricted to problems with periodic boundary conditions, Chebyshev spectral methods can consider a broad variety of boundary conditions [7]. This is important for many applications such as optical imaging [16].

In this paper, we show that the underlying ideas of the Chebyshev spectral method shown in [15] are robust to generalizations such as time-dependent, inhomogeneous, and higher-dimensional problems. Keeping the ease of implementation and the low computational cost of a basic algorithm, we introduce modifications needed for these general problems. In section 2, we start with the basic algorithm for the one-dimensional, scalar problem in homogeneous media. We generalize this method to inhomogeneous, vector, and higher-dimensional problems in section 3. We present results from numerical experiments in section 4. A summary of our work and some concluding remarks appear in section 5.

2. Time-dependent, scalar problems in one-dimensional homogeneous media.

In this section, let us consider the plane-parallel problem shown in Figure 2.1. The medium, which infinitely extends in the x-y directions, is bounded by two planes located at $z = 0$ and $z = d$, and its spatial properties, σ_t and σ_s, are constant.

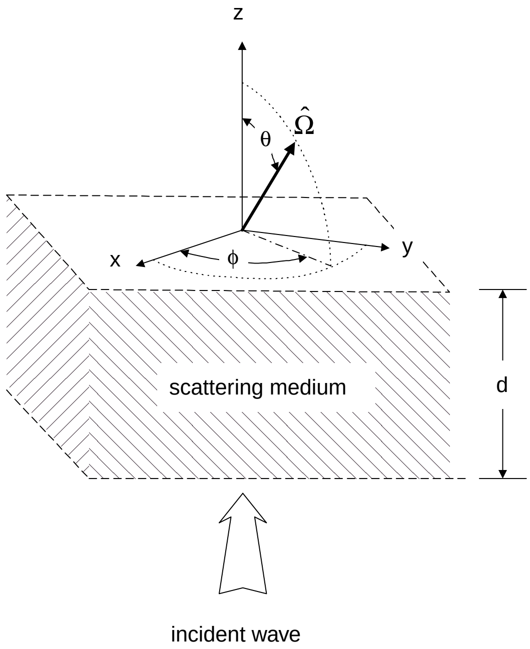


FIG. 2.1. *The plane-parallel problem.*

In addition, the source F does not vary with respect to x and y. We also neglect polarization so that $Q = U = V = 0$ in (1.1)–(1.4), and the 4×4 matrix $\mathbf{S}(\widehat{\Omega} \cdot \widehat{\Omega}')$ is replaced by the scalar function $P(\widehat{\Omega}, \widehat{\Omega}') = \mathbf{S}_{11}(\widehat{\Omega}, \widehat{\Omega}')$ (the $(1,1)$ entry of $\mathbf{S}$). The scalar function P is called the phase function. Under these assumptions, the vector radiative transport equation (1.1) reduces to

$$\frac{1}{v}\frac{\partial}{\partial t} I(z,\mu,\phi,t) + \mu\frac{\partial}{\partial z} I(z,\mu,\phi,t) + \mathcal{Q}[I](z,\mu,\phi,t) = F(z,\mu,\phi,t), \tag{2.1}$$

with

$$\mathcal{Q}[I] = \sigma_t I(z,\mu,\phi,t) - \sigma_s \int_0^{2\pi}\int_{-1}^{1} P(\mu,\mu',\phi-\phi')\, I(z,\mu',\phi',t)\, d\mu' d\phi'. \tag{2.2}$$

Here, $\mu = \cos\theta$, where θ is the propagation direction angle defined with respect to the positive z-direction, and ϕ is the azimuthal angle (see Figure 2.1). The special form of the azimuthal dependence, $\phi - \phi'$, in the phase function P is a direct consequence of the rotational invariance of the scattering matrix.

By representing the azimuthal dependence of the intensity as a Fourier series,

$$I(z,\mu,\phi,t) = \sum_{n=-\infty}^{\infty} I_n(z,\mu,t)\, \mathrm{e}^{in\phi}, \tag{2.3}$$

where the coefficients of this expansion are defined as

$$I_n(z,\mu,t) = \frac{1}{2\pi}\int_0^{2\pi} I(z,\mu,\phi,t)\, \mathrm{e}^{-in\phi} d\phi, \tag{2.4}$$

one finds that each Fourier mode decouples from the others in (2.1). Dropping the index n for simplicity, we find that the resultant problem for each harmonic is

$$\frac{1}{v}\frac{\partial}{\partial t}I(z,\mu,t)+\mu\frac{\partial}{\partial z}I(z,\mu,t)+\mathcal{Q}[I](z,\mu,t)=F(z,\mu,t) \quad \text{in} \quad \mathbb{X}, \tag{2.5a}$$

$$I(z=0,\mu,t)=0 \quad \text{on} \quad (0,1]\times[0,T], \tag{2.5b}$$

$$I(z=d,\mu,t)=0 \quad \text{on} \quad [-1,0)\times[0,T], \tag{2.5c}$$

$$I(z,\mu,t=0)=0 \quad \text{in} \quad [0,d]\times[-1,1], \tag{2.5d}$$

where $\mathbb{X}=[0,d]\times[-1,1]\times[0,T]$ and

$$\mathcal{Q}[I]=\sigma_t I(z,\mu,t)-\sigma_s\int_{-1}^{1}p(\mu,\mu')\,I(z,\mu',t)\,d\mu'. \tag{2.6}$$

In (2.5a) F is a coefficient of an azimuthal Fourier series expansion of the source. Here, we distinguish the Fourier coefficient of the phase function P defined in (1.2) by p in (2.6). It is normalized according to

$$\int_{-1}^{1}p(\mu,\mu')d\mu'=1.$$

Henceforth, we concentrate on solving (2.5) for only one Fourier mode since each mode is decoupled from the others.

2.1. Spatial discretization. Let us change variables in (2.5) from $z\in[0,d]$ to $s\in[-1,1]$ by the linear transformation $s=2z/d-1$. Under this change of variables, (2.5a) becomes

$$\frac{1}{v}\frac{\partial}{\partial t}I(s,\mu,t)+\frac{2}{d}\mu\frac{\partial}{\partial s}I(s,\mu,t)+\mathcal{Q}[I](s,\mu,t)=F(s,\mu,t). \tag{2.7}$$

Now we approximate the spatial dependence of the intensity by the Chebyshev spectral expansion

$$I(s,\mu,t)\cong\sum_{k=0}^{N}a_k(\mu,t)T_k(s). \tag{2.8}$$

The Chebyshev polynomials $T_k(s)$ are orthogonal with respect to the weighted inner product

$$(T_j,T_k)=\int_{-1}^{1}T_j(s)\,T_k(s)\,\frac{ds}{\sqrt{1-s^2}}=\begin{cases}\pi\delta_{j,k} & \text{for } k=0,\\ \frac{\pi}{2}\delta_{j,k} & \text{for } k\geq 1\end{cases} \tag{2.9}$$

and are normalized so that $T_k(s=\pm1)=(\pm1)^k$. Here, $\delta_{j,k}$ is the Kronecker delta. Since the expansion functions T_k do not satisfy the boundary conditions (2.5b) and (2.5c), the extra set of equations

$$I(s=-1,\mu,t)=\sum_{k=0}^{N}(-1)^k a_k(\mu,t)=0 \qquad \text{on } (0,1]\times[0,T], \tag{2.10a}$$

$$I(s=+1,\mu,t)=\sum_{k=0}^{N}a_k(\mu,t)=0 \qquad \text{on } [-1,0)\times[0,T] \tag{2.10b}$$

must be included. Each boundary condition imposes the value of the intensity only over half of the angular data ($\mu < 0$ or $\mu > 0$). Therefore, the composition of (2.10a) and (2.10b) defines a single equation for the expansion coefficients on $[-1,1] \times [0,T]$. Furthermore, we note that considering different boundary conditions does not alter any other aspect of this method.

The Chebyshev spectral approximation to (2.7) is given by (2.8) with (2.10) and

$$\frac{1}{v}\frac{\partial a_k}{\partial t} + \frac{2}{d}\mu\left(T_k, \frac{\partial I}{\partial s}\right) + (T_k, \mathcal{Q}[I]) = (T_k, F), \qquad k = 0, \dots, N. \tag{2.11}$$

Next, we approximate the spatial derivative by another spectral expansion,

$$\frac{\partial I(s,\mu,t)}{\partial s} \cong \sum_{k=0}^{N} A_k(\mu,t) T_k(s), \tag{2.12}$$

where A_k is related to a_k through

$$a_k = \frac{1}{2k}\left[c_{k-1}A_{k-1} - A_{k+1}\right] \quad \text{for } k = 1, 2, \dots, N. \tag{2.13}$$

In (2.13), we assume that $A_{N+1} \ll A_N$ so that $a_N = \frac{1}{2N}c_{N-1}A_{N-1}$. The normalization quantity c_k is defined as

$$c_k = \begin{cases} 2 & \text{for } k = 0, \\ 1 & \text{for } k = 1, \dots, N-1. \end{cases} \tag{2.14}$$

Substituting (2.8) and (2.12) into (2.11) we obtain the system of integro-differential equations

$$\frac{\partial a_k(\mu,t)}{\partial t} + \frac{2v}{d}\mu A_k(\mu,t) + v\mathcal{Q}[a_k](\mu,t) = vF_k(\mu,t), \qquad k = 0, \dots, N, \tag{2.15}$$

for the expansion coefficients a_k and A_k. Note that there are $N+1$ equations in (2.15), an extra equation due to the boundary condition (2.10), plus the N relations (2.13) for the $2N+2$ unknowns in $\{a_k, A_k\}$. Later in this discussion, we use (2.13) to reduce the number of unknowns to $N+1$. In (2.15), F_k are the coefficients of the Chebyshev spectral approximation of the source function. Finally, we note that the resolution requirements needed to compute an accurate numerical solution of the system do not directly depend on the thickness of the medium $z = d$. Rather, spectrally resolving the source function over the spatial domain dictates the resolution requirements. In other words, sources that rapidly vary in the spatial domain require more Chebyshev modes than smoothly varying ones.

2.1.1. Angular discretization. In using a Chebyshev spectral approximation for the spatial variable, we obtain the integro-differential system (2.15) for expansion coefficients that depend on angle and time. We now focus attention on accurately treating the scattering operation. While there are several different ways of treating the integral operation in the radiative transfer equation, such as spherical harmonics and finite element expansions, a simple and effective method is the discrete ordinate method. By using this method, one can accurately approximate the scattering operator and easily adjust the angular resolution needed for different levels of anisotropy. Methods using spherical harmonics expansions require moderately low levels of anisotropy for efficient use. Otherwise, the number of spherical harmonics needed becomes restrictively large for practical computations. Finite element methods are useful for highly anisotropic scattering [20] but are more complicated to implement than discrete ordinate methods. Choosing any of these methods for the angular discretization would work without interfering with the rest of the algorithm.

Now let us replace the continuous angular variable μ with a set of discrete points $\{\mu_i\}$. Then

$$\frac{\partial a_k(\mu_i,t)}{\partial t}+\frac{2v}{d}\mu_i A_k(\mu_i,t)+v\mathrm{Q}[a_k](\mu_i,t)=vF_k(\mu_i,t)\qquad \text{for } i=1,\dots,q, \tag{2.16}$$

where $k=0,\dots,N$. We use these discrete points μ_i to approximate the scattering operator as

$$\mathrm{Q}[a_k](\mu_i,t)=\sigma_t a_k(\mu_i,t)-\sigma_s\sum_{j=1}^{q} w_j p(\mu_i,\mu_j)\,a_k(\mu_j,t). \tag{2.17}$$

The quality of this approximation will of course depend on the choice of the quadrature rule. We refer to [17] and references therein for a more detailed discussion on this subject. We chose a quadrature rule where μ_j are the zeros of the Legendre polynomial of order q and w_j are the corresponding Gaussian weights.

Using matrix notation, (2.16) can be written as

$$\frac{\partial \mathbf{a}_k(t)}{\partial t}+\frac{2v}{d}\mathrm{L}\mathbf{A}_k(t)+v\mathrm{Q}[\mathbf{a}_k](t)=v\mathbf{F}_k(t)\qquad \text{for } k=0,\dots,N, \tag{2.18}$$

where we have introduced the $q\times 1$ vectors

$$\mathbf{a}_k(t)=(a_k(\mu_1,t),a_k(\mu_2,t),\dots,a_k(\mu_q,t)), \tag{2.19a}$$

$$\mathbf{A}_k(t)=(A_k(\mu_1,t),A_k(\mu_2,t),\dots,A_k(\mu_q,t)), \tag{2.19b}$$

$$\mathbf{F}_k(t)=(F_k(\mu_1,t),F_k(\mu_2,t),\dots,F_k(\mu_q,t)), \tag{2.19c}$$

and the $q\times q$ matrix $\mathrm{L}=\mathrm{diag}(\mu_1,\mu_2,\dots,\mu_q)$.

2.2. Temporal discretization. To integrate (2.18) in time, we use a trapezoid rule and obtain

$$\left[\mathbb{I}+\frac{v\Delta t}{2}\mathrm{Q}\right]\mathbf{a}_k^{n+1}+\frac{v\Delta t}{d}\mathrm{L}\mathbf{A}_k^{n+1}=\left[\mathbb{I}-\frac{v\Delta t}{2}\mathrm{Q}\right]\mathbf{a}_k^{n}-\frac{v\Delta t}{d}\mathrm{L}\mathbf{A}_k^{n}+\frac{v\Delta t}{2}\left[\mathbf{F}_k^{n+1}+\mathbf{F}_k^{n}\right], \tag{2.20}$$

where $\mathbb{I}$ is the identity matrix and $\mathbf{a}_k^n$ and $\mathbf{A}_k^n$ are the coefficients for the intensity and its spatial derivative evaluated at time $t_n=n\Delta t$, respectively. The numerical scheme (2.20) corresponds to the Crank–Nicholson method which is fully implicit, second-order accurate, and unconditionally stable.

Now we discuss an efficient algorithm to solve (2.20). At each time step we must solve

$$\mathrm{K}\mathbf{a}_k+\mathrm{M}\mathbf{A}_k=\mathbf{f}_k, \tag{2.21}$$

where $\mathrm{K}=\mathbb{I}+v\Delta t\mathrm{Q}/2$, $\mathrm{M}=v\Delta t\mathrm{L}/d$, and $\mathbf{f}_k$ is a known quantity corresponding to the right-hand side of (2.20).

From (2.13) we see that transforming from $\mathbf{a}_k$ to $\mathbf{A}_k$ is a simple operation possessing an inherent sparsity. To take advantage of this sparsity, we construct a system of equations to solve for the coefficients of the intensity's spatial derivative rather than coefficients of the intensity (see [18] and references therein for details regarding this technique). After transforming terms involving $\mathbf{a}_k$ to operations on $\mathbf{A}_k$, we obtain a block tridiagonal system

$$\mathrm{M}\mathbf{A}_0 + \mathrm{K}\mathbf{a}_0 = \mathbf{f}_0 \qquad \text{for } k = 0, \tag{2.22a}$$

$$\mathrm{M}\mathbf{A}_1 + \mathrm{K}\mathbf{A}_0 - \frac{1}{2}\mathrm{K}\mathbf{A}_2 = \mathbf{f}_1 \qquad \text{for } k = 1, \tag{2.22b}$$

$$\mathrm{M}\mathbf{A}_k + \frac{1}{2k}\mathrm{K}\left[\mathbf{A}_{k-1} - \mathbf{A}_{k+1}\right] = \mathbf{f}_k \qquad \text{for } k = 2, \dots, N-1, \tag{2.22c}$$

$$\mathrm{M}\mathbf{A}_N + \frac{1}{2N}\mathrm{K}\mathbf{A}_{N-1} = \mathbf{f}_N \qquad \text{for } k = N. \tag{2.22d}$$

For the boundary conditions (2.5b) and (2.5c), we evaluate each expansion coefficient a_k at the quadrature points μ_i and operate on that result by (2.13) to obtain

$$\mathbf{a}_0^+ - \mathbf{A}_0^+ + \frac{1}{4}\mathbf{A}_1^+ - \sum_{k=2}^{N-1} \frac{(-1)^k}{2}\left[\frac{1}{k+1} - \frac{1}{k-1}\right]\mathbf{A}_k^+ + \frac{1}{2(N-1)}\mathbf{A}_N^+ = 0, \tag{2.23a}$$

$$\mathbf{a}_0^- + \mathbf{A}_0^- + \frac{1}{4}\mathbf{A}_1^- + \sum_{k=2}^{N-1} \frac{1}{2}\left[\frac{1}{k+1} - \frac{1}{k-1}\right]\mathbf{A}_k^- - \frac{1}{2(N-1)}\mathbf{A}_N^- = 0. \tag{2.23b}$$

Here, we have defined $\mathbf{A}_k^{\mp}$ to be subvectors of $\mathbf{A}_k$,

$$\mathbf{A}_k = \begin{bmatrix} \mathbf{A}_k^- \\ \mathbf{A}_k^+ \end{bmatrix}, \tag{2.24}$$

where $\mathbf{A}_k^-$ corresponds to the part of $\mathbf{A}_k$ in which $\mu_i < 0$ and $\mathbf{A}_k^+$ corresponds to the part of $\mathbf{A}_k$ in which $\mu_i > 0$. The same notation applies for $\mathbf{a}_0^{\mp}$.

Equations (2.22) and (2.23) define a system of equations for $\mathbf{A}_0, \mathbf{A}_1, \dots, \mathbf{A}_N$ and $\mathbf{a}_0$. After computing a solution for these unknowns, we can compute the desired expansion coefficients of the intensity $\mathbf{a}_k$ by applying (2.13) again. One can solve this system using Gaussian elimination, but the cost is $O\left(q^3(N+1)^3\right)$. Instead, we use the generalized deflated block elimination method [19] that takes advantage of the inherent sparsity of this system to reduce the number of operations significantly. The number of operations needed to solve (2.22) and (2.23) using this method is $O(q^3(N-1))$. We discuss the details of this solution method in Appendix A.

3. Generalizations.

We now study the modifications needed to deal with more general problems. We carry out these modifications in a way that preserves the order of accuracy, computational cost, and ease of implementation.

3.1. Layered media.

In many applications the properties of the medium in the vertical direction are not homogeneous. In that case, a multilayered model where the scattering and total cross-sections are piecewise constant functions in space might be applied. Using the Chebyshev spectral method, we find that solving problems in layered media is essentially the same as solving homogeneous problems in each layer. Dealing with interfaces between layers requires only adding extra conditions that reside with the boundary conditions in the resultant system of equations.

For simplicity, let us consider a medium with only two layers. In that case

$$\sigma_{s,t}(z) = \begin{cases} \sigma_{s,t}^{(1)} & \text{for } z \in [0, d_1), \\ \sigma_{s,t}^{(2)} & \text{for } z \in [d_1, d_2]. \end{cases} \tag{3.1}$$

In addition to the boundary conditions (2.5b) and (2.5c), we impose that the intensity at the interface $z = d_1$ between the layers is continuous so that

$$I(z,\mu,t) = \begin{cases} I^{(1)}(z,\mu,t) & \text{for } z \in [0, d_1], \\ I^{(2)}(z,\mu,t) & \text{for } z \in [d_1, d_2]. \end{cases} \tag{3.2}$$

Now let us introduce two spatial variables

$$s_1 = 2\frac{z}{d_1} - 1 \qquad \text{for } z \in [0, d_1], \tag{3.3}$$

$$s_2 = 2\frac{z-d_1}{d_2-d_1} - 1 \quad \text{for } z \in [d_1, d_2] \tag{3.4}$$

so that the radiative transfer equation for each harmonic is

$$\left[\frac{1}{v}\frac{\partial}{\partial t} + \frac{2}{d_1}\mu\frac{\partial}{\partial s_1} + \mathcal{Q}^{(1)}\right] I^{(1)}(s_1,\mu,t) = F^{(1)}(s_1,\mu,t) \qquad \text{for } s_1 \in [-1,1], \tag{3.5a}$$

$$\left[\frac{1}{v}\frac{\partial}{\partial t} + \frac{2}{d_2-d_1}\mu\frac{\partial}{\partial s_2} + \mathcal{Q}^{(2)}\right] I^{(2)}(s_2,\mu,t) = F^{(2)}(s_2,\mu,t) \quad \text{for } s_2 \in [-1,1]. \tag{3.5b}$$

Here, we define the scattering operator as

$$\mathcal{Q}^{(j)}[I] = \sigma_t^{(j)} I(s_j,\mu,t) - \sigma_s^{(j)} \int_{-1}^{1} p(\mu,\mu') I(s_j,\mu',t)\, d\mu', \qquad j = 1,2, \tag{3.6}$$

and the source function as

$$F(z,\mu,t) = \begin{cases} F^{(1)}(z,\mu,t) & \text{for } z \in [0, d_1], \\ F^{(2)}(z,\mu,t) & \text{for } z \in [d_1, d_2]. \end{cases} \tag{3.7}$$

The boundary conditions for this problem are

$$I^{(1)}(s_1 = -1,\mu,t) = 0 \quad \text{on} \quad (0,1] \times [0,T], \tag{3.8a}$$

$$I^{(2)}(s_2 = +1,\mu,t) = 0 \quad \text{on} \quad [-1,0) \times [0,T], \tag{3.8b}$$

and the continuity condition at the interface implies that

$$I^{(1)}(s_1 = +1,\mu,t) = I^{(2)}(s_2 = -1,\mu,t) \qquad \text{on} \quad [-1,1] \times [0,T]. \tag{3.9}$$

Now we consider Chebyshev spectral approximations for $I^{(1)}$ and $I^{(2)}$:

$$I^{(j)}(s_j,\mu,t) \cong \sum_{k=0}^{N} a_k^{(j)}(\mu,t) T_k(s_j), \qquad j = 1,2. \tag{3.10}$$

In performing the same analysis used for the homogeneous case, we obtain

$$\frac{\partial}{\partial t}a_k^{(1)}(\mu,t)+\frac{2v}{d_1}\mu A_k^{(1)}(\mu,t)+v\mathcal{Q}^{(1)}[a_k^{(1)}](\mu,t)=vF_k^{(1)}(\mu,t), \tag{3.11a}$$

$$\frac{\partial}{\partial t}a_k^{(2)}(\mu,t)+\frac{2v}{d_2-d_1}\mu A_k^{(2)}(\mu,t)+v\mathcal{Q}^{(2)}[a_k^{(2)}](\mu,t)=vF_k^{(2)}(\mu,t) \tag{3.11b}$$

for $k=0,\dots,N$. Notice that (3.11a) and (3.11b) each resemble (2.15) modulo scalar factors. Hence, we use exactly the same method outlined above for the homogeneous case for each equation.

After applying a transformation similar to the one discussed in section 2.2 on (3.11), we construct a system of equations for $(\mathbf{A}_k^{(1)},\mathbf{A}_k^{(2)},\mathbf{a}_0^{(1)},\mathbf{a}_0^{(2)})$. In particular, with (3.11), we have $2N+2$ equations for $2N+4$ unknowns. The other two needed equations are the boundary conditions

$$\sum_{k=0}^{N}(-1)^k a_k^{(1)}(\mu,t)=0 \qquad \text{on } (0,1]\times[0,T], \tag{3.12a}$$

$$\sum_{k=0}^{N} a_k^{(2)}(\mu,t)=0 \qquad \text{on } [-1,0)\times[0,T], \tag{3.12b}$$

and the continuity condition

$$\sum_{k=0}^{N}\left[a_k^{(1)}(\mu,t)-(-1)^k a_k^{(2)}(\mu,t)\right]=0. \tag{3.13}$$

Once again, since both (3.12a) and (3.12b) each impose conditions on half of the angular domain, the composition of these two equation defines a single equation on $[-1,1]\times[0,T]$. The continuity condition (3.13) gives another equation defined on $[-1,1]\times[0,T]$. Therefore, we have $2N+4$ equations for $2N+4$ unknowns.

Because coupling between expansion coefficients corresponding to different layers occurs only through the boundary and continuity conditions, the solution method for layered media is essentially the solution for two homogeneous problems. In general, the solution method for layered media problems corresponds to a composition of n homogeneous problems where n is the number of layers. Although the addition of more layers linearly increases the amount of work needed to compute a solution, in practice one needs fewer numbers of Chebyshev modes to resolve each layer.

3.2. Inhomogeneous media. For more general problems where a multilayered model cannot be applied, we must consider continuously varying cross-sections. The difficulty in dealing with these variable coefficients in a spectral approximation is that they necessarily lead to convolution sums. These convolutions make standard fully implicit time stepping impractical to compute since they couple all of the Chebyshev modes, thereby destroying the sparsity used to compute solutions efficiently. On the other hand, it is well known that fully explicit schemes for standard Chebyshev methods with N modes have a stability condition of $v\Delta t < O(1/N^2)$. This is a disadvantage when compared with other fully explicit schemes using Fourier spectral or finite difference methods with stability conditions $v\Delta t < O(1/N)$. To circumvent this restriction, we use a semi-implicit method where only the spatial inhomogeneities are treated explicitly.

We first express the variable cross-sections as

$$\sigma_{s,t} = \sigma_{s,t}^{(0)} + \tilde{\sigma}_{s,t}(s), \tag{3.14}$$

where $\sigma_s^{(0)}$ and $\sigma_t^{(0)}$ are constants. We choose to represent the variable cross-sections in this way so we can treat a portion of the cross-section implicitly. This is important for the time stability limit of the scheme. By substituting (3.14) into the radiative transfer equation, we obtain

$$\frac{1}{v}\frac{\partial}{\partial t}I(s,\mu,t) + \frac{2}{d}\mu\frac{\partial}{\partial s}I(s,\mu,t) + \mathcal{Q}[I](s,\mu,t) = F(s,\mu,t) - \tilde{\mathcal{Q}}[I](s,\mu,t), \tag{3.15}$$

with

$$\tilde{\mathcal{Q}}[I](s,\mu,t) = \tilde{\sigma}_t(s)I(s,\mu,t) - \tilde{\sigma}_s(s)\int_{-1}^{1} p(\mu,\mu')\, I(s,\mu',t)\, d\mu'. \tag{3.16}$$

The only difference between (2.7) for the homogeneous case and (3.15) is the inhomogeneous term $\tilde{\mathcal{Q}}[I]$.

In order to maintain high accuracy and low computational cost, we treat $\tilde{\mathcal{Q}}[I]$ explicitly in time with a second-order Adams–Bashforth scheme so that

$$\begin{aligned}\left[\mathbb{I} + \frac{v\Delta t}{2}\mathrm{Q}\right]\mathbf{a}_k^{n+1} + \frac{v\Delta t}{d}\mathbf{L}\mathbf{A}_k^{n+1} &= \left[\mathbb{I} - \frac{v\Delta t}{2}\mathrm{Q}\right]\mathbf{a}_k^{n} - \frac{v\Delta t}{d}\mathbf{L}\mathbf{A}_k^{n} \\ &\quad + \frac{v\Delta t}{2}\left[\mathbf{F}_k^{n+1} + \mathbf{F}_k^{n}\right] - \frac{v\Delta t}{2}\left[3\tilde{\mathrm{Q}}[\mathbf{a}]_k^n - \tilde{\mathrm{Q}}[\mathbf{a}]_k^{n-1}\right].\end{aligned} \tag{3.17}$$

The convolution terms

$$\left[\tilde{\sigma}_{t,s} \star a\right]_k^n(\mu_i) = \int_{-1}^{1} \tilde{\sigma}_{t,s}(s) I(s,\mu_i,t_n) T_k(s)\frac{ds}{\sqrt{1-s^2}} \tag{3.18}$$

contained within $\tilde{\mathrm{Q}}[\mathbf{a}]_k^n$ are evaluated explicitly in time, so there is no coupling between different k modes at time level $n+1$. These convolutions can be computed either in the spatial domain (pseudospectrally) or in the Chebyshev transform domain. Notice that the only difference between the homogeneous and inhomogeneous problem lies in constructing the right-hand side of (3.17).

Dealing with the inhomogeneities explicitly in time compromises the stability of the numerical scheme. Specifically, the explicit terms in (3.17) require $v\Delta t \max(\tilde{\sigma}_t) < 1$ for stability. This range is not restrictive in most applications since only a portion of the variable cross-sections is treated explicitly. However, if this stability condition does become problematic, an implicit fractional step method similar to that presented in [14] may be applied. Another possibility is to decompose the medium into strips over which the variable cross-sections are assumed to be piecewise constant with some variable perturbation. Then, one can construct and solve the corresponding layered background medium problem with the same method for inhomogeneous media presented above in each layer. This approach would minimize the size of the variable perturbation within each strip, thereby reducing the time step restriction.

3.3. Vector radiative transfer. Modifying the methods presented in section 2 to include polarization is straightforward. In fact, the only modification required lies in constructing the system of equations for

$$\begin{bmatrix} I(s,\mu,t) \\ Q(s,\mu,t) \\ U(s,\mu,t) \\ V(s,\mu,t) \end{bmatrix} \cong \sum_{k=0}^{N} \begin{bmatrix} a_k^{(I)}(\mu,t) \\ a_k^{(Q)}(\mu,t) \\ a_k^{(U)}(\mu,t) \\ a_k^{(V)}(\mu,t) \end{bmatrix} T_k(s), \tag{3.19}$$

where $a_k^{(I)}$, $a_k^{(Q)}$, $a_k^{(U)}$, and $a_k^{(V)}$ are the spatial projections of the Stokes parameters onto the Chebyshev polynomial of order k. Proceeding in a similar manner as in section 2, we obtain

$$\frac{\partial}{\partial t} \begin{bmatrix} \mathbf{a}_k^{(I)} \\ \mathbf{a}_k^{(Q)} \\ \mathbf{a}_k^{(U)} \\ \mathbf{a}_k^{(V)} \end{bmatrix} + \frac{2v}{d}\overline{\mathrm{L}} \begin{bmatrix} \mathbf{A}_k^{(I)} \\ \mathbf{A}_k^{(Q)} \\ \mathbf{A}_k^{(U)} \\ \mathbf{A}_k^{(V)} \end{bmatrix} + v\mathrm{Q} \begin{bmatrix} \mathbf{a}_k^{(I)} \\ \mathbf{a}_k^{(Q)} \\ \mathbf{a}_k^{(U)} \\ \mathbf{a}_k^{(V)} \end{bmatrix} = v \begin{bmatrix} \mathbf{f}_k^{(I)} \\ \mathbf{f}_k^{(Q)} \\ \mathbf{f}_k^{(U)} \\ \mathbf{f}_k^{(V)} \end{bmatrix}, \qquad k = 0,\dots,N, \tag{3.20}$$

where

$$\mathbf{a}_k^{(j)} = (a_k^{(j)}(\mu_1,t),\, a_k^{(j)}(\mu_2,t),\, \dots\, , a_k^{(j)}(\mu_q,t)), \qquad j = I,Q,U,V. \tag{3.21}$$

Similar notation applies for $\mathbf{A}_k^{(j)}$ and $\mathbf{f}_k^{(j)}$. Here, $\overline{\mathrm{L}}$ is the $4q \times 4q$ block diagonal matrix defined as $\overline{\mathrm{L}} = \mathrm{diag}(\mathrm{L},\mathrm{L},\mathrm{L},\mathrm{L})$, where $\mathrm{L} = \mathrm{diag}(\mu_1,\mu_2,\dots,\mu_q)$ as in section 2. By introducing the vectors

$$\boldsymbol{\psi}_k = \left(\mathbf{a}_k^{(I)},\, \mathbf{a}_k^{(Q)},\, \mathbf{a}_k^{(U)},\, \mathbf{a}_k^{(V)}\right), \tag{3.22}$$

$$\Psi_k = \left(\mathbf{A}_k^{(I)},\, \mathbf{A}_k^{(Q)},\, \mathbf{A}_k^{(U)},\, \mathbf{A}_k^{(V)}\right), \tag{3.23}$$

we see that (3.20) can be written in the form

$$\frac{\partial}{\partial t}\boldsymbol{\psi}_k + \frac{2v}{d}\overline{\mathrm{L}}\Psi_k + v\mathrm{Q}[\boldsymbol{\psi}_k] = v\mathbf{F}_k, \qquad k = 0,\dots,N. \tag{3.24}$$

This system of equations is exactly the same as (2.18), and so it can be treated in exactly the same way. Because each unknown Chebyshev mode $\boldsymbol{\psi}$ is a $4q \times 1$ vector, the work needed to compute the solution for a vector problem is $O(4^3q^3(N-1))$.

3.4. Two spatial dimensions. We now consider scalar radiative transfer equations in two spatial dimensions. From the numerical point of view, the most important changes appear in going from the one-dimensional case to the two-dimensional one. Generalizations from two spatial dimensions to three are usually straightforward.

We first consider the two-dimensional radiative transfer equation arising from a source term $\mathbf{F}$ that depends on x and z. This source can model problems involving for example, beam waves. The homogeneous medium through which the wave propagates is still infinite in the x-y direction and bounded by two planes located at $z = 0$ and $z = d_z$. After the linear transformation $s = 2z/d_z - 1$, the scalar two-dimensional radiative transfer equation is

$$\begin{aligned} \frac{1}{v}\frac{\partial}{\partial t}I(s,x,\widehat{\Omega},t) + \frac{2}{d_z}\mu\frac{\partial}{\partial s}I(s,x,\widehat{\Omega},t) + \sqrt{1-\mu^2}\cos\phi\frac{\partial}{\partial x}I(s,x,\widehat{\Omega},t) \\ + \mathcal{Q}[I](s,x,\widehat{\Omega},t) = F(s,x,\widehat{\Omega},t), \end{aligned} \tag{3.25}$$

with

$$\mathcal{Q}[I] = \sigma_t I(s,x,\mu,\phi,t) - \sigma_s \int_0^{2\pi}\int_{-1}^{1} P(\mu,\mu',\phi-\phi')\, I(s,x,\mu',\phi',t)\, d\mu' d\phi'. \tag{3.26}$$

Boundary and initial conditions are given by (2.5b)–(2.5d) for all points x. Due to the $\cos\phi$ in front of the partial derivative with respect to x, the Fourier modes of the azimuthal variable do not decouple as in the one-dimensional problem. Therefore, the discrete ordinate method must be extended to include a quadrature rule for the azimuthal variable as well.

Since the medium is infinite in the horizontal direction, it is natural to deal with the x dependence of the intensity I by a Fourier series. Consequently, we approximate I by the expansion

$$I(s,x,\widehat{\Omega},t) \cong \sum_{k=0}^{N_v}\sum_{l=-N_h/2}^{N_h/2} a_{kl}(\widehat{\Omega},t) T_k(s) e^{ilx}. \tag{3.27}$$

Proceeding in an analogous way to section 2 we obtain a semidiscrete equation for each mode l that is decoupled from the others. Each decoupled semidiscrete equation has the same form as (2.15), where the operator $\mathcal{Q}$ has to be replaced by

$$\begin{aligned}\mathcal{Q}_l[a_{kl}](\mu,\phi,t) &= \sigma_t\, a_{kl}(\mu,\phi,t) + il\sqrt{1-\mu^2}\cos\phi\, a_{kl}(\mu,\phi,t)\\ &\quad - \sigma_s \int_0^{2\pi}\int_{-1}^{+1} P(\mu,\mu',\phi-\phi')\, a_{kl}(\mu',\phi',t)\, d\mu' d\phi'.\end{aligned} \tag{3.28}$$

This problem can be solved by the same procedure as the one-dimensional case for each mode l independently. It is also clear that the case where the coefficients σ_t and σ_s depend on the position can be treated with a semi-implicit method as in section 3.2.

As a second example, let us consider that the horizontal direction is bounded by two planes at $x = 0$ and $x = d_x$ so that we have the two extra boundary conditions

$$I(z, x=0, \widehat{\Omega}, t) = 0 \qquad \text{on} \qquad \Gamma_x, \tag{3.29}$$

$$I(z, x=d_x, \widehat{\Omega}, t) = 0 \qquad \text{on} \qquad \Gamma_x, \tag{3.30}$$

where Γ_x denotes the set of points $[0,d_z]\times\mathbb{S}^2\times[0,T]$ such that $\nu(x)\cdot\widehat{\Omega} < 0$. After the linear transformation $r = 2x/d_x - 1$, we approximate the intensity I and its derivatives by the expansions

$$\begin{aligned} I(s,r,\widehat{\Omega},t) &\cong \sum_{k=0}^{N_v}\sum_{l=0}^{N_h} a_{kl}(\widehat{\Omega},t)T_k(s)T_l(r),\\ \frac{\partial}{\partial s}I(s,r,\widehat{\Omega},t) &\cong \sum_{k=0}^{N_v}\sum_{l=0}^{N_h} A_{kl}(\widehat{\Omega},t)T_k(s)T_l(r),\\ \frac{\partial}{\partial r}I(s,r,\widehat{\Omega},t) &\cong \sum_{k=0}^{N_v}\sum_{l=0}^{N_h} B_{kl}(\widehat{\Omega},t)T_k(s)T_l(r).\end{aligned} \tag{3.31}$$

Again, proceeding in a similar fashion we obtain the semidiscrete set of equations

$$\frac{\partial a_{kl}}{\partial t} + \frac{2v}{d_z}\mu A_{kl} + \frac{2v}{d_x}\sqrt{1-\mu^2}\cos\phi B_{kl} + v\mathcal{Q}[a_{kl}] = vF_{kl} \tag{3.32}$$

for $k = 0, 1, \ldots, N_v$ and $l = 0, 1, \ldots, N_h$. After discretization in the angular variables μ and ϕ and the time variable t, and using relations

$$a_{kl} = \frac{1}{2k}\left[c_{k-1}A_{k-1,l} - A_{k+1,l}\right], \tag{3.33a}$$

$$a_{kl} = \frac{1}{2l}\left[c_{l-1}B_{k,l-1} - B_{k,l+1}\right], \tag{3.33b}$$

one could resolve the resulting system for the $2(N+1)+(N+1)^2$ variables a_{0l}, B_{kN_h}, and A_{kl} as in section 2. However, this procedure is cumbersome due to the recursion relation between B_{kl} and A_{kl}. Therefore, we solve the fully discretized equations arising from (3.32) with an alternating direction implicit or approximate factorization method. The two-step method (see Appendix B) is

$$\left[\mathbb{I} + \frac{v\Delta t}{4}\mathrm{Q}\right]\mathbf{a}^*_{kl} + \frac{v\Delta t}{d_z}\mathrm{L}_z\mathbf{A}^*_{kl} = \left[\mathbb{I} - \frac{v\Delta t}{4}\mathrm{Q}\right]\mathbf{a}^n_{kl} - \frac{v\Delta t}{d_z}\mathrm{L}_z\mathbf{A}^n_{kl} + \frac{v\Delta t}{2}\left[\mathbf{F}^{n+1}_{kl} + \mathbf{F}^n_{kl}\right],$$

$$\left[\mathbb{I} + \frac{v\Delta t}{4}\mathrm{Q}\right]\mathbf{a}^{n+1}_{kl} + \frac{v\Delta t}{d_x}\mathrm{L}_x\mathbf{B}^{n+1}_{kl} = \left[\mathbb{I} - \frac{v\Delta t}{4}\mathrm{Q}\right]\mathbf{a}^*_{kl} - \frac{v\Delta t}{d_x}\mathrm{L}_x\mathbf{B}^*_{kl} \tag{3.34}$$

for each k and each l. The matrices Q, L_z, and L_x in (3.34) are defined in Appendix B. Algorithm (3.34) again has the desired property of preserving the structure of the one-dimensional problem. For each fixed l, one needs only to solve two block tridiagonal systems. This method is second-order accurate and unconditionally stable.

4. Numerical examples.

Here, we present computations for different radiative transfer problems. We also examine the convergence of these methods in space and time. The convergence of the discrete ordinate method is well established [21]. Kim and Ishimaru [15] demonstrated that the Chebyshev spectral method, in conjunction with the discrete ordinate method, is able to resolve highly anisotropic scattering for these problems.

As an example, we consider a normally incident pulsed plane wave entering the medium at $z = 0$ and solve for the incoherent or scattered intensity. Figures 4.1(a) and (b) show the numerical solutions to scalar and vector problems, respectively. In these examples and all that follow, we normalize spatial units by $\ell = 1/\sigma_t^{(0)}$ and time units by ℓ/v.

For the scalar problem (Figure 4.1(a)), we computed the magnitude of the backscattered flux

$$\mathcal{F}(t) = 2\pi\int_{-1}^{0} I(z=0,\mu,t)|\mu|d\mu \tag{4.1}$$

for three different media of thickness $d = 20$. The scattering and absorption cross-sections

$$\sigma_{s,a} = \sigma_{s,a}^{(0)} + \tilde{\sigma}_{s,a}(z)$$

all have $\sigma_s^{(0)} = 0.98$ and $\sigma_a^{(0)} = 0.02$, and

$$\tilde{\sigma}_{s,a}(z) = A_{s,a}\exp\left[\left(\frac{z - z_{s,a}}{w_{s,a}}\right)^2\right].$$

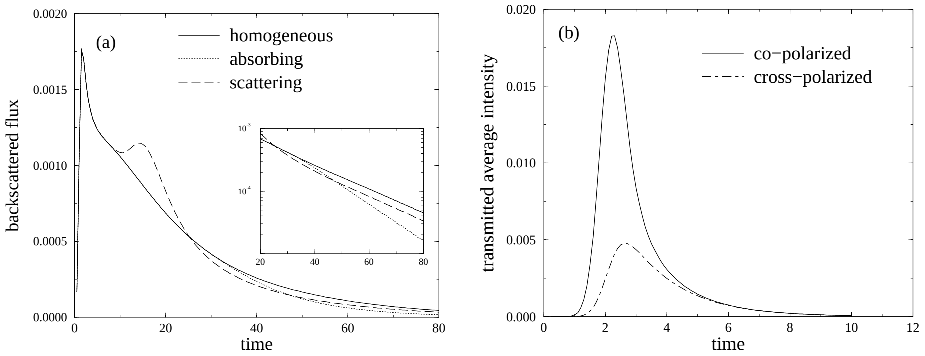


FIG. 4.1. *Example of radiative transfer computations: the left plot* (a) *shows the backscattered flux response of a normally incident pulsed plane wave from a plane-parallel medium of thickness* $d = 20$ *using the Henyey–Greenstein phase function with* $g = 0.85$*. The inset shows a detail of the flux responses' tails on a logarithmic scale. Here, we used* 65 *Chebyshev modes and* 20 *quadrature points with* $\Delta t = 0.01$*. The right plot* (b) *shows the transmitted average intensity response of a left-handed circularly polarized pulsed plane wave normally incident upon a plane-parallel medium containing randomly distributed dielectric spheres immersed in water. The incident light pulse has wavelength* $\lambda = 633\,nm$ *(wavelength generated by a He-Ne laser) and the spherical scatterers are identically sized with radius* $a = 0.05\,\mu m$*. The refractive index of the spheres relative to that of water is approximately* $n = 1.19$*. The scattering matrix for this medium is given by Mie theory. For these computations, we used* 65 *Chebyshev modes,* 40 *quadrature points, and* $\Delta t = 0.01$*.*

The parameters for each medium are given in Table 4.1.

We use the Henyey–Greenstein phase function

$$P(\cos\Theta) = \frac{1}{2}\frac{1-g^2}{(1+g^2-2g\cos\Theta)^{3/2}}, \tag{4.2}$$

where $\cos\Theta = \mu\mu' + \sqrt{1-\mu^2}\sqrt{1-\mu'^2}\cos(\phi-\phi')$ is the cosine of the scattering angle and g is the mean cosine scattering angle or anisotropy factor. Scattering is isotropic for $g = 0$ and becomes more and more sharply peaked in the "forward" direction ($\cos\Theta = 1$) as $g \to 1$.

Since the source is normally impinging the medium, the problem is independent of ϕ and therefore requires only the zero Fourier ϕ-harmonic. Consequently, we use the phase function

$$p(\mu,\mu') = \frac{1}{2\pi}\int_0^{2\pi} P(\mu,\mu',\phi-\phi')d(\phi-\phi'). \tag{4.3}$$

The source function is

$$F(z,\mu,t) = p(\mu,1)f(vt-z)\,\exp\left[-\int_0^z \sigma_t(z')dz'\right], \tag{4.4}$$

where

$$f(t) = \exp\left[-\left(\frac{t-t_o}{T}\right)^2\right] \tag{4.5}$$

is the temporal pulse shape. This source function comes about from the incident intensity as it attenuates from scattering and absorption while propagating into the

TABLE 4.1
Parameter values for the three different media depicted in Figure 4.1(a).

Medium	A_s	z_s	w_s	A_a	z_a	w_a
homogeneous	0	—	—	0	—	—
absorbing	0	—	—	0.5	15	2
scattering	1	5	1	0	—	—

medium, thereby giving rise to the scattered component of the intensity [8]. The time at which the pulse center reaches the boundary at $z = 0$ is t_o and the pulse width is T. For this computation, $t_o = 1$, $T = 0.5$, and $g = 0.85$.

We observe in Figure 4.1(a) that the presence of the scattering inhomogeneity centered at $z_s = 5$ gives rise to an increase of the backscattered response (dashed line). By causality, the first indication of this presence takes place at $t \approx 11$. This is the time that it takes for light to propagate from the boundary to the inhomogeneity and back to the boundary. Similarly, we observe a decrease in the backscattered response at $t \approx 31$ due to the absorption inhomogeneity centered at $z_a = 15$ corresponding to the time needed to reach the inhomogeneity and return to the surface.

In Figure 4.1(b), we show the copolarized (solid line) and cross-polarized (dot-dashed line) components of the transmitted average intensity

$$\mathcal{U}(t) = \frac{1}{2}\int_0^1 I(z = d, \mu, t) d\mu.$$

The incident pulse is left-handed circularly polarized so that $I = V = 1$ and $Q = U = 0$ and normally impinges a medium of thickness $d = 1$. For these computations, $t_o = 1$ and $T = 0.5$. We define the copolarized intensity to be the component that is left-handed circularly polarized

$$I_{\mathrm{LHC}} = \frac{1}{2}(I + V)$$

and the cross-polarized intensity to be the component that is right-handed circularly polarized

$$I_{\mathrm{RHC}} = \frac{1}{2}(I - V).$$

Because early transmitted responses consist of light that has undergone very little scattering as it propagated through the medium, it is entirely copolarized. As time increases, scattering gives rise to depolarization so that the cross-polarized component increases. For very large times, we observe in Figure 4.1(b) that both components are equal.

4.1. Temporal convergence. To verify that the temporal discretization is second-order accurate, we examine time-resolved, backscattered flux responses for the scalar problems explained in sections 2, 3.1, and 3.2. As a test problem, we considered a pulse with $t_o = 1$ and $T = 0.5$ in a medium with thickness $d = 1$ and anisotropy $g = 0.5$ up to time $t = 10$. Since there is no known analytical solution to this problem, we compared flux responses using different time steps to a reference solution computed with $\Delta t = 0.001$. Specifically, we computed the infinity norm of the difference of a solution with lower resolution and the reference solution normalized by the infinity norm of the reference solution. For all of these computations, we used 20 quadrature points.

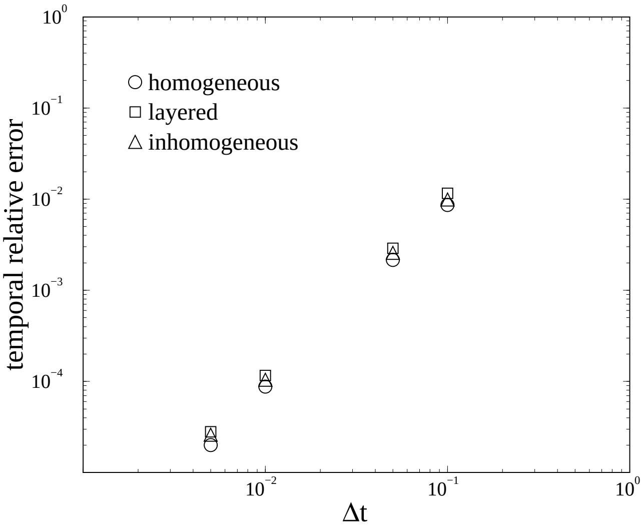


Fig. 4.2. *Computed relative errors of time-resolved, backscattered flux responses as a function of the time step* Δt*. These computations used* 20 *quadrature points. The medium is* $d = 1$ *thick with anisotropy* $g = 0.5$*.*

Table 4.2

Observed rates of convergence for the time discretization using the infinity norm of the difference between the computed solution and a reference solution with $\Delta t = 0.001$*.*

Medium	Convergence rate
homogeneous	2.0174
layered	2.0055
inhomogeneous	1.9874

For the homogeneous problem, we considered a medium with $\sigma_s = 0.98$ and $\sigma_a = 0.02$, and used 65 Chebyshev modes. For the layered problem, we considered three layers with interfaces at $z = 0.4$ and $z = 0.7$ with

$$\sigma_s(z) = \begin{cases} 0.98 & \text{for } z \in [0, 0.4], \\ 1.96 & \text{for } z \in (0.4, 0.7], \\ 0.90 & \text{for } z \in (0.7, 1], \end{cases} \qquad \sigma_a(z) = \begin{cases} 0.02 & \text{for } z \in [0, 0.4], \\ 0.04 & \text{for } z \in (0.4, 0.7], \\ 0.10 & \text{for } z \in (0.7, 1], \end{cases}$$

and used 17 Chebyshev modes in each layer. For the inhomogeneous problem, we considered

$$\tilde{\sigma}_s(z) = 0.98 \exp\left[-\left(\frac{z - 0.4}{0.3}\right)^2\right], \qquad \tilde{\sigma}_a(z) = 0.02 \exp\left[-\left(\frac{z - 0.4}{0.3}\right)^2\right],$$

and used 65 Chebyshev modes.

Plots of the backscattered flux errors appear in Figure 4.2. We tabulated the observed rates of convergence in Table 4.2. As expected, the time discretization exhibits a second-order convergence rate.

4.2. Spatial convergence. Since we use a Chebyshev spectral method to approximate the spatial dependence of the intensity, we expect a superalgebraic convergence as the number of Chebyshev modes increases. To isolate the spatial properties

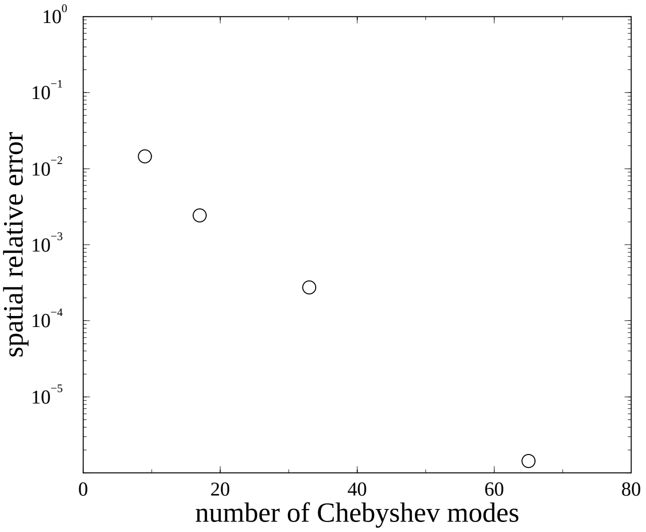


FIG. 4.3. *Relative errors of average intensities as a function of the number of Chebyshev modes for the continuous wave problem. These computations used* 60 *quadrature points for an isotropic medium* ($g = 0$)*.*

of this method, we perform computations for a continuous wave source in a homogeneous medium with $\sigma_s = 0.98$ and $\sigma_a = 0.02$. Furthermore, we consider isotropic scattering ($g = 0$) with 100 quadrature points.

To model an incident continuous wave, we consider the limit in which the temporal pulse width tends toward infinity ($T \to \infty$). In that case, the source function is

$$F(z, \mu) = p(\mu, 1)\,\mathrm{e}^{-\sigma_t z}, \tag{4.6}$$

and, consequently, the specific intensity is independent of time.

To examine the convergence properties of the spatial approximation for this problem, we examine the average intensity

$$\mathcal{U}(z) = \frac{1}{2}\int_{-1}^{1} I(z, \mu) d\mu \tag{4.7}$$

computed using the Gaussian quadrature rule used with the scattering operator. As with the temporal convergence study, we do not have a known analytical solution. Instead, we compare results to a highly resolved reference solution with $N = 513$ Chebyshev modes.

A plot of this spatial convergence study appears in Figure 4.3. We compute the 2-norm of the difference between a particular solution and the reference solution normalized by the 2-norm of the reference solution. In Figure 4.3 we observe the expected superalgebraic convergence of this method. In fact, the relative errors for Chebyshev modes larger than 65 are below the allowed accuracy of double floating-point arithmetic.

5. Summary and concluding remarks. We have presented a complete discussion of Chebyshev spectral methods for solving radiative transfer problems. In particular, we have examined problems in which we apply Chebyshev spectral methods to the spatial variable, Gaussian quadrature methods to the angular variable, and finite differences in the time variable. The resultant linear system of equations at each time step is sparse. This method easily handles homogeneous as well as continuously inhomogeneous and layered media. We have shown that these methods maintain the same structure and ease of implementation as the homogeneous problem. In addition, we show that it modifies easily to solve vector problems and has the potential to solve higher-dimensional problems efficiently.

For all of the cases presented here, we use the generalized deflated block elimination method to solve the bordered, block tridiagonal system of equations at each time step. This method is adequate in computing solutions for the problems we have examined here. However, it is noteworthy to mention that there is an obvious richness in the structure of this linear system beyond its sparsity (see Appendix A). It is quite possible that more can be done to take advantage of this structure to construct even more efficient methods for computing solutions. As we begin to implement this method for larger problems, examining this system in greater detail will become more important and hence will be the subject of future studies. Furthermore, we intend to use this method to examine polarization techniques for optical imaging problems in future work.

Appendix A. The generalized deflated block elimination method. To solve the system of equations (2.22) and (2.23) efficiently, let us construct two vectors that contain the unknown quantities,

$$X = \begin{bmatrix}\mathbf{A}_0, & \mathbf{A}_1, & \cdots, & \mathbf{A}_N\end{bmatrix}^T, \tag{A.1}$$

$$Y = \begin{bmatrix}\mathbf{a}_0\end{bmatrix}, \tag{A.2}$$

where the superscript T denotes the transpose. Then, we can compactly write (2.22) and (2.23) as

$$\begin{bmatrix} A & B \\ C^T & D \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} F \\ G \end{bmatrix}. \tag{A.3}$$

Here, A is the $(N+1)\times(N+1)$ block tridiagonal matrix composed by $q \times q$ matrices M and K

$$A = \begin{bmatrix} \mathrm{M} & 0 & & & & & & \\ \mathrm{K} & \mathrm{M} & -\frac{1}{2}\mathrm{K} & & & & & \\ & \frac{1}{4}\mathrm{K} & \mathrm{M} & -\frac{1}{4}\mathrm{K} & & & & \\ & & \ddots & \ddots & \ddots & & & \\ & & & \frac{1}{2k}\mathrm{K} & \mathrm{M} & -\frac{1}{2k}\mathrm{K} & & \\ & & & & \ddots & \ddots & \ddots & \\ & & & & & \frac{1}{2(N-1)}\mathrm{K} & \mathrm{M} & -\frac{1}{2(N-1)}\mathrm{K} \\ & & & & & & \frac{1}{2N}\mathrm{K} & \mathrm{M} \end{bmatrix}, \tag{A.4}$$

B is the $(N+1)\times 1$ block column matrix

$$B = \begin{bmatrix}\mathrm{K}, 0, \dots, 0\end{bmatrix}^T, \tag{A.5}$$

D is the $q \times q$ identity matrix, and C^T is the $1 \times (N+1)$ block row matrix

$$C^T = \left[\mathbb{I}_0, \quad \tfrac{1}{4}\mathbb{I}_1, \quad \cdots, \quad \tfrac{1}{2}\left(\tfrac{1}{k+1} - \tfrac{1}{k-1}\right)\mathbb{I}_k, \quad \cdots, \quad -\tfrac{1}{2(N-1)}\mathbb{I}_N\right], \tag{A.6}$$

where $\mathbb{I}_k$ is a $q \times q$ matrix made up of $\frac{1}{2}q \times \frac{1}{2}q$ submatrices

$$\mathbb{I}_k = \begin{bmatrix} \mathbb{I} & 0 \\ 0 & (-1)^{k+1}\mathbb{I} \end{bmatrix}. \tag{A.7}$$

The right-hand side vectors of this linear system are

$$F = \begin{bmatrix} \mathbf{f}_0\,, & \mathbf{f}_1\,, & \cdots, & \mathbf{f}_N \end{bmatrix}^T, \tag{A.8}$$

$$G = \begin{bmatrix} 0 \end{bmatrix}. \tag{A.9}$$

Each of the blocks in these matrices and vectors is $q \times q$. Therefore, (A.3) is a bordered, block tridiagonal system of equations.

By arranging the linear system of equations in this way, we can apply the generalized deflated block elimination method to solve this system [19]. This algorithm isolates the block tridiagonal part of this system from its borders, allowing one to exploit the sparsity. Contributions from the borders, B, C^T, and D come about as corrections to the block tridiagonal system solve. The generalized deflated block elimination algorithm is

1. solve $AW = B$,
2. solve $Aw = F$,
3. compute $S = D - C^T W$,
4. solve $SY = G - C^T w$,
5. compute $X = w - WY$.

In addition, the storage requirements in solving this system using this algorithm is very small. The diagonal blocks of A are diagonal matrices, and the off-diagonal blocks of A are all the same modulo a scalar factor. Therefore, only the $q \times q$ matrix, K, and a $q \times 1$ vector containing the diagonal elements of M are necessary to effectively store the matrix A. Furthermore, since each individual block in C^T are scalar multiples of identity matrices, computing the product of it and some other vector requires very few operations.

Appendix B. Alternating direction method. Replacing the continuous angular variables μ and ϕ by a set of discrete quadrature points μ_i and ϕ_j, where $i = 1, \ldots, q_1$ and $j = 1, \ldots, q_2$, we find that the semidiscrete spectral approximation (3.32) in matrix notation is

$$\frac{\partial \mathbf{a}_{kl}(t)}{\partial t} + \frac{2v}{d_z}\mathrm{L}_1 \mathbf{A}_{kl}(t) + \frac{2v}{d_x}\mathrm{L}_2 \mathbf{B}_{kl}(t) + v\mathrm{Q}[\mathbf{a}_{kl}](t) = v\mathbf{F}_{kl}(t) \quad \text{for } k, l = 0, \ldots, N, \tag{B.1}$$

with

$$\mathbf{a}_{kl}(t) = \left(a_{kl}(\mu_1, \phi_1, t), a_{kl}(\mu_1, \phi_2, t), \ldots, a_{kl}(\mu_1, \phi_q, t), \quad \ldots, \quad a_{kl}(\mu_{q_1}, \phi_{q_2}, t)\right), \tag{B.2}$$

and similar representations for $\mathbf{A}_{kl}$, $\mathbf{B}_{kl}$, and $\mathbf{F}_{kl}$. The $(q_1 q_2) \times (q_1 q_2)$ diagonal matrices L_1 and L_2 are defined by

$$\mathrm{L}_1 = \mathrm{diag}(\mu_1, \overset{q_2 times}{\cdots}, \mu_1, \mu_2, \overset{q_2 times}{\cdots}, \mu_2, \quad \ldots \quad \mu_{q_1}, \ldots, \mu_{q_1})$$
$$\mathrm{L}_2 = \mathrm{diag}(\mu_1 \cos\phi_1, \mu_1 \cos\phi_2, \ldots, \mu_1 \cos\phi_{q_2}, \quad \ldots \quad \mu_{q_1}\cos\phi_1, \ldots, \mu_{q_1}\cos\phi_{q_2}).$$

The Crank–Nicholson time-differencing scheme is given by

$$\begin{aligned}\text{(B.3)}\quad \left[\mathbb{I}+\frac{v\Delta t}{2}\mathrm{Q}\right]\mathbf{a}_{kl}^{n+1}+\mathrm{M}_1\mathbf{A}_{kl}^{n+1}+\mathrm{M}_2\mathbf{B}_{kl}^{n+1}\\ =\left[\mathbb{I}-\frac{v\Delta t}{2}\mathrm{Q}\right]\mathbf{a}_{kl}^{n}-\mathrm{M}_1\mathbf{A}_{kl}^{n}-\mathrm{M}_2\mathbf{B}_{kl}^{n}+\frac{v\Delta t}{2}\left[\mathbf{F}_k^{n+1}+\mathbf{F}_k^{n}\right],\end{aligned}$$

where $\mathrm{M}_1=v\Delta t\mathrm{L}_1/d_z$ and $\mathrm{M}_2=v\Delta t\mathrm{L}_2/d_x$. The expansion coefficients of the spatial derivatives B_{kl} and A_{kl} are related to a_{kl} through (3.33). Let us write formally the inverse relations of (3.33) as $A_{kl}=T_1a_{kl}$ and $B_{kl}=T_2a_{kl}$ so that we can make an approximate factorization of (B.3)

$$\begin{aligned}\text{(B.4)}\quad \left[\mathbb{I}+\frac{v\Delta t}{4}\mathrm{Q}+\mathrm{M}_1T_1\right]\left[\mathbb{I}+\frac{v\Delta t}{4}\mathrm{Q}+\mathrm{M}_2T_2\right]\mathbf{a}_{kl}^{n+1}\\ =\left[\mathbb{I}-\frac{v\Delta t}{4}\mathrm{Q}-\mathrm{M}_1T_1\right]\left[\mathbb{I}-\frac{v\Delta t}{4}\mathrm{Q}-\mathrm{M}_2T_2\right]\mathbf{a}_{kl}^{n}+\frac{v\Delta t}{2}\left[\mathbf{F}_k^{n+1}+\mathbf{F}_k^{n}\right].\end{aligned}$$

Observe that all the extra terms introduced in (B.4) are of order $O(\Delta t^2)$. Furthermore, the same terms are introduced in the left- and right-hand side. We can conclude that by subtraction of the extra terms in both sides of the equation, (B.4) is equivalent to (B.3) up to $O(\Delta t^3)$ (since $\mathbf{a}_{kl}^{n+1}-\mathbf{a}_{kl}^{n}\sim O(\Delta t)$). Then the two-stage method

$$\text{(B.5)}\quad \left[\mathbb{I}+\frac{v\Delta t}{4}\mathrm{Q}+\mathrm{M}_1T_1\right]\mathbf{a}_{kl}^{*}=\left[\mathbb{I}-\frac{v\Delta t}{4}\mathrm{Q}-\mathrm{M}_1T_1\right]\mathbf{a}_{kl}^{n}+\frac{v\Delta t}{2}\left[\mathbf{F}_k^{n+1}+\mathbf{F}_k^{n}\right],$$

$$\text{(B.6)}\quad \left[\mathbb{I}+\frac{v\Delta t}{4}\mathrm{Q}+\mathrm{M}_2T_2\right]\mathbf{a}_{kl}^{n+1}=\left[\mathbb{I}-\frac{v\Delta t}{4}\mathrm{Q}-\mathrm{M}_2T_2\right]\mathbf{a}_{kl}^{*}$$

solves (B.1). Again using that $A_{kl}=T_1a_{kl}$ and $B_{kl}=T_2a_{kl}$ we obtain the numerical scheme (3.34).

REFERENCES

[1] V. Borue and S. A. Orszag, *Self-similar decay of 3-dimensional homogeneous turbulence with hyperviscosity*, Phys. Rev. E, 51 (1995), pp. R856–R859.

[2] K. Black, *Spectral elements on infinite domains*, SIAM J. Sci. Comput., 19 (1998), pp. 1667–1681.

[3] M. D. Feit, J. A. Fleck, Jr., and A. Steiger, *Solution of the Schrödinger equation by a spectral method*, J. Comput. Phys., 47 (1982), pp. 412–433.

[4] R. Renaut and J. Frohlich, *A pseudospectral Chebyshev method for the 2D wave equation with domain stretching and absorbing boundary conditions*, J. Comput. Phys., 124 (1996), pp. 324–336.

[5] F. Ben Belgacem and M. Grundmann, *Approximation of the wave and electromagnetic diffusion equations by spectral methods*, SIAM J. Sci. Comput., 20 (1998), pp. 13–32.

[6] D. Gottlieb and S. A. Orszag, *Numerical Analysis of Spectral Methods: Theory and Applications*, SIAM, Philadelphia, 1977.

[7] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zhang, *Spectral Methods in Fluid Dynamics*, Springer-Verlag, Berlin, New York, 1988.

[8] A. Ishimaru, *Wave Propagation and Scattering in Random Media*, IEEE Press, New York, 1997.

[9] S. Chandrasekhar, *Radiative Transfer*, Dover, New York, 1960.

[10] L. Ryzhik, G. Papanicolaou, and J. B. Keller, *Transport equations for elastic and other waves in random media*, Wave Motion, 24 (1996), pp. 327–370.

[11] G. Bal and M. Moscoso, *Polarization effects of seismic waves on the basis of radiative transport theory*, Geophys. J. Int., 142 (2000), pp. 571–585.

[12] K. M. Case and P. F. Zweifel, *Linear Transport Theory*, Plenum Press, New York, 1967.

[13] M. Moscoso, J. B. Keller, and G. Papanicolaou, *Depolarization and blurring of optical images by biological tissues*, J. Opt. Soc. Amer. A, 18 (2001), pp. 948–960.

[14] B. Ritchie, P. G. Dykema, and D. Braddy, *Use of fast Fourier-transform computational methods in radiation transport*, Phys. Rev. E, 56 (1997), pp. 2217–2227.

[15] A. D. Kim and A. Ishimaru, *A Chebyshev spectral method for radiative transfer equations applied to electromagnetic wave propagation and scattering in discrete random media*, J. Comput. Phys., 152 (1999), pp. 264–280.

[16] O. Dorn, *A transport-backtransport method for optical tomography*, Inverse Problems, 14 (1998), pp. 1107–1130.

[17] L. M. Delves and J. L. Mohamed, *Computational Methods for Integral Equations*, Cambridge University Press, Cambridge, UK, 1985.

[18] E. A. Coutsias, T. Hagstrom, and D. Torres, *An efficient spectral method for ordinary differential equations with rational function coefficients*, Math. Comp., 65 (1996), pp. 611–635.

[19] T. F. Chan and D. C. Resasco, *Generalized deflated block-elimination*, SIAM J. Numer. Anal., 23 (1986), pp. 913–924.

[20] V. B. Kisslelev, L. Roberti, and G. Perona, *An application of the finite-element method to the solution of the radiative transfer equation*, J. Quant. Spectros. Radiat. Transfer, 51 (1994), pp. 603–614.

[21] H. B. Keller, *On the pointwise convergence of the discrete-ordinate method*, J. Soc. Indust. Appl. Math., 8 (1960), pp. 560–567.

SIAM J. SCI. COMPUT.
Vol. 23, No. 6, pp. 2095–2122

QUADRATIC CONVERGENCE FOR VALUING AMERICAN OPTIONS USING A PENALTY METHOD[*]

P. A. FORSYTH[†] AND K. R. VETZAL[‡]

Abstract. The convergence of a penalty method for solving the discrete regularized American option valuation problem is studied. Sufficient conditions are derived which both guarantee convergence of the nonlinear penalty iteration and ensure that the iterates converge monotonically to the solution. These conditions also ensure that the solution of the penalty problem is an approximate solution to the discrete linear complementarity problem. The efficiency and quality of solutions obtained using the implicit penalty method are compared with those produced with the commonly used technique of handling the American constraint explicitly. Convergence rates are studied as the timestep and mesh size tend to zero. It is observed that an implicit treatment of the American constraint does not converge quadratically (as the timestep is reduced) if constant timesteps are used. A timestep selector is suggested which restores quadratic convergence.

Key words. American option, penalty iteration, linear complementarity

AMS subject classifications. 65M12, 65M60, 91B28

PII. S1064827500382324

1. Introduction.

The valuation and hedging of financial option contracts is a subject of considerable practical significance. The holders of such contracts have the right to undertake certain actions so as to receive certain payoffs. The valuation problem consists of determining a fair price to charge for granting these rights. A related issue, perhaps of even more importance to practitioners, is how to hedge the risk exposures which arise from selling these contracts. An important feature of such contracts is the time when contract holders can exercise their rights. If this occurs only at the maturity date of the contract, the option is classified as "European." If holders can exercise any time up to and including the maturity date, the option is said to be "American." The value of a European option is given by the solution of the Black–Scholes PDE (see, e.g., [33]). An analytical solution can be obtained for cases with constant coefficients and simple payoffs. However, most options traded on exchanges are American. Such options must be priced numerically, even for constant coefficients and simple payoffs. Note also that the derivatives of the solution are of interest since they are used in hedging. More formally, the American option pricing problem can be posed as a time dependent variational inequality or a differential linear complementarity problem (LCP).

In current practice, the most common method of handling the early exercise condition is simply to advance the discrete solution over a timestep, ignoring the constraint, and then to apply the constraint explicitly. This has the disadvantage that the solution is in an inconsistent state at the beginning of each timestep (i.e.,

[*] Received by the editors December 11, 2000; accepted for publication (in revised form) October 12, 2001; published electronically February 27, 2002. This work was supported by the Natural Sciences and Engineering Research Council of Canada, the Social Sciences and Humanities Research Council of Canada, the Royal Bank of Canada, and the Financial Industry Solutions Center (FISC), an SGI/Cornell University Joint Venture. In addition, Rational Software provided access to the Purify software development tool under the SEED program.

http://www.siam.org/journals/sisc/23-6/38232.html

[†] Department of Computer Science, University of Waterloo, Waterloo, ON, Canada N2L 3G1 (paforsyt@elora.math.uwaterloo.ca).

[‡] School of Accountancy, University of Waterloo, Waterloo, ON, Canada N2L 3G1 (kvetzal@watarts.uwaterloo.ca).

a discrete form of the LCP is not approximately satisfied). As well, this approach can obviously be only first order correct in time. On the other hand, this explicit application of the constraint is computationally very inexpensive.

Another common technique is to solve the discrete LCP using a relaxation method [33]. In terms of complexity, this method is particularly poor for pricing problems with one space-like dimension. A lower bound for the number of iterations required to solve the LCP to a given tolerance with a relaxation method would be the number of iterations required to solve the unconstrained problem using a preconditioned conjugate gradient method. Assuming that the mesh spacing in the asset price S direction is $O(\Delta S)$ and that the timestep size is $O(\Delta t)$, then the condition number of a discrete form of the parabolic option pricing PDE is $O\left[\Delta t/(\Delta S)^2\right]$. Let N be the number of timesteps. If we assume that $\Delta S = O(\Delta t) = 1/N$, then the number of iterations required per timestep would be $O(N^{1/2})$.

A multigrid method has been suggested in [5] to accelerate convergence of the basic relaxation method. Although this is a promising technique, multigrid methods are usually strongly coupled to the type of discretization used, and hence they are complex to implement in general purpose software.

There are a large number of general purpose methods for solving linear complementarity problems [22, 7, 25]. We can divide these methods into essentially two categories: direct methods, such as pivoting techniques [7], and iterative methods, such as Newton iteration [25] and interior point algorithms [22].

Some of these methods which have been applied specifically to American option pricing include linear programming [9], pivoting methods, [14], and interior point methods [15]. As pointed out in [15], pivoting methods (such as Lemke's algorithm [7]) and LP approaches are not well equipped to handle sparse systems, especially in more than one dimension (multifactor options).

Complementarity problems (both linear and nonlinear) can be posed in the form of a set of nonlinear equations. Various nonsmooth Newton methods have been suggested for these types of problems [26, 27, 11, 19, 17]. More recently, combinations of nonsmooth Newton and smoothing methods have been proposed [20].

It is well known that an LCP (or, equivalently, a variational inequality) can be solved by a penalty method [10, 30, 24, 12, 8]. In this article, we will explore some aspects of using penalty methods for pricing American options. We will restrict attention to one dimensional problems, which are more amenable to analysis. However, we have successfully used penalty methods for two factor (two dimensional) problems [40, 39]. In this work, the nonlinear discrete penalized equations are solved using Newton iteration. Another approach which also uses a Newton method has been suggested in [6]. Note that relaxation methods are frequently used to solve the discrete penalized nonlinear equations [8].

The advantage of the penalty method is that a single technique can be used for one dimensional or multidimensional problems, and standard sparse matrix software can be used to solve the Jacobian matrix. This technique can be used for any type of discretization, in any dimension, and on unstructured meshes. In particular, there is no difficulty in handling cases where the early exercise region is multiply-connected, as in [40]. As well, a single method can be used to handle American options and other nonlinearities, such as uncertain volatility and transaction cost models [33, 1]. In addition, nonlinearities due to the use of flux limiters for drift-dominated problems [37] can also be handled easily.

The objective of this article is to analyze the properties of penalty methods for the solution of a discrete form of a comparatively simple problem: a single factor

American option. In this way, we hope to gain some insight into the use of penalty methods for more complex problems. A single factor American option can be posed as a variational inequality, which in turn can be expressed as a discrete LCP problem at each timestep. However, it is important to examine the penalty method in the context of the overall problem: we need to find time accurate solutions to the variational inequality. Consequently, we can expect that we have a good initial guess for the solution of the variational inequality from the previous timestep. In fact, it is important to recall that simply advancing the solution for a timestep (ignoring the constraint) and applying the constraint in an explicit fashion will solve the time dependent variational inequality to $O(\Delta t)$. Consequently, any method used to solve the LCP at each step should take full advantage of a good initial guess.

We will also study the convergence of these methods as the timestep and mesh size are reduced to zero. We will determine sufficient conditions for monotone convergence of the penalty method in one dimension. It may be possible to require weaker conditions, perhaps using the methods in [24]. However, option pricing problems are typically degenerate parabolic and in nonconservative form. This can be expected to complicate the methods in [24].

In practice, we observe that the penalty method works well for multifactor options [40, 39] and for nonlinear problems. In other words, although the conditions we derive are sufficient, they do not appear to be necessary. Consequently, it appears that the penalty method can be used for more general situations. In addition, we will compare the penalty method (where the LCP is approximately solved at each timestep) with an explicit technique for handling the American constraint.

Essentially, the method proposed in this work uses a nonsmooth Newton iteration [4] to solve the penalized problem. A disadvantage of the penalty method as formulated in this work is that the American constraint is satisfied only approximately, but since this error can be easily made to be much smaller than the discretization error, this does not appear to be a practical disadvantage. On the other hand, the advantages of this approach are (under certain conditions) as follows:

- This method has finite termination (in exact arithmetic); i.e., for an iterate sufficiently close to the solution, the algorithm terminates in one iteration. This is especially advantageous when dealing with American option pricing, since we have an excellent initial guess from the previous timestep. In fact, as we shall see, for typical grids and timesteps, the algorithm takes, on average, less than two iterations per timestep to converge. Finite termination also implies that the number of iterations required for convergence is insensitive to the size of the penalty factor (until the limits of machine precision is reached).
- The iteration is globally convergent using full Newton steps.

Of course, if we did not have the advantage of having a good initial guess, a pivoting method [7] can be very efficiently implemented if the coefficient matrix is a tridiagonal M-matrix. We emphasize here that the penalty method should not be regarded as a general purpose method for LCP problems. The real advantage of the penalty method is that this technique takes full advantage of the fact that a good initial iterate is available, and we take full advantage of sparsity, which is important in multifactor problems. Another approach which attempts to take advantage of a good initial iterate combined with standard sparse solvers is described in [21].

If we solve the LCP at each step (using the penalty approach), and if constant timesteps are used, we observe that second order convergence is not obtained as the timesteps and mesh size tend to zero. This phenomenon can be explained by examining the asymptotic behavior of the solution near the exercise boundary. A timestep selector is developed which restores second order convergence.

Asymptotically, the second order method is superior to the commonly used binomial lattice technique [16]. However, it is of practical interest to determine at what levels of accuracy a second order PDE method will be computationally more efficient than the lattice method. We present numerical comparisons to assist in this determination.

In the following we will restrict attention to the American put option. However, these methods can be applied to dividend paying calls, as well as complex options which involve solving a set of one dimensional American-type problems embedded in a higher dimensional space. Examples of these types of options include discretely observed Asian options [38], Parisian options [31], shout options [36], and segregated fund guarantees [35].

2. Formulation.

Consider an asset with price S which follows the stochastic process

$$dS = \mu S dt + \sigma S dz, \tag{2.1}$$

where μ is the drift rate, σ is volatility, and dz is the increment of a Wiener process. We wish to determine the value $V(S,t)$ of an American option where the holder can exercise at any time and receive the payoff $V^*(S,t)$. Denote the expiry time of the option by T, and let $\tau = T - t$. Then the American pricing problem can be formally stated as an LCP [33]

$$\begin{aligned} \mathcal{L}V &\geq 0, \\ (V - V^*) &\geq 0, \\ (\mathcal{L}V = 0) &\vee (V - V^* = 0), \end{aligned} \tag{2.2}$$

where the notation $(\mathcal{L}V = 0) \vee (V - V^* = 0)$ denotes that either $(\mathcal{L}V = 0)$ or $(V - V^* = 0)$ at each point in the solution domain, and

$$\mathcal{L}V \equiv V_\tau - \left(\frac{\sigma^2}{2} S^2 V_{SS} + rSV_S - rV \right), \tag{2.3}$$

and r is the risk free rate of interest. A put option is a contract which gives the holder the right to sell the asset for K (known as the “strike”). A call option is similar except that the holder has the right to buy the asset for K. The payoff for a put is

$$V^*(S) = V(S, \tau = 0) = \max(K - S, 0). \tag{2.4}$$

The boundary conditions are

$$V(S,\tau) = 0, \qquad S \to \infty, \tag{2.5}$$

$$\mathcal{L}V = V_\tau - rV, \qquad S \to 0. \tag{2.6}$$

Condition (2.5) follows from the payoff (2.4), while (2.6) is obvious given (2.3).

3. The penalty method.

The basic idea of the penalty method is simple. We replace problem (2.2) by the nonlinear PDE [10]

$$V_\tau = \frac{\sigma^2}{2} S^2 V_{SS} + rSV_S - rV + \rho \max(V^* - V, 0), \tag{3.1}$$

where, in the limit as the positive penalty parameter $\rho \to \infty$, the solution satisfies $V \geq V^*$.

4. Discretization. We will now discretize (3.1) and select a suitable form for the discrete penalty term. Let $V(S_i, \tau_n) = V_i^n$ be the discrete solution to (3.1) at asset value S_i and time (going backwards) τ_n. Applying a standard finite volume method with variable timeweighting [40] then gives

$$\mathcal{F}V_i^{n+1} = q_i^{n+1}, \tag{4.1}$$

where

$$\begin{aligned}\mathcal{F}V_i^{n+1} \equiv{} & A_i\left(\frac{V_i^{n+1} - V_i^n}{\Delta\tau}\right)\\ & + (1-\theta)\left(\sum_{j\in\eta_i}\gamma_{ij}(V_j^{n+1} - V_i^{n+1}) + \sum_{j\in\eta_i}\vec{L_{ij}}\cdot\mathbf{U}_i V_{ij+1/2}^{n+1} - A_i r V_i^{n+1}\right)\\ & + \theta\left(\sum_{j\in\eta_i}\gamma_{ij}(V_j^n - V_i^n) + \sum_{j\in\eta_i}\vec{L_{ij}}\cdot\mathbf{U}_i V_{ij+1/2}^{n} - A_i r V_i^{n}\right).\end{aligned} \tag{4.2}$$

Fully implicit and Crank–Nicolson discretizations correspond to cases of $\theta = 0$ and $\theta = 1/2$, respectively, and

$$\begin{aligned} A_i &= (S_{i+1} - S_{i-1})/2,\\ \eta_i &= \{i-1, i+1\},\\ \Delta\tau &= \tau^{n+1} - \tau^n,\\ \gamma_{ij} &= \frac{\sigma^2 S_i^2}{2|S_j - S_i|},\\ V_{ij+1/2}^{n+1} &= \text{ value of } V \text{ at the face between nodes } i \text{ and } j,\\ \mathbf{U}_i &= (-rS_i)\hat{\mathbf{i}},\\ \vec{L_{ij}} &= \begin{cases} -\hat{\mathbf{i}} & \text{if } j = i+1,\\ +\hat{\mathbf{i}} & \text{if } j = i-1,\end{cases}\\ \hat{\mathbf{i}} &= \text{ unit vector in the positive } S \text{ direction.}\end{aligned} \tag{4.3}$$

The discrete penalty term q_i^{n+1} in (4.1) is given by

$$q_i^{n+1} = \begin{cases} (A_i/\Delta\tau)(V_i^* - V_i^{n+1})\mathit{Large} & \text{if } V_i^{n+1} < V_i^*,\\ 0 & \text{otherwise,}\end{cases} \tag{4.4}$$

where *Large* is the penalty factor. (This will be related to the desired convergence tolerance in section 4.1.) The face value $V_{ij+1/2}^{n+1}$ can be evaluated using either central weighting or, to ensure nonoscillatory solutions, a flux limiter [37]

$$V_{ij+1/2}^{n+1} = \begin{cases} (V_i^{n+1} + V_j^{n+1})/2 & \text{if } \gamma_{ij}^{n+1} + \vec{L_{ij}}\cdot\mathbf{U}_i/2 > 0,\\ \text{second order flux limiter [37]} & \text{otherwise.}\end{cases} \tag{4.5}$$

In general, for standard options with typical values for σ, r, central weighting can be used at most nodes (except perhaps as $S \to 0$). In order to determine sufficient conditions for the convergence of the nonlinear iteration for the penalized American equation, we require that the coefficients of the discrete equations have certain properties. We will ensure that these conditions are satisfied by using central or upstream weighting. (In practice, we have observed that even if these conditions are not met, convergence of the penalty method is still rapid [40].) If we use central or upstream weighting in the following, then (4.1) becomes

$$
\begin{aligned}
V_i^{n+1} - V_i^n = (1-\theta)&\left(\Delta\tau \sum_{j\in\eta_i} \left(\bar{\gamma}_{ij} + \bar{\beta}_{ij}\right)\left(V_j^{n+1} - V_i^{n+1}\right) - r\Delta\tau V_i^{n+1}\right)\\
&+ \theta\left(\Delta\tau \sum_{j\in\eta_i} \left(\bar{\gamma}_{ij} + \bar{\beta}_{ij}\right)\left(V_j^{n} - V_i^{n}\right) - r\Delta\tau V_i^{n}\right)\\
&+ P_i^{n+1}(V_i^* - V_i^{n+1}),
\end{aligned}
\tag{4.6}
$$

where

$$
P_i^{n+1} = \begin{cases} \mathit{Large} & \text{if } V_i^{n+1} < V_i^*,\\ 0 & \text{otherwise,}\end{cases}
\tag{4.7}
$$

and where

$$
\begin{aligned}
\bar{\gamma}_{ij} &= \frac{\sigma^2 S_i^2}{2A_i|S_j - S_i|},\\
\bar{\beta}_{ij} &= \begin{cases} \vec{L_{ij}} \cdot \mathbf{U}_i/2A_i & \text{if } \gamma_{ij} + \vec{L_{ij}} \cdot \mathbf{U}_i/2 \geq 0,\\ \max(\vec{L_{ij}} \cdot \mathbf{U}, 0)/A_i & \text{otherwise.}\end{cases}
\end{aligned}
$$

For future reference, we can write the discrete equations (4.6) in matrix form. Let $V^{n+1} = \left[V_0^{n+1}, V_1^{n+1}, \ldots, V_m^{n+1}\right]'$, $V^n = [V_0^n, V_1^n, \ldots, V_m^n]'$, $V^* = [V_0^*, V_1^*, \ldots, V_m^*]'$, and

$$
[\hat{M}V^n]_i = -\left(\Delta\tau \sum_{j\in\eta_i} \left(\bar{\gamma}_{ij} + \bar{\beta}_{ij}\right)\left(V_j^{n} - V_i^{n}\right) - r\Delta\tau V_i^{n}\right).
\tag{4.8}
$$

Note that the first and last rows of $\hat{M}$ will have to be modified to take into account the boundary conditions. (An obvious method for applying conditions (2.5)–(2.6) results in the first and last rows of $\hat{M}$ being identically zero except for positive entries on the diagonal). In the following, we will assume that upstream and central weighting are selected so that $\bar{\gamma}_{ij} + \bar{\beta}_{ij} \geq 0$. This implies that the matrix $\hat{M}$ is an M-matrix, i.e., a diagonally dominant matrix with positive diagonals and nonpositive off-diagonals. Note that all of the elements of the inverse of an M-matrix are nonnegative.

Let the diagonal matrix $\bar{P}$ be given by

$$
\bar{P}(V^{n+1})_{ij} = \begin{cases} \mathit{Large} & \text{if } V_i^{n+1} < V_i^* \text{ and } i = j,\\ 0 & \text{otherwise.}\end{cases}
\tag{4.9}
$$

We can then write the discrete equations (4.6) as

$$
\left[I + (1-\theta)\hat{M} + \bar{P}(V^{n+1})\right] V^{n+1} = \left[I - \theta\hat{M}\right] V^n + \left[\bar{P}(V^{n+1})\right] V^*.
\tag{4.10}
$$

4.1. Solution of the discrete LCP. The discrete form of the LCP (2.2) can be written as

$$\begin{aligned} &\mathcal{F}V_i^{n+1} \geq 0, \\ &V_i^{n+1} - V_i^* \geq 0, \\ &(\mathcal{F}V_i^{n+1} = 0) \vee (V_i^{n+1} - V_i^* = 0), \end{aligned} \tag{4.11}$$

where $\mathcal{F}$ is given by (4.2). On the other hand, the discrete solution of the penalized problem (4.10) has the property that either

$$V_i^{n+1} - V_i^* \geq 0 \qquad (\Rightarrow q_i^{n+1} = 0 \text{ and } \mathcal{F}V_i^{n+1} = 0) \tag{4.12}$$

or

$$V_i^{n+1} - V_i^* \leq 0 \qquad (\Rightarrow q_i^{n+1} > 0 \text{ and } \mathcal{F}V_i^{n+1} > 0). \tag{4.13}$$

However, from (4.11) the exact solution of the discrete LCP has $V_i^{n+1} - V_i^* = 0$ at those nodes where $\mathcal{F}V_i^{n+1} > 0$. In order to obtain an approximate solution of (4.11) with an arbitrary level of precision, we need to show that the solution of (4.10) satisfies $V_i^{n+1} - V_i^* \to 0$ as *Large* $\to \infty$ for nodes where $\mathcal{F}V_i^{n+1} > 0$. This follows if we can show that the term

$$P_i^{n+1}(V_i^* - V_i^{n+1}) \tag{4.14}$$

in (4.6) is bounded independent of *Large*. It is also desirable that the bound be independent of the timestep and mesh spacing, so that *Large* can be chosen without regard to grid and timestep size. In Appendix A we determine sufficient conditions which allow us to bound (4.14). The results can be summarized as the following theorem.

THEOREM 4.1 (error in the penalty formulation of the discrete LCP). *Under the assumptions that the matrix $\hat{M}$ in* (4.10) *is an M-matrix and that timesteps are selected so that*

$$1 - \theta\left(\Delta\tau \sum_{j\in\eta_i} \left(\bar{\gamma}_{ij} + \bar{\beta}_{ij}\right) + r\Delta\tau\right) \geq 0, \tag{4.15}$$

$$\frac{\Delta\tau}{\Delta S} < \text{const. } \Delta\tau, \Delta S \to 0, \tag{4.16}$$

$$\Delta S = \min_i S_{i+1} - S_i,$$

then the penalty method for the American put ((4.10) *with terminal condition* (2.4)) *solves*

$$\mathcal{F}V_i^{n+1} \geq 0, \tag{4.17}$$

$$V_i^{n+1} - V_i^* \geq -\frac{C}{Large}, \qquad C > 0, \tag{4.18}$$

$$(\mathcal{F}V_i^{n+1} = 0) \vee \left(|V_i^{n+1} - V_i^*| \leq \frac{C}{Large}\right), \tag{4.19}$$

where C is independent of Large, $\Delta\tau$, and ΔS.

Note that condition (4.16) is not practically restrictive, since violation of this condition would result in an imbalance between time and space discretization errors.

Condition (4.15) is trivially satisfied for any timestep size if a fully implicit method ($\theta = 0$) is used. However, if Crank–Nicolson timestepping is used, condition (4.15) essentially requires the boundedness of $\Delta\tau/(\Delta S)^2$. This timestep condition arises since we require that $(I - \theta\hat{M})V^n$ be bounded. This is essentially a requirement on the smoothness of the discrete solution using Crank–Nicolson timestepping. Note that for pure parabolic problems, careful analysis is required [28] to obtain quadratic convergence estimates for Crank–Nicolson methods (without restrictions on $\Delta\tau/(\Delta S)^2$). In fact, in [28], it is necessary to take a finite number of fully implicit steps initially in order to smooth rough data. This approach will be used in our numerical examples and will be discussed in detail in later sections.

In our numerical experiments, we routinely violate condition (4.15). As a check on the solution, we monitor the quantity

$$\text{max American error } = \max_{n,i} \frac{\max[0, (V_i^* - V_i^n)]}{\max(1, V_i^*)}. \tag{4.20}$$

This is a measure of the maximum relative error in enforcing the American constraint using the penalty method. This quantity will be small if the quantity (4.14) is bounded, and *Large* is sufficiently large. As long as we use the modification to Crank–Nicolson timestepping suggested in [28], we observe that the a posteriori error check (4.20) is indeed small.

It remains an open question if we can remove condition (4.15) for the timestepping method suggested in [28] (Crank–Nicolson with a finite number of fully implicit steps).

In practice, we can use the following heuristic argument to estimate the size of *Large* in terms of the relative accuracy required. In (4.10), suppose that $(1-\theta)\hat{M}V^{n+1}$ and $(I - \theta\hat{M})V^n$ are bounded independent of *Large*. Then, as $\mathit{Large} \to \infty$, (4.10) reduces to

$$V_i^{n+1} \simeq \left(\frac{\mathit{Large}}{1 + \mathit{Large}} \right) V_i^* \tag{4.21}$$

for nodes where $V_i^{n+1} < V_i^*$. If $V_i^* \neq 0$, then we have

$$\left| \frac{V_i^{n+1} - V_i^*}{V_i^*} \right| \simeq \frac{1}{\mathit{Large}}. \tag{4.22}$$

Therefore, if we require that the LCP be computed with a relative precision of *tol* for those nodes where $V_i^{n+1} < V_i^*$, then we should have $\mathit{Large} \simeq 1/\mathit{tol}$.

Note that in theory, if we are taking the limit as $\Delta S, \Delta\tau \to 0$, then we should have

$$\mathit{Large} = O\left(\frac{1}{\min\left[(\Delta S)^2, (\Delta\tau)^2\right]} \right). \tag{4.23}$$

This would mean that any error in the penalized formulation would tend to zero at the same rate as the discretization error. However, in practice it seems easier (to us at any rate) to specify the value of *Large* in terms of the required accuracy. In other words, we specify the maximum allowed error in the discrete penalized problem. We then reduce $\Delta S, \Delta\tau$ until the discretization error is reduced to this level of accuracy.

5. Penalty iteration. We will use Newton iteration to solve the discrete nonlinear equations (4.10). Of course, due regard must be paid to the discontinuous derivative which appears in the penalty term. More formally, we are solving the nonsmooth equation (4.10) using a generalized Newton iteration [4, 27, 25].

We will define the derivative of the penalty term, which is required in the Newton iteration as

$$\frac{\partial P_i^{n+1}(V_i^* - V_i^{n+1})}{\partial V_i^{n+1}} = \begin{cases} -Large & \text{if } V_i^{n+1} < V_i^*, \\ 0 & \text{otherwise,} \end{cases} \tag{5.1}$$

which is a particular choice of a member of the generalized Jacobian of (4.10).

Consequently, a generalized Newton iteration applied to (4.10) yields the following algorithm. Let $(V^{n+1})^k$ be the kth estimate for V^{n+1}. For notational convenience, we will define $\bar{P}^k \equiv \bar{P}\left((V^{n+1})^k\right)$ and $\bar{V}^k \equiv (V^{n+1})^k$. If $\bar{V}^0 = V^n$, then we have the following algorithm.

Penalty American Constraint Iteration.

For $k = 0, \ldots$ until convergence

$$\left[I + (1-\theta)\hat{M} + \bar{P}^k\right] \bar{V}^{k+1} = \left[I - \theta\hat{M}\right] V^n + \bar{P}^k V^* \tag{5.2}$$

$$\text{if } \left[\max_i \frac{|\bar{V}_i^{k+1} - \bar{V}_i^k|}{\max(1, |\bar{V}_i^{k+1}|)} < tol\right] \text{ or } \left[\bar{P}^{k+1} = \bar{P}^k\right] \quad \text{quit}$$

EndFor.

It is worthwhile at this point to determine the complexity of the above iteration compared to an explicit evaluation of the American constraint. Assuming that all of the coefficients are stored, that Crank–Nicolson timestepping is used with nonconstant timesteps, and that there are I nodes in the S direction, the first iteration of the penalty algorithm requires the following:

(i) $6I$ multiplies to evaluate the right-hand side of (5.2), where we have made the pessimistic assumption that $\bar{P}^k V^*$ requires I multiplies. This step also determines the entries in $\hat{M}$, assuming all possible quantities are precomputed and stored.

(ii) $2I$ multiply/divides to factor the matrix in (5.2).

(iii) $3I$ multiply/divides for the forward and back solve.

(iv) I divides for the convergence test. (This is also pessimistic, since we can skip the test on the first iteration, or if no constraint switches have occurred.)

This gives a total of $12I$ multiply/divides for the first penalty iteration and $7I$ multiply/divides for subsequent iterations.

If constant timesteps are used, $4I$ multiplies are needed to evaluate the right-hand side of (5.2), leading to a total of $10I$ multiply/divides for the first penalty iteration and $7I$ multiply/divides for subsequent iterations.

If an explicit method is used to evaluate the constraint, then there is only one matrix solve per timestep. To be precise here, an explicit method for handling the constraint is the following algorithm.

Explicit American Constraint Timestep.

$$\left[I + (1-\theta)\hat{M}\right] \hat{V}^{n+1} = \left[I - \theta\hat{M}\right] V^n, \tag{5.3}$$

$$V^{n+1} = \max(\hat{V}^{n+1}, V^*).$$

For constant timesteps (assuming that all coefficients are precomputed and stored),

(i) $3I$ multiply/divides are required to evaluate the right-hand side of (5.3), assuming that $\bar{P} = 0$;

(ii) assuming that the matrix is factored once and the factors stored, $3I$ multiply/divides are required for the forward and back solve;

giving a total of $6I$ multiply/divides per timestep. For nonconstant timesteps,

(i) $5I$ multiply/divides are required to evaluate the right-hand side of (5.3), assuming that $\bar{P} = 0$;

(ii) $2I$ multiply/divides are required to factor the matrix;

(iii) $3I$ multiply/divides are required for the forward and back solve;

giving a total of $10I$ multiply/divides per timestep.

6. Convergence of the penalty iteration.

Recall that the basic penalty algorithm (5.2) can be written as

$$\left[I + (1-\theta)\hat{M} + \bar{P}^k\right]\bar{V}^{k+1} = \left[I - \theta\hat{M}\right]V^n + \bar{P}^k V^*. \tag{6.1}$$

In Appendix B, we prove the following result.

THEOREM 6.1 (convergence of the nonlinear iteration of the penalized equations). *Under the assumptions that the matrix $\hat{M}$ in* (4.10) *is an M-matrix, then*

- *the nonlinear iteration* (5.2) *converges to the unique solution to* (4.10) *for any initial iterate $\bar{V}^0$;*
- *the iterates converge monotonically, i.e., $\bar{V}^{k+1} \geq \bar{V}^k$ for $k \geq 1$;*
- *the iteration has finite termination; i.e., for an iterate sufficiently close to the solution of the penalized problem* (4.10)*, convergence is obtained in one step.*

In [40], it was demonstrated experimentally that using a smooth form of the penalty function (4.4) did not aid convergence of the solution of the nonlinear equations. Intuitively, this is somewhat surprising. It might be expected that the nonsmooth penalty function (4.4), which has a discontinuous derivative, might cause oscillations during the iterations. However, the above result concerning monotonic convergence explains why the penalty iteration works so well, even with a nonsmooth derivative. Since $\bar{V}^{k+1} \geq \bar{V}^k$ for $k \geq 1$, in the worst case we have $\bar{V}_i^0 \geq V_i^*$, $\bar{V}_i^1 < V_i^*$, $\bar{V}_i^p > V_i^*$ for some $p \geq 2$. No further constraint switches will occur. In other words, for any given node, the iteration will not oscillate between $\bar{V}_i^k > V_i^*$ and $\bar{V}_i^{k+1} < V_i^*$ ($k \geq 1$). Note that $\bar{V}^0$ can be arbitrary but that $\bar{V}^1$ is given by the solution to (6.1). After $\bar{V}^1$ is determined, the iterates increase monotonically.

It is interesting to observe the connection between the penalty iteration and a pivoting method for solving the LCP. Let the set of all nodes be denoted by ν. If we let the set of nodes i where V_i^{n+1} satisfy $\mathcal{F}V_i^{n+1} = 0$ be denoted by χ, then as pointed out in [7], we can regard a pivoting method as a technique for determining χ in a systematic way. Once χ is known, then we can order the nodes $i \in \chi$ first, and those nodes in $\nu - \chi$ last, and solve the resulting system. In the case where $\hat{M}$ is an M matrix, then the pivoting method [7] becomes very simple. Let χ^k be the set of nodes where $\mathcal{F}(V_i^{n+1})^k = 0$. Suppose that at the k'th pivoting step a node is placed in χ^k. Any further pivoting operations will not remove that node from $\chi^p, p > k$. In the terminology of LCP algorithms, once a node has become basic, it will never become nonbasic as the algorithm proceeds. In this case, it is clear the pivoting algorithm terminates in at most number of pivoting steps equal to the size of the matrix.

Each iteration of the penalty method carries out a sorting step. Those nodes in χ^k are labelled by having $P_i = 0$. If the coefficient matrix is an M-matrix, once a node is placed in χ^k, it remains in $\chi^p, p > k$ on subsequent iterations. (Note that this is true only for $k \geq 1$.) The next iteration simply moves nodes in $\nu - \chi^k$ to χ^{k+1} or terminates. This property would hold for a problem in any dimension, provided the coefficient matrix is an M-matrix. If we have a good estimate for χ^0, and the number of nodes which move into or out of χ^0 is small, then the convergence of the penalty method will be rapid.

There also appears to be a connection between the approach used here and in [21]. In [21], each iteration consists of obtaining an initial estimate of constrained nodes using a few projected relaxation iterations; then the remaining nodes are solved using a sparse iterative method. It is argued in [21] that this is a good method if a good estimate of the constrained nodes is available. In some sense the penalty method combines these two steps, since an outcome of the sparse solve in the penalty iteration is an indication of whether or not a node is still constrained.

More interestingly, we observe that the penalty iteration is rapidly convergent for multifactor options with nonzero correlation [39]. The discretized equations in this case are not M-matrices, and in the convection dominated case, the discrete equations are nonlinear.

7. Numerical examples. In order to carry out a careful convergence study, we need to take into consideration the fact that the payoff function (2.4) has only piecewise smooth derivatives. This can cause problems if Crank–Nicolson timestepping is used. Specifically, oscillatory solutions can be generated [41]. For example, if we consider a simple European put option, then we know that the asymptotic solution near the expiry time $\tau = 0$ and close to the strike K is [33]

$$\text{put value } = O\left(\tau^{1/2}\right). \tag{7.1}$$

This would suggest that $V_{ttt} = O(\tau^{-5/2})$. The *local* finite difference truncation error for a Crank–Nicolson step (near $\tau = 0$) would then be $O\left[V_{ttt}(\Delta\tau)^3\right]$. If we set $\tau = \Delta\tau$ (the first step), then the local error would be $O[(\Delta\tau)^{1/2}]$, resulting in poor convergence. Fortunately, this analysis is a bit too simplistic. The behavior of the solution in (7.1) is due to the nonsmooth payoff near K, which causes V_{SS} to behave (near $\tau = 0, S = K$) as $O\left(\tau^{-1/2}\right)$ [33]. This large value of V_{SS} causes a very rapid smoothing effect due to the parabolic nature of the PDE. Consequently, if an appropriate timestepping method is used, we can expect the initial errors to be damped very quickly. However, there is a problem with Crank–Nicolson timestepping. Crank–Nicolson is only *A-stable*, not *strongly A-stable*. This means that some errors are damped very slowly, resulting in oscillations in the numerical solution.

Since a finite volume discretization in one dimension can be viewed as a special type of finite element discretization, we can appeal to the finite element analysis in [28]. This analysis was specifically directed towards the case of parabolic PDEs with nonsmooth initial conditions. Essentially, in [28] it is shown that if we take constant timesteps with a Crank–Nicolson method, then second order convergence (in time) can be guaranteed if (i) after each nonsmooth initial state, we take *two* fully implicit timesteps and then use Crank–Nicolson thereafter (payoffs with discontinuous derivatives qualify as nonsmooth); and (ii) the initial conditions are l_2 projected onto the space of basis functions. In our case, this means that the initial condition should be approximated by continuous, piecewise linear basis functions.

However, consider the case of a simple payoff such as that for a put option. Although this has a discontinuous derivative at K, no smoothing is required provided

TABLE 1

Value of a European put, $\sigma = .8$, $T = .25$, $r = .10$, $K = 100$, $S = 100$. Exact solution (to seven figures): 14.45191. *Change is the difference in the solution from the coarser grid. Ratio is the ratio of the changes on successive grids.*

		No smoothing			Rannacher smoothing		
Nodes	Timesteps	Value	Change	Ratio	Value	Change	Ratio
68	25	14.50470			14.41872		
135	50	14.41032	.09438		14.44357	.02485	
269	100	14.43238	.02215	4.3	14.44982	.00625	4.0
539	200	14.44246	.01008	2.2	14.45138	.00156	4.0
1073	400	14.44726	.00480	2.1	14.45177	.00039	4.0

we have a node at K. This is because we have a piecewise linear representation of the initial condition, consistent with the implied basis functions used in the finite volume method. In the case of a discontinuous initial condition, smoothing is necessary since this is not in the space of continuous piecewise linear basis functions. Finally, we remark that although second order convergence does not guarantee that the solution is nonoscillatory, in practice the above methods work well.

We can demonstrate the effectiveness of the simple idea of taking two fully implicit methods at the start and Crank–Nicolson thereafter (which we will henceforth refer to as *Rannacher smoothing* [28]) for a European put option with known solution. We will use the rather extreme value of $\sigma = .8$ for illustrative purposes. Results are provided in Table 1, which demonstrates that the solution with no smoothing converges erratically as the grid spacing and timestep size are reduced. In contrast, the smoothed solution shows quadratic convergence.

The reason for the poor convergence of the nonsmoothed runs can be explained by examining plots of the value V, delta (V_S), and gamma (V_{SS}), as shown in the left side of Figure 1. (Recall that it is of practical importance to determine delta and gamma for hedging purposes [16]). Note that although the value appears smooth, oscillations appear in delta (near the strike) and are magnified in gamma. The same problem was run using Rannacher smoothing, and the results are shown in the right side of Figure 1. The oscillations in delta and gamma have disappeared. All subsequent runs will use Rannacher smoothing.

It might appear appropriate to use a timestepping method with better error damping properties, such as a second order BDF method [2]. However, our experience with this method for complex American style problems (see [34]) was poor. We conjecture that this is due to a lack of smoothness in the time direction, causing problematic behavior for multistep methods. This effect will be addressed in some detail below.

8. Implicit and explicit handling of the American constraint. We will now compare an implicit treatment of the American constraint (using the penalty technique) with an explicit treatment (see pseudocode (5.3)). In these examples we use constant timesteps, a convergence tolerance of $tol = 10^{-6}$ (see pseudocode (5.2)), and consequently a value of $Large = 10^6$.

Two volatility values were used in these examples: $\sigma = .2, .8$. We truncate the computational domain at $S = S_{\max}$, where condition (2.5) is applied. The grid for $\sigma = .2$ used $S_{\max} = 200$, while the grid for $\sigma = .8$ used $S_{\max} = 1000$. Both grids were identical for $0 < S < 200$. The grid for $\sigma = .8$ added additional nodes for $200 < S < 1000$.

Table 2 compares results for implicit (penalty method) and explicit handling of the American constraint with constant timesteps. First, we note that for the penalty

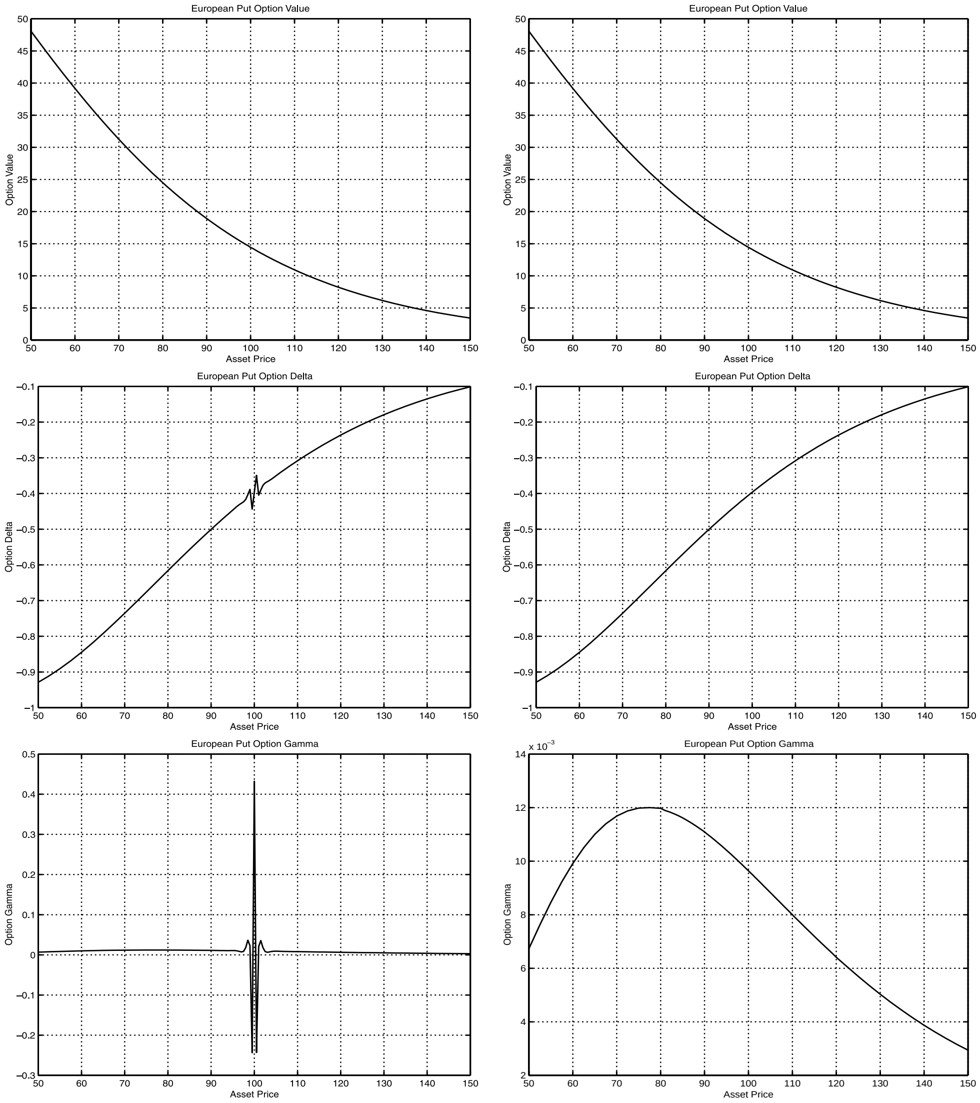


FIG. 1. *European put, $\sigma = .8$, $T = .25$, $r = .10$, $K = 100$. Crank–Nicolson timestepping, grid with* 135 *nodes. Left: no smoothing, right: Rannacher smoothing. Top: option value (V), middle: delta (V_S), bottom: gamma (V_{SS}).*

method the total number of nonlinear iterations is roughly the same across the two values of σ at each refinement level. This indicates that the volatility parameter has little effect on the number of iterations required. Now consider the results for the case where $\sigma = .2$. Taking into account the work per unit accuracy, the implicit method is slightly superior to the explicit technique. However, note that the implicit method does not appear to be converging quadratically. (The ratio of changes is about 3 instead of 4, which we would expect for quadratic convergence.) The explicit method appears to be converging at a first order rate (ratio of 2). Now consider the high volatility ($\sigma = .8$) results. Taking into account the total work, it would appear that in this case the explicit method is a little better than the implicit method. The latter

Table 2

Value of an American put option, $T = .25$, $r = .10$, $K = 100$, $S = 100$. Iters is the total number of nonlinear iterations. Change is the difference in the solution from the coarser grid. Ratio is the ratio of the changes on successive grids. Constant timesteps. Rannacher smoothing used. Work is measured in terms of number of multiply/divides.

Nodes	Timesteps	Iters	Value	Change	Ratio	Work (flops)
Explicit constraint, $\sigma = .2$						
55	25	25	3.04600			8.3×10^3
109	50	50	3.06049	.01449		3.3×10^4
217	100	100	3.06598	.00549	2.6	1.3×10^5
433	200	200	3.06822	.00224	2.5	5.2×10^5
865	400	400	3.06922	.00100	2.2	2.1×10^6
Implicit constraint, $\sigma = .2$						
55	25	33	3.05607			1.7×10^4
109	50	66	3.06555	.00948		6.7×10^4
217	100	134	3.06854	.00299	3.2	2.7×10^5
433	200	269	3.06953	.00099	3.0	1.1×10^6
865	400	543	3.06988	.00035	2.8	4.3×10^6
Explicit constraint, $\sigma = .8$						
68	25	25	14.61682			1.0×10^4
135	50	50	14.65685	.40020		4.1×10^4
269	100	100	14.67045	.01360	2.9	1.6×10^5
539	200	200	14.67542	.00497	2.7	6.5×10^5
1073	400	400	14.67738	.00196	2.5	2.6×10^6
Implicit constraint, $\sigma = .8$						
68	25	33	14.62708			2.1×10^4
135	50	68	14.66219	.03511		8.5×10^4
269	100	142	14.67324	.01105	3.2	3.5×10^5
537	200	299	14.67686	.00362	3.1	1.5×10^6
1073	400	627	14.67813	.00127	2.9	6.0×10^6

seems to have an error ratio of about 2.9, while the explicit method has a somewhat lower convergence rate.

As an additional accuracy check, for all runs we also monitored the quantity (4.20), which is a measure of the maximum relative error in enforcing the American constraint using the penalty method.

As well, we also monitored the size of the normalized residual of (4.10) for all unconstrained nodes (i.e., those nodes where $P_i = 0$). More precisely,

$$\begin{aligned}
\text{max linear solver error } &= \max_{n, i \in \kappa^n} \frac{|\mathrm{K}_i^n|}{|\mathrm{B}^n|}, \\
\mathrm{K}_i^{n+1} &= \left(\left[I + (1-\theta)\hat{M}^{n+1} \right] \bar{V}^{n+1} - \left[I - \theta \hat{M}^n \right] V^n \right)_i, \\
\mathrm{B}^{n+1} &= \max_{i \in \kappa^{n+1}} \left| \left(\left[I - \theta \hat{M}^n \right] V^n \right)_i \right|, \\
\kappa^{n+1} &= \{ i \mid \bar{P}_i^{n+1} = 0 \}.
\end{aligned} \tag{8.1}$$

In Table 3, we give the statistics for a single run, varying the *Large* parameter ((4.9)). Note that the value of $Large = 10^6$ results in a maximum relative error in enforcing the American constraint of $\simeq 10^{-9}$. Consequently, since this is well below the time and spatial discretization errors, we will use this value for all subsequent

Table 3

Test of varying the penalty parameter Large ((4.9)). American put option, $T = .25$, $r = .10$, $K = 100$, sigma $= .8$, value at $S = 100$. Nodes $= 269$, timesteps $= 100$. Iters is the total number of nonlinear iterations. Constant timesteps. Rannacher smoothing used. tol $= 1/Large$ ((5.2)). $***$ indicates iteration failed to converge, due to machine precision limitations.

Large	Total iterations	Value	Max linear solver error (8.1)	Max American error (4.20)
10^4	142	14.67323	6.0×10^{-14}	5×10^{-8}
10^6	142	14.67324	4.7×10^{-14}	5×10^{-10}
10^8	142	14.67324	4.1×10^{-14}	5×10^{-12}
10^{10}	142	14.67324	4.2×10^{-14}	5×10^{-14}
10^{12}	***	***	***	***

tests. It is interesting to see that the number of iterations is independent of the value of *Large*. This is a result of the finite termination property of the iteration. We can see from Table 3 that the upper limit to *Large* is determined by machine precision. Consequently, we can solve the discrete LCP problem to arbitrary accuracy (limited by machine precision) with a fixed number of nonlinear iterations.

9. Analysis of constant timestep examples. In terms of approximately solving the discrete LCP (4.11), the penalty method performs as the analysis predicts. The number of nonlinear iterations per timestep is typically of the order 1.4–1.6, independent of the volatility, for reasonable timesteps. The a posteriori check (4.20) (a maximum relative error of 10^{-9}, with $tol = 10^{-6}$ in terms of satisfaction of the discrete LCP constraint) indicates that the error introduced by the penalty method is quite small. This error is a function of $tol, Large$ (pseudocode (5.2)), and hence can be adjusted to the desired level. In these examples we have violated condition (4.15), which indicates that this condition is sufficient but not necessary for bounding the size of the penalty term independent of $\Delta t, \Delta S$. However, the results are disappointing in terms of the convergence of the discretization of the LCP. We do not observe quadratic convergence for the implicit handling of the American constraint.

An error ratio of about 2.8 would be consistent with global timestepping convergence at a rate of $O[(\Delta\tau)^{3/2}]$. Now, from [29, 32], we know that the value of an American call option (where the underlying asset pays a proportional dividend) behaves like $V = \text{const} . + O\left(\tau^{3/2}\right)$ near the exercise boundary and close to the expiration of the contract ($\tau \to 0$). This would give a value for $V_{\tau\tau\tau}$ in this region of

$$V_{\tau\tau\tau} = O[\tau^{-3/2}]. \tag{9.1}$$

It appears that the behavior of the American put near the exercise boundary and close to expiry is [23]

$$\begin{aligned} V &= \text{const} . + O[(\tau \log \tau)^{3/2}] \\ &\simeq \text{const} . + O[(\tau^{1-\epsilon})^{3/2})], \quad \epsilon > 0, \quad \epsilon \ll 1, \quad \tau \to 0. \end{aligned} \tag{9.2}$$

In the following we will ignore the ϵ in (9.2) and assume that the behavior of $V_{\tau\tau\tau}$ is given by (9.1).

From (9.1), the local time truncation error for Crank–Nicolson timestepping is (near the exercise boundary)

$$\text{local error } = O\left[\frac{(\Delta\tau)^3}{\tau^{3/2}}\right]. \tag{9.3}$$

Assuming that the global error is of the order of the sum of the local errors, from (9.3) we obtain

$$\text{global error } = O\left[\sum_{i=1}^{i=1/\Delta\tau}\frac{(\Delta\tau)^3}{(i\Delta\tau)^{3/2}}\right] \simeq O[(\Delta\tau)^{3/2}], \tag{9.4}$$

which is consistent with the observed rate of convergence. Now, instead of taking constant timesteps, suppose we take timesteps which satisfy

$$\max_i(|V_i^{n+1} - V_i^n|) \simeq d, \tag{9.5}$$

where d is a specified constant. In order to take the limit to convergence, at each grid refinement we will halve both the grid spacing and d. It is reasonable to assume that the maximum change over a timestep (at least near $\tau = 0$) will occur near K. Therefore, from (7.1),

$$\Delta V^{n+1} = \max_i(|V_i^{n+1} - V_i^n|) \simeq O\left[\frac{\Delta\tau^{n+1}}{\sqrt{\tau^n}}\right]. \tag{9.6}$$

Therefore, from (9.5) and (9.6), we have

$$\Delta\tau^{n+1} = O[d\sqrt{\tau^n}]. \tag{9.7}$$

Assuming a local error of the form (9.3), and using (9.3) and (9.7), this gives a local error with the variable timesteps satisfying (9.5) as

$$\text{local error } = O\left[\frac{\left((\Delta\tau)^{n+1}\right)^3}{(\tau^n)^{3/2}}\right] = O\left[\frac{d^3(\tau^n)^{3/2}}{(\tau^n)^{3/2}}\right] = O(d^3). \tag{9.8}$$

This implies a global error (with $O(1/d)$ timesteps) of

$$\text{global error } = O(d^2). \tag{9.9}$$

Therefore, suppose that we take variable timesteps consistent with (9.5). Then at each refinement stage, where we double the number of grid nodes, and double the number of timesteps (by halving d), we should see quadratic convergence. Note that we should reduce the initial timestep $\Delta\tau^0$ by four at each refinement. We make no claim that the above analysis of the time truncation error is in any way precise but only suggestive of an appropriate timestepping strategy.

10. A timestep selector.

The timestep selector used is based on a modified form of that suggested in [18]. Given an initial timestep $\Delta\tau^{n+1}$, then a new timestep is selected so that

$$\Delta\tau^{n+2} = \left(\min_i\left[\frac{\texttt{dnorm}}{\frac{|V(S_i,\tau^n+\Delta\tau^{n+1})-V(S_i,\tau^n)|}{\max(D,|V(S_i,\tau^n+\Delta\tau^{n+1})|,|V(S_i,\tau^n)|)}}\right]\right)\Delta\tau^{n+1}, \tag{10.1}$$

where dnorm is a target relative change (during the timestep) specified by the user. The scale D is selected so that the timestep selector does not take an excessive number of timesteps in regions where the value is small. (For options valued in dollars, $D = 1$ is typically appropriate.) In (10.1), we have normalized the factor used to estimate the new timestep. This is simply to avoid slow timestep growth for large values of the contract. This could be a problem with call options, for example, where the computational domain is truncated at a large value of S. If we did not examine the relative changes over a timestep, then it is possible that the timestep would be limited by large absolute changes in the solution (which would occur as $S \to \infty$), even though the relative changes were small.

Since $V(S_i = K, \tau \simeq 0) \simeq 0$, we expect that the denominator of (10.1) will take its largest value near $S = K$, since V increases rapidly there. Consequently, the timestep selector (10.1) will approximately enforce the condition that

$$\Delta V^{n+1} \simeq D \times \mathtt{dnorm}. \tag{10.2}$$

Hence we will have

$$\Delta\tau^{n+1} = O(\mathtt{dnorm}\sqrt{\tau^n}), \tag{10.3}$$

so that we should see a global error of $O[(\mathtt{dnorm})^2]$, which follows from (9.9).

Note that timestep selector (10.1) estimates the change in the solution at the new timestep based on changes observed over the old timestep. Some adjustments can be made to this simple model if a more precise form for the time evolution of the solution is assumed, but we prefer (10.1) since it is simple and conservative.

In practice, we select a $(\Delta\tau)^0$ for the coarsest grid, and then $(\Delta\tau)^0$ is cut by four at each grid refinement. There is not much of a penalty for underestimating a suitable $(\Delta\tau)^0$ since the timestep will increase rapidly if the estimate is too conservative. In the following runs, we used values of $(\Delta\tau)^0 = 10^{-3}$ and $\mathtt{dnorm} = .2$ on the coarsest grid. The value of dnorm was reduced by two at each grid refinement.

11. Variable timestep examples.

Table 4 presents results for the cases considered in Table 2 but this time using the timestep selector (10.1). In this case, the implicit method appears to be a clear winner in terms of flops per unit accuracy. Use of variable timesteps actually seems to degrade the convergence of the explicit method. This can be explained by looking at the timestep history. The timestep selector uses small timesteps at the start and then takes large steps at the end. Note that the average timestep size (total time divided by number of timesteps) is larger for the variable timestep run compared to the constant timestep run (Table 2). This clearly negatively impacts the explicit method, which seems to show a first order rate of convergence. On the other hand, the implicit method appears to exhibit close to quadratic convergence.

Figure 2 shows value, delta, and gamma for the $\sigma = .2$ case, using both explicit and implicit treatments of the American constraint. Although the value and delta appear similar for both cases, there are clearly large oscillations in the gamma near the early exercise boundary for the explicit method. The implicit method does show some small oscillations near the exercise boundary. However, this is due to the use of Crank–Nicolson timestepping, as noted in [6]. These oscillations disappear if fully implicit timestepping is used, as shown in Figure 3.

12. Comparison with binomial lattice methods.

It is interesting to compare the results here with those obtained using the binomial lattice method, which

Table 4

Value of an American put option, $T = .25$, $r = .10$, $K = 100$, $S = 100$. Iters is the total number (over all time steps) of nonlinear iterations. Change is the difference in the solution from the coarser grid. Ratio is the ratio of the changes on successive grids. Variable timesteps. Rannacher smoothing used. Work is measured in terms of number of multiply/divides.

Nodes	Timesteps	Iters	Value	Change	Ratio	Work (flops)
Explicit constraint, $\sigma = .2$						
55	18	18	3.04499			9.9×10^3
109	33	33	3.05825	.01326		3.6×10^4
217	63	63	3.06425	.00600	2.2	1.4×10^5
433	122	122	3.06717	.00292	2.1	5.3×10^5
865	239	239	3.06863	.00146	2.0	2.1×10^6
Implicit constraint, $\sigma = .2$						
55	18	27	3.06403			1.5×10^4
109	33	51	3.06867	.00464		5.7×10^4
217	63	98	3.06975	.00108	4.3	2.2×10^5
433	122	194	3.07002	.00027	4.0	8.5×10^5
865	239	385	3.07008	.00006	4.5	3.2×10^6
Explicit constraint, $\sigma = .8$						
68	31	31	14.64828			2.1×10^4
135	66	66	14.66856	.02022		8.9×10^4
269	136	136	14.67472	.00616	3.3	3.7×10^5
537	276	276	14.67703	.00231	2.7	1.5×10^6
1073	554	554	14.67800	.00097	2.4	5.9×10^6
Implicit constraint, $\sigma = .8$						
68	31	45	14.65863			3.2×10^4
135	66	98	14.67417	.01554		1.4×10^5
269	136	208	14.67778	.00361	4.3	5.7×10^5
537	276	430	14.67862	.00084	4.3	2.4×10^6
1073	554	872	14.67882	.00020	4.2	9.5×10^6

is commonly used in finance [33]. In Appendix C, we show that this technique is simply an explicit finite difference method on a log-transformed grid. Consequently, the truncation error is $O(\Delta\tau)$, where the total number of steps is $N = O[1/(\Delta\tau)]$.

The binomial lattice method requires about $3/2N^2$ flops (counting only multiplies, and assuming all necessary factors are precomputed). Note that we obtain the value of the option at $t = 0$ only at the single point S_0^0, in contrast to the PDE methods which obtain values for all $S \in [0, S_{\max}]$. As a result, the methods are not directly comparable. Nevertheless, assuming that we are interested only in obtaining the solution at a single point, it is interesting and useful to compare these two techniques.

Given $N = O[1/(\Delta\tau)]$, the complexity of the binomial method is $O(N^2)$. Since the error in the lattice method is $O(\Delta\tau) = O(1/N)$, we have

$$\text{error binomial lattice } = O\left[(\text{complexity})^{-1/2}\right]. \tag{12.1}$$

Suppose instead that we use an implicit finite volume method with Crank–Nicolson timestepping and that the penalty method is employed for handling the American constraint. The complexity of this approach is $O(N^2)$, where we have assumed that $N = O[1/(\Delta S)]$. (Note that this is the case if we use the timestep selector (10.1) and `dnorm` $= O(\Delta S)$.) It is also assumed that the the number of nonlinear iterations

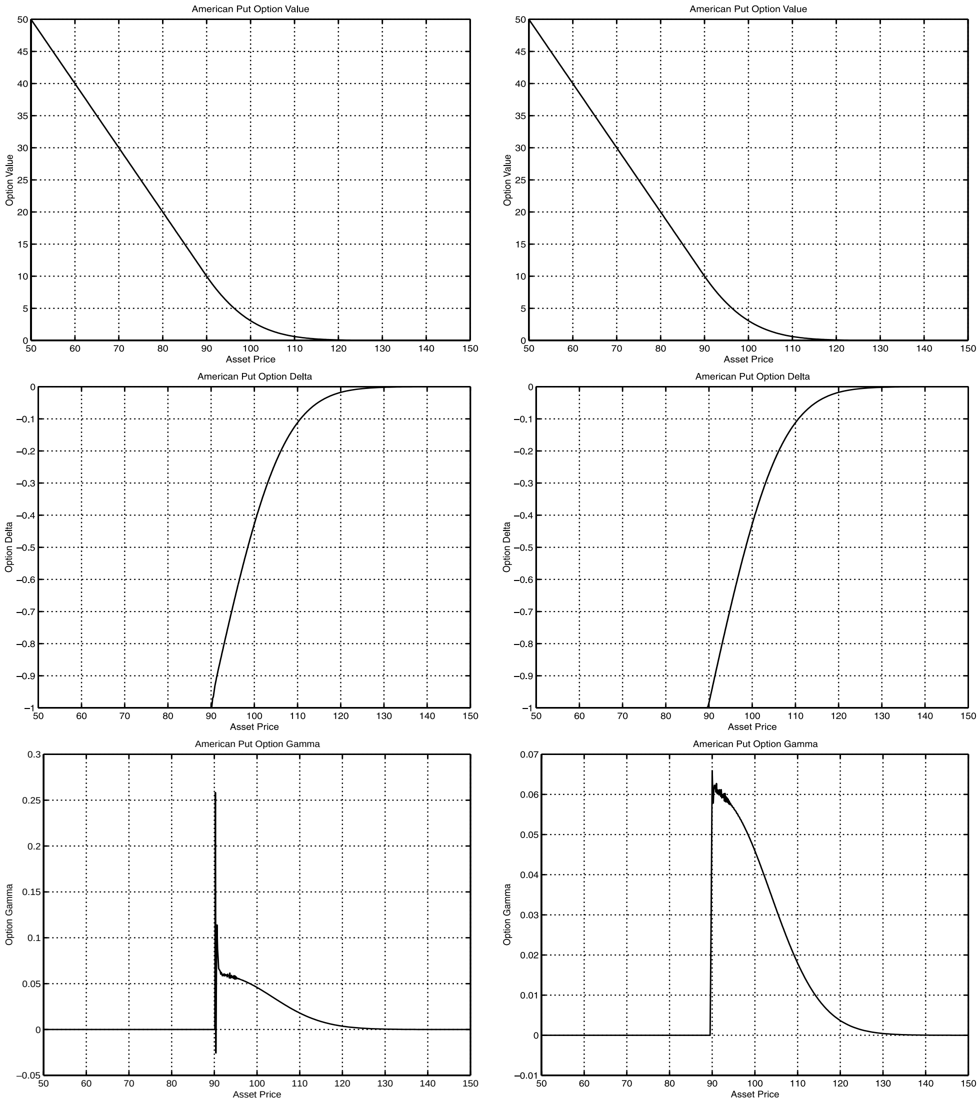


FIG. 2. *American put, $\sigma = .2$, $T = .25$, $r = .10$, $K = 100$. Crank–Nicolson timestepping, Rannacher smoothing, variable timesteps, grid with 433 nodes. Left: explicit constraint, right: implicit constraint. Top: option value (V), middle: delta (V_S), bottom: gamma (V_{SS}).*

per timestep is constant as $\Delta S \to 0$, which is observed as long as `dnorm` $= O(\Delta S)$. When timesteps are selected using (10.1), we have observed quadratic convergence. This implies

$$\text{error implicit finite volume } = O\left(N^{-2}\right) = O\left[(\text{complexity})^{-1}\right]. \tag{12.2}$$

Therefore, the implicit finite volume method is asymptotically superior to the binomial lattice method, even if the solution is desired at only one point.

It is interesting to determine at what levels of accuracy we can expect the implicit PDE method to become more efficient than the binomial method. Table 5 gives the

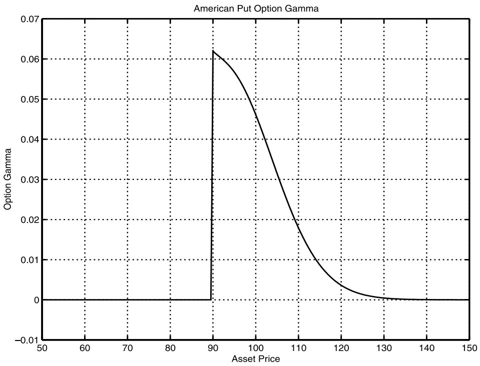


FIG. 3. *Gamma (V_{SS}) of an American put, $\sigma = .2$, $T = .25$, $r = .10$, $K = 100$. Fully implicit timestepping; Rannacher smoothing; variable timesteps. Grid with 433 nodes used. Constraint imposed implicitly.*

results for a binomial lattice solution (algorithm (C.2)) for the problems solved earlier using an implicit PDE approach. This table should be compared to Table 4.

For further points of comparison, we also computed solutions to the problem used in [13]. We used two versions of the problem in [13], one with an expiry time of $T = 1$ and the other with $T = 5$. Figure 4 summarizes the convergence of both the binomial lattice and PDE methods for all four problems. The absolute error is computed by taking the *exact* solution as obtained by extrapolating the PDE solution down to zero grid and timestep size, assuming quadratic behavior. The PDE method becomes more efficient than the binomial lattice method at tolerances between .01–.003 depending on the problem parameters. These crossover points occur at tolerances which would be used in practice. Note that in these comparisons, we are putting the best possible

TABLE 5

Binomial lattice method. Value of an American put, $T = .25$, $r = .10$, $K = 100$, $S = 100$. Change is the difference in the solution from the coarser grid. Ratio is the ratio of the changes on successive grids. Work is measured as the number of multiplies.

Timesteps	Value	Change	Ratio	Work (flops)
$\sigma = .2$				
50	3.06186			3.8×10^3
100	3.06611	0.00425		1.5×10^4
200	3.06810	0.00199	2.1	6.0×10^4
400	3.06913	0.00103	1.9	2.4×10^5
800	3.06962	0.00049	2.1	9.6×10^5
1600	3.06987	0.00025	2.0	3.8×10^6
3200	3.06999	0.00012	2.1	1.5×10^7
$\sigma = .8$				
50	14.62649			3.8×10^3
100	14.65269	0.02620		1.5×10^4
200	14.66582	0.01313	2.0	6.0×10^4
400	14.67238	0.00656	2.0	2.4×10^5
800	14.67563	0.00325	2.0	9.6×10^5
1600	14.67726	0.00163	2.0	3.8×10^6
3200	14.67807	0.00081	2.0	1.5×10^7

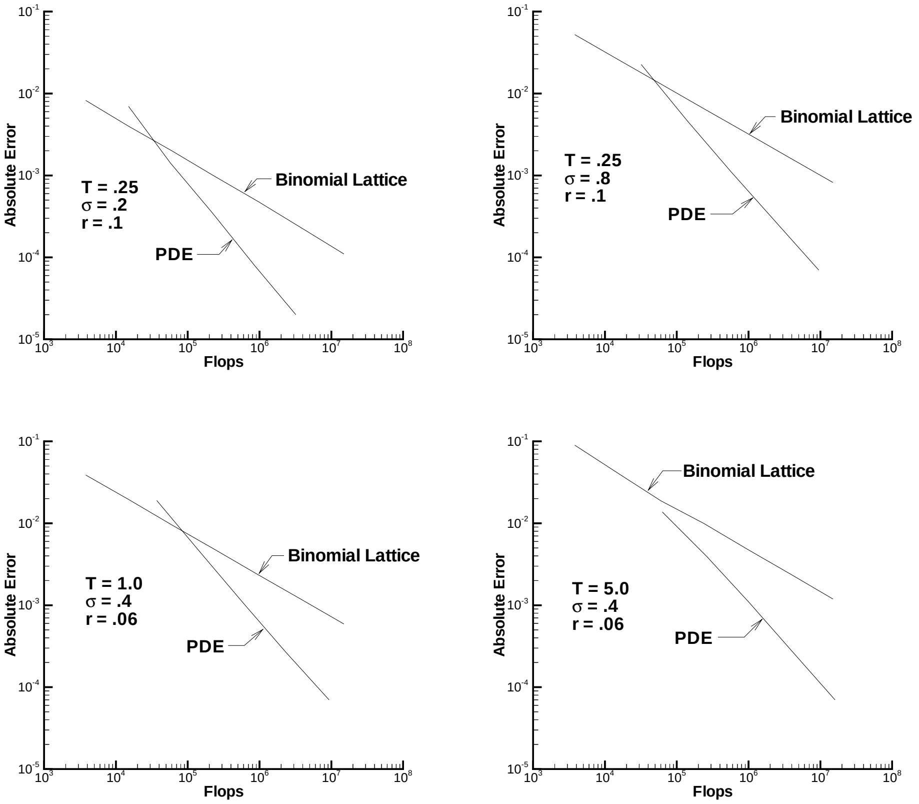


FIG. 4. *American put, $K = 100$. Absolute error as a function of number of floating point operations (flops), measured as the number of multiply/divides for all the test problems, at $S = 100$, $t = 0$.*

light on the binomial lattice method, since we ignore the fact that we obtain much more information with the implicit PDE technique.

13. Application of penalty methods to more general problems. As derived in the appendices, a sufficient condition for monotone convergence of the penalty iteration is that the discretized differential operator is an M-matrix. In practice, we have found that this condition is not necessary for rapid convergence of the penalty iteration. For example, in [40, 39], we have applied the penalty method to American options with stochastic volatility, convertible bonds (which have American type maximum and minimum constraints), and American options on two assets, with good results. In this case, the discretized differential operator was not an M-matrix, and, if a flux limiter was used, the discretized differential operator was nonlinear. We also routinely violated the timestep condition (4.15). As long as the Rannacher smoothing technique was used, the solution was sufficiently smooth enough that no ill effects were observed with the penalty iteration.

Note that the discrete M-matrix condition was not required in the proof of convergence of the penalty method for an elliptic obstacle problem with the Laplacian as the differential operator [24]. However, timestep restrictions were required in the proofs of convergence of the penalty method for parabolic problems in [30]. In view of our computational experience, it appears to us that these conditions are artificial. We conjecture that the penalty iteration converges rapidly under much weaker conditions than those outlined in the appendices.

14. Conclusion.

We have derived sufficient conditions so that the solution of the discrete penalized equations solves an approximate version of the discrete LCP formulation of the American option pricing problem. The error in the approximation can be made arbitrarily small by increasing the penalty factor. We have also given sufficient conditions so that a Newton iteration method for solving the discrete nonlinear penalized equations converges monotonically to the unique solution of the nonlinear algebraic equations. This explains the observed rapid convergence of this technique.

If constant timesteps are used, the computed solution appears to converge at less than a second order rate in the limit as the grid spacing and timestep are reduced to zero. An heuristic analysis of the behavior of the solution near the exercise boundary indicates that convergence (with constant timesteps) occurs only at the rate $(\Delta t)^{3/2}$. However, a timestep selection method was suggested which, based on our analysis, should be expected to restore quadratic convergence. Numerical experiments confirmed that this convergence rate was indeed obtained using this timestep selector.

In general, the use of an implicit penalty method combined with the timestep selector can be recommended. As well as being more efficient in terms of the number of flops per unit accuracy, the solution obtained using an implicit method is qualitatively superior to the solution obtained using an explicit method for handling the American constraint. The explicit solution exhibited large oscillations in gamma near the exercise boundary.

The implicit PDE method is asymptotically superior to the standard (in finance) binomial lattice method, which has only linear convergence. However, if low accuracy solutions are required at only a single point, then a binomial method can be more efficient than the PDE approach. For typical parameters, the crossover point where the PDE method is to be preferred occurs at an absolute error tolerance of between $.01-.003$. However, if information at more than a single point is desired, then the PDE method is always preferable. As well, the binomial lattice method is highly optimized for simple cases. For example, the addition of discretely observed barriers [3, 42] causes difficulties for binomial methods. However, this case presents no particular difficulty for a PDE finite volume method.

The penalty method described here has been applied to multidimensional problems as shown in [40, 39]. This method has the advantage that standard sparse matrix software can be used to solve the Jacobian matrix. This is especially important for multifactor problems. In fact, the penalty method in [40] was applied to problems which did not satisfy the sufficient conditions derived in this work, with no apparent ill effects. It is a topic of further research to extend the convergence results in this paper to the more general problems described in [40, 39].

Appendix A. Error in the penalty formulation.

In this appendix, we determine sufficient conditions which allow us to bound (4.14). Suppose that node k is the node where the penalty term $P_k^{n+1}(V_k^* - V_k^{n+1})$ attains its maximum. Consider the term

$$\begin{aligned}[\hat{M}(V^* - V^{n+1})]_k =& \Delta\tau \sum_{j \in \eta_k} \left(\bar{\gamma}_{kj} + \bar{\beta}_{kj}\right) [(V_k^* - V_k^{n+1}) - (V_j^* - V_j^{n+1})] \\ & + r\Delta\tau[V_k^* - V_k^{n+1}]. \end{aligned} \tag{A.1}$$

Since the penalty term attains its maximum value at node k, we have

$$\begin{aligned}&[(V_k^* - V_k^{n+1}) - (V_j^* - V_j^{n+1})] \geq 0 \\ &V_k^* - V_k^{n+1} \geq 0. \end{aligned} \tag{A.2}$$

Since $\bar{\gamma}_{kj} + \bar{\beta}_{kj} \geq 0$, it follows from (A.1)–(A.2) that $[\hat{M}(V^* - V^{n+1})]_k \geq 0$ at node k. Alternatively,

$$[\hat{M}V^{n+1}]_k \leq [\hat{M}V^*]_k, \tag{A.3}$$

implying

$$[I + (1-\theta)\hat{M}V^{n+1}]_k \leq [I + (1-\theta)\hat{M}V^*]_k. \tag{A.4}$$

Writing (4.10) at node k, we have

$$\left(\left[I + (1-\theta)\hat{M}\right] V^{n+1}\right)_k = \left(\left[I - \theta\hat{M}\right] V^n\right)_k + \left(\left[\bar{P}(V^{n+1})\right] (V^* - V^{n+1})\right)_k . \tag{A.5}$$

Noting that $P_k^{n+1}(V_k^* - V_k^{n+1}) \geq 0$, it follows from (A.5) and (A.3) that

$$\begin{aligned}\left|[P_k^{n+1}(V_k^* - V_k^{n+1})]\right| &= \left\|P^{n+1}(V^* - V^{n+1})\right\|_\infty \\ &\leq \left|\left(\left[I + (1-\theta)\hat{M}\right] V^{n+1}\right)_k\right| + \left|\left(\left[I - \theta\hat{M}\right] V^n\right)_k\right| \\ &\leq \left|\left(\left[I + (1-\theta)\hat{M}\right] V^*\right)_k\right| + \left|\left(\left[I - \theta\hat{M}\right] V^n\right)_k\right| \\ &\leq \left\|\left[I + (1-\theta)\hat{M}\right] V^*\right\|_\infty + \left\|\left[I - \theta\hat{M}\right] V^n\right\|_\infty \\ &\leq \|V^*\|_\infty + (1-\theta)\left\|\hat{M}V^*\right\|_\infty + \left\|\left[I - \theta\hat{M}\right] V^n\right\|_\infty . \end{aligned} \tag{A.6}$$

We now proceed to bound each of the terms on the right-hand side of (A.6). Given a put payoff of the form

$$V^* = V^0 = \max(K - S, 0), \tag{A.7}$$

where K is the strike, we have $\|V^*\|_\infty = K$. In bounding $||\hat{M}V^*\|_\infty$, we note that the worst case occurs at $S_i = K$ so that

$$\begin{aligned}||\hat{M}V^*\|_\infty &\leq \text{const}\,.\ |\hat{M}V^*|_i, \qquad S_i = K \\ &= O\left(\frac{\Delta\tau}{\Delta S}\right), \end{aligned} \tag{A.8}$$

where $\Delta S = \min_i(S_i - S_{i-1})$. We assume that the timestep and mesh size are reduced to zero in such a way that

$$\frac{\Delta\tau}{\Delta S} = \text{const}\,., \tag{A.9}$$

where this constant is independent of $\Delta\tau, \Delta S$. (It does not make any sense to drive the S discretization to zero if the timestep truncation error is also not reduced as well.) Consequently, we can assume that $\|\hat{M}V^*\|_\infty$ is bounded independent of *Large* and $\Delta\tau, \Delta S$.

If we also assume that the timestep is selected so that

$$1 - \theta \left(\Delta\tau \sum_{j \in \eta_i} \left(\bar{\gamma}_{ij} + \bar{\beta}_{ij} \right) + r\Delta\tau \right) \geq 0, \tag{A.10}$$

then we have (recalling that $\hat{M}$ is an M-matrix with row sum $r\Delta\tau$)

$$\left\| \left[I - \theta \hat{M} \right] V^n \right\|_\infty \leq (1 - r\Delta\tau) \left\| V^n \right\|_\infty \leq \left\| V^n \right\|_\infty . \tag{A.11}$$

Assuming condition (A.10) is satisfied, it follows from (4.10) and (A.7) that

$$\left\| V^n \right\|_\infty \leq \max(\left\| V^{n-1} \right\|_\infty, \left\| V^* \right\|_\infty) = \left\| V^* \right\|_\infty = K. \tag{A.12}$$

Note that if a fully implicit discretization is used ($\theta = 0$), then condition (A.10) is trivially satisfied. For Crank–Nicolson timestepping, condition (A.10) implies that $\Delta\tau/(\Delta S)^2 \leq \text{const}.$ as $\Delta S, \Delta\tau \to 0$.

Consequently, we have shown that

$$\left\| P^{n+1}(V^* - V^{n+1}) \right\|_\infty \leq 2K + O\left(\frac{\Delta\tau}{\Delta S} \right). \tag{A.13}$$

In other words, at any node where $V_i^{n+1} < V_i^*$, we have $|Large(V_i^* - V_i^{n+1})| \leq C$, where C is independent of *Large*. Therefore, by choosing *Large* sufficiently large, the error in the solution of the LCP can be made arbitrarily small, and Theorem 4.1 follows.

Appendix B. Monotone convergence. We will first prove that iteration (6.1) has a *monotone* property. Writing (6.1) for iteration k gives

$$\left[I + (1-\theta)\hat{M} + \bar{P}^{k-1} \right] \bar{V}^k = \left[I - \theta \hat{M} \right] V^n + \bar{P}^{k-1} V^*. \tag{B.1}$$

First, note that (B.1) always has a solution, since $I + (1-\theta)\hat{M} + \bar{P}^{k-1}$ is a diagonally dominant M matrix and is consequently nonsingular.

This can be written as

$$\left[I + (1-\theta)\hat{M} + \bar{P}^{k} \right] \bar{V}^k + \left[\bar{P}^{k-1} - \bar{P}^k \right] \bar{V}^k = \left[I - \theta \hat{M} \right] V^n + \bar{P}^{k-1} V^*. \tag{B.2}$$

Subtracting (B.2) from (6.1) gives

$$\left[I + (1-\theta)\hat{M} + \bar{P}^{k} \right] (\bar{V}^{k+1} - \bar{V}^k) = \left[\bar{P}^k - \bar{P}^{k-1} \right] (V^* - \bar{V}^k), \qquad k \geq 1. \tag{B.3}$$

Now examine each of the components of the right-hand side of (B.3). There are two possible cases:

$$\begin{aligned}
\text{Case 1:} \qquad \bar{V}_i^k < V_i^* &\Rightarrow \bar{P}_{ii}^k = Large \\
&\Rightarrow (Large - \bar{P}_{ii}^{k-1})(V^* - \bar{V}^k)_i \geq 0 \\
&\Rightarrow \left[\bar{P}^k - \bar{P}^{k-1} \right]_i (V^* - \bar{V}^k)_i \geq 0, \\
\text{Case 2:} \qquad \bar{V}_i^k \geq V_i^* &\Rightarrow \bar{P}_{ii}^k = 0 \\
&\Rightarrow (-\bar{P}_{ii}^{k-1})(V^* - \bar{V}^k)_i \geq 0 \\
&\Rightarrow \left[\bar{P}^k - \bar{P}^{k-1} \right]_i (V^* - \bar{V}^k)_i \geq 0.
\end{aligned}$$

Thus we always have

$$\left[\bar{P}^k - \bar{P}^{k-1}\right](V^* - \bar{V}^k) \geq 0, \qquad k \geq 1. \tag{B.4}$$

Since $[I+(1-\theta)\hat{M}+\bar{P}^k]$ is an M-matrix, it follows from (B.3)–(B.4) that $(\bar{V}^{k+1}-\bar{V}^k) \geq 0$ for $k \geq 1$, or, in component form, $(\bar{V}^{k+1} - \bar{V}^k)_i \geq 0 \ \forall i$ for $k \geq 1$.

We now demonstrate that the iteration (6.1) has finite termination. Let the set of all nodes in the discretization be denoted by ν. Given any iterate $\bar{V}^k$, we can define

$$\begin{aligned} \chi^k &= \{i \mid (\bar{P}^k)_i = 0\}, \\ \nu - \chi^k &= \{i \mid (\bar{P}^k)_i > 0\}. \end{aligned} \tag{B.5}$$

Since for $k \geq 1$ we have that the iterates increase monotonically, any node in χ^k remains in $\chi^j \ \forall j > k \geq 1$. If at any stage $\nu - \chi^k = \nu - \chi^{k-1}$, then $(\bar{P}^k - \bar{P}^{k-1}) = 0$, and the iteration terminates with a zero update ((B.3)). If a node in $\nu - \chi^k$ becomes unconstrained, this node moves into the set χ^{k+1}. This node will always remain in $\chi^j, j > k \geq 1$. The number of nodes in the set $\nu - \chi^{k+1}$ is then at least one less than the number of nodes in $\nu - \chi^k$. Eventually, either $\nu - \chi^{k+1} = \nu - \chi^k$ or $\nu - \chi^{k+1}$ is exhausted, and the iteration terminates. Hence the iteration always converges, and a solution exists.

The above argument assumes exact arithmetic. In practice, we apply the extra termination condition based on the update in algorithm (5.2) as a precaution against errors in floating point arithmetic. Of course, we will also need the update termination test if the problem has nonlinearities other than the simple American constraint.

We now demonstrate that the solution obtained by the penalty iteration is unique. Suppose there are two solutions $\bar{V}_1$ and $\bar{V}_2$ to the penalized problem. Let $\bar{P}^1 \equiv P(\bar{V}_1)$ and $\bar{P}^2 \equiv P(\bar{V}_2)$. Then

$$\left[I + (1-\theta)\hat{M} + \bar{P}^1\right]\bar{V}_1 = \left[I - \theta\hat{M}\right]V^n + \bar{P}^1 V^*, \tag{B.6}$$

$$\left[I + (1-\theta)\hat{M} + \bar{P}^2\right]\bar{V}_2 = \left[I - \theta\hat{M}\right]V^n + \bar{P}^2 V^*. \tag{B.7}$$

We can write (B.6) as

$$\left[I + (1-\theta)\hat{M} + \bar{P}^2\right]\bar{V}_1 + \left[\bar{P}^1 - \bar{P}^2\right]\bar{V}_1 = \left[I - \theta\hat{M}\right]V^n + \bar{P}^1 V^*. \tag{B.8}$$

Subtracting (B.7) from (B.8) gives

$$\left[I + (1-\theta)\hat{M} + \bar{P}^2\right](\bar{V}_1 - \bar{V}_2) = \left[\bar{P}^1 - \bar{P}^2\right](V^* - \bar{V}_1). \tag{B.9}$$

Using a similar argument as we used in proving monotone iteration, we have that

$$\left[\bar{P}^1 - \bar{P}^2\right](V^* - \bar{V}_1) \geq 0. \tag{B.10}$$

Since $[I + (1-\theta)\hat{M} + \bar{P}^2]$ is an M-matrix, it follows from equations (B.9–B.10) that $(\bar{V}_1 - \bar{V}_2) \geq 0$. Interchanging subscripts, we have $(\bar{V}_2 - \bar{V}_1) \geq 0$, and hence $\bar{V}_2 = \bar{V}_1$. Consequently, Theorem 6.1 follows.

Appendix C. The binomial lattice method. Let $S_m^n = u^{2m-n} S_0^0$ for $m = 0, \ldots, n$ denote the value of the asset price at time $t_n = n\Delta t$ and lattice point m, where $u = e^{\sigma\sqrt{\Delta t}}$ and $\Delta t = T/N$. Note that T is the expiry time of the option and N is the number of timesteps. Also note that we are considering time t going forward in this case, in contrast to $\tau = T - t$ (time going backwards) as in the previous sections. This results in a solution algorithm which proceeds backwards from $t = T$ to $t = 0$ (i.e., from $t = t_N$ to $t = 0$).

Let V_m^n be the value of the option associated with asset price S_m^n, at time $t = n\Delta t$. Of course, we have $V_m^N = \max(K - S_m^N, 0)$ for $m = 0, \dots, N$. Define

$$p = \frac{e^{r\Delta t} - e^{-\sigma\sqrt{\Delta t}}}{e^{\sigma\sqrt{\Delta t}} - e^{-\sigma\sqrt{\Delta t}}}. \tag{C.1}$$

Then the value of the American put option V_0^0 (at the single point $S = S_0^0$) is obtained from the following algorithm.

Binomial Lattice Algorithm.

```
For $n = N - 1, \dots, 0$
    For $m = 0, \dots, n$
        (C.2)  $\bar{V}_m^n = e^{-r\Delta t}\left(pV_{m+1}^{n+1} + (1-p)V_m^{n+1}\right),$
               $V_m^n = \max(K - S_m^n, \bar{V}_m^n)$
    EndFor
EndFor.
```

The above method is usually derived in the financial literature based on probabilistic arguments. In fact, we can see that this is equivalent to an explicit finite difference method with a particular choice for the timestep. Consider the following Black–Scholes equation for a European option:

$$V_t + \frac{\sigma^2}{2}S^2 V_{SS} + rSV_S - rV = 0. \tag{C.3}$$

Define a new variable $X = \log S$ so that (C.3) becomes

$$V_t + \frac{\sigma^2}{2}V_{XX} + \left(r - \frac{\sigma^2}{2}\right)V_X - rV = 0. \tag{C.4}$$

Letting $V = e^{rt}W$, (C.4) becomes

$$W_t + \frac{\sigma^2}{2}W_{XX} + \left(r - \frac{\sigma^2}{2}\right)W_X = 0. \tag{C.5}$$

Now let $W_m^n = W(\log S_0^0 + (2m-n)\sigma\sqrt{\Delta\tau}, n\Delta\tau)$ for $m = 0, \dots, n$. Discretizing (C.5) using central differencing in the X direction and an explicit timestepping method, we obtain

$$W_m^n = \left[p^*\left(W_{m+1}^{n+1}\right) + (1-p^*)\left(W_m^{n+1}\right)\right] + O[(\Delta t)^2], \tag{C.6}$$

where $p^* = 1/2[1 + \sqrt{\Delta t}(r/\sigma - \sigma/2)]$. Writing (C.6) in terms of V_m^n gives

$$V_m^n = e^{-r\Delta t}\left[p^*\left(V_{m+1}^{n+1}\right) + (1-p^*)\left(V_m^{n+1}\right)\right] + O[(\Delta t)^2]. \tag{C.7}$$

Expanding p in (C.1) in a Taylor series, noting the definition of p^*, and assuming that $V_{m+1}^{n+1} - V_m^{n+1} = O(\sqrt{\Delta\tau})$, we obtain

$$V_m^n = e^{-r\Delta t}\left[p\left(V_{m+1}^{n+1}\right) + (1-p)\left(V_m^{n+1}\right)\right] + O[(\Delta t)^2]. \tag{C.8}$$

Comparing (C.8) with algorithm (C.2), we can see that the binomial lattice method is simply an explicit finite difference discretization of the discrete LCP (2.2), with the American constraint applied explicitly.

REFERENCES

[1] M. Avellaneda, A. Levy, and A. Parás, *Pricing and hedging derivative securities in markets with uncertain volatilities*, Appl. Math. Fin., 2 (1995), pp. 73–88.

[2] J. Becker, *A second order backward difference method with variable timesteps for a parabolic problem*, BIT, 38 (1998), pp. 644–662.

[3] P. Boyle and S. Lau, *Bumping up against the barrier*, J. Derivatives, 1 (1994), pp. 6–14.

[4] F. Clarke, *Optimization and Nonsmooth Analysis*, Wiley, New York, 1983.

[5] N. Clarke and K. Parrot, *The multigrid solution of two factor American put options*, Appl. Math. Fin., 6 (1999), pp. 177–195.

[6] T. Coleman, Y. Li, and A. Verma, *A Newton Method for American Option Pricing*, Technical report CTC9907, Cornell Theory Center, Cornell University, Ithaca, NY, 1999.

[7] R. Cottle, J.-S. Pang, and R. Stone, *The Linear Complementarity Problem*, Academic Press, Boston, 1992.

[8] K. Deckelnick and K. Siebert, *$W^{1,\infty}$ convergence of the discrete free boundary for obstacle problems*, IMA J. Numer. Anal., 20 (2000), pp. 481–489.

[9] M. Dempster and J. Hutton, *Fast numerical valuation of American, exotic and complex options*, Appl. Math. Fin., 4 (1997), pp. 1–20.

[10] C.M. Elliott and J. Ockendon, *Weak and Variational Methods for Moving Boundary Problems*, Pitman, Boston, 1982.

[11] A. Fisher and C. Kanzow, *On the finite termination of an iterative method for linear complmetarity problems*, Math. Programming, 74 (1996), pp. 279–292.

[12] A. Friedman, *Variational Principles and Free-Boundary Problems*, Robert E. Krieger, Malabar, FL, 1988.

[13] S. Heston and G. Zhou, *On the rate of convergence of discrete-time contingent claims*, Math. Finance, 10 (2000), pp. 53–75.

[14] J. Huang and J.-S. Pang, *Option pricing and linear complimentarity*, J. Comp. Fin., 2 (1998), pp. 31–60.

[15] J. Huang and J.-S. Pang, *A mathematical programming with equilibrium constraints approach to the implied volatility surface of American options*, J. Comp. Fin., 4 (2000), pp. 21–56.

[16] J. Hull, *Options, Futures and Other Derivatives*, 4th ed., Prentice-Hall, Englewood Cliffs, NJ, 2000.

[17] H. Jiang and L. Qi, *A new nonsmooth equations approach to nonlinear complementarity problems*, SIAM J. Control Optim., 35 (1997), pp. 178–193.

[18] C. Johnson, *Numerical Solution of Partial Differential Equations by the Finite Element Method*, Cambridge University Press, Cambridge, UK, 1987.

[19] C. Kanzow, *Some noninterior continuation methods for linear complementarity problems*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 851–868.

[20] C. Kanzow and H. Pieper, *Jacobian smoothing methods for nonlinear complementarity problems*, SIAM J. Optim., 9 (1999), pp. 342–373.

[21] M. Kocvara and J. Zowe, *An iterative two step algorithm for linear complementarity problems*, Numer. Math., 68 (1994), pp. 95–106.

[22] M. Kojima, N. Megiddo, T. Noma, and A. Yoshise, *A Unified Approach to Interior Point Algorithms for Linear Complementarity Problems*, Lecture Notes in Comput. Sci. 5328, Springer-Verlag, Berlin, 1991.

[23] R. Kuske and J. Keller, *Optimal exercise boundary for an American put option*, Appl. Math. Fin., 5 (1998), pp. 107–116.

[24] R. Nochetto, *Sharp l^∞ error estimates for semilinear problems with free boundaries*, Num. Math., 54 (1988), pp. 243–255.

[25] J. Outrata, M. Kocvara, and J. Zowe, *Nonsmooth Approach to Optimization Problems with Equilibrium Constraints*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1998.

[26] J.-S. Pang, *A B-differential equation-based, globally and locally quadratically convergent algorithm for nonlinear programs*, Math. Programming, 51 (1991), pp. 101–131.

[27] J.-S. Pang and L. Qi, *Nonsmooth equations: Motivation and algorithms*, SIAM J. Optim., 3 (1993), pp. 443–465.

[28] R. Rannacher, *Finite element solution of diffusion problems with irregular data*, Numer. Math., 43 (1984), pp. 309–327.

[29] I. Rupf, J. Dewynne, S. Howison, and P. Wilmott, *Some mathematical results in the pricing of American options*, European J. Appl. Math., 4 (1993), pp. 381–398.

[30] R. Scholz, *Numerical solution of the obstacle problem by a penalty method: Part* II. *Time dependent problems*, Numer. Math., 49 (1986), pp. 255–268.

[31] K. Vetzal and P. Forsyth, *Discrete Parisian and delayed barrier options: A general numerical approach*, Adv. Fut. Opt. Res., 10 (1999), pp. 1–15.

[32] P. Wilmott, *Derivatives*, Wiley, New York, 1998.

[33] P. Wilmott, J. Dewynne, and S. Howison, *Option Pricing*, Oxford Financial Press, Oxford, UK, 1993.

[34] H. Windcliff, P. Forsyth, and K. Vetzal, *Shout options: A framework for pricing contracts which can be modified by the investor*, J. Comput. Appl. Math., 134 (2001), pp. 213–241.

[35] H. Windcliff, P. Forsyth, and K. Vetzal, *Valuation of segregated funds: Shout options with maturity extensions*, Insurance Math. Econom., 29 (2001), pp. 1–21.

[36] H. Windcliff, K. Vetzal, P. Forsyth, A. Verma, and T. Coleman, *An object-oriented framework for valuing shout options on high-performance computer architectures*, J. Econ. Dyn. Con., to appear.

[37] R. Zvan, P. Forsyth, and K. Vetzal, *Robust numerical methods for PDE models of Asian options*, J. Comp. Fin., 1 (1998), pp. 39–78.

[38] R. Zvan, P. Forsyth, and K. Vetzal, *Discrete Asian barrier options*, J. Comp. Fin., 3 (1999), pp. 41–68.

[39] R. Zvan, P. Forsyth, and K. Vetzal, *A finite volume approach to contingent claims valuations*, IMA J. Numer. Anal., 21 (2001), pp. 703–731.

[40] R. Zvan, P. A. Forsyth, and K. R. Vetzal, *Penalty methods for American options with stochastic volatility*, J. Comput. Appl. Math., 91 (1998), pp. 119–218.

[41] R. Zvan, K. Vetzal, and P. Forsyth, *Swing low, swing high*, RISK, 11 (1998), pp. 71–75.

[42] R. Zvan, K. Vetzal, and P. Forsyth, *PDE methods for pricing barrier options*, J. Econ. Dyn. Con., 24 (2000), pp. 1563–1590.

SIAM J. Sci. Comput.
Vol. 23, No. 6, pp. 2123–2140

A NEW METHOD FOR MICRO-MACRO SIMULATIONS OF VISCOELASTIC FLOWS[*]

C. CHAUVIERE[†]

Abstract. In this paper we develop a robust numerical method derived from the Brownian configuration field method [M.A. Hulsen, A.P.G. Van Heel, and B.H.A.A. Van Den Brule, *J. Non-Newtonian Fluid Mech.*, 70 (1997), pp. 79–101] for the simulation of flows of dilute polymeric solutions. The statistical properties of the Wiener stochastic process in the stochastic differential equation describing the evolution of the configuration vector are exploited in order to derive a simple expression for the polymeric contribution to the stress. The method is tested numerically by solving the benchmark problem of the flow of an Oldroyd B fluid past a cylinder in a channel using a spectral element method. Results are presented to demonstrate the advantages of the proposed method.

Key words. viscoelastic flows, spectral element methods, mesoscopic models

AMS subject classifications. 76A10, 65N35

PII. S1064827501384603

1. Introduction.

Among the more important recent advances in computational rheology, the use of kinetic theory models to evaluate the polymeric contribution to the stress ranks highly. Whereas in macroscopic methods closed-form constitutive equations have been used, these have often been of limited use in faithfully simulating complex polymeric flows. In bypassing a macroscopic description of the stress evolution and using a coarse-grained microscopic description instead, more of the physics of viscoelastic fluids may be incorporated. The use of stochastic differential equations for the simulation of complex viscoelastic flows was initiated in 1993 by Laso and Öttinger [13]. Then, a new and powerful Brownian dynamics technique, the so-called *Brownian configuration field method* was introduced by Hulsen, Van Heel, and Van Den Brule [12]. Here, the idea is that a stochastic differential equation is solved for *ensembles* of conformation vectors corresponding to dumbbell connectors having the same initial configuration and subject to the same Brownian forces in time. This technique avoids the difficulties associated with the tracking of individual particles. However, as with all micro-macro simulations to date, the computations are significantly more costly in terms of time and memory than is the case with closed-form constitutive equations. The main reason for this extra cost is that a significant number of configuration fields (typically several thousand) have to be computed and stored in order to reduce the noise in the computed polymeric stress. Despite the success enjoyed by variance reduction methods [3] in reducing the number of degrees of freedom required in the problem, Brownian configuration methods remain expensive. In the case of Hookean dumbbells, the present paper is intended to be a step towards making such methods more affordable.

The paper is organized as follows. In the next section, we describe two common approaches for the numerical calculation of the flow of an Oldroyd B fluid. Then, in the case of mesoscopic calculations, section 3 is devoted to an explanation of a new

[*] Received by the editors February 7, 2001; accepted for publication (in revised form) October 26, 2001; published electronically February 27, 2002.
http://www.siam.org/journals/sisc/23-6/38460.html

[†] DGM, Ecole Polytechnique Fédérale de Lausanne, CH-1015 Lausanne, Switzerland. Current address: Division of Applied Mathematics, Brown University, Box F, Providence, RI 02912 (cedric@cfm.brown.edu).

way of computing the extra-stress which overcomes the CPU and memory limitations of the Brownian configuration field method. In sections 4 and 5, we discretize the governing equations using a spectral element method for the benchmark problem of an Oldroyd B fluid past a cylinder in a channel. Finally, in section 6 we show how the present results can be extended to the Oldroyd B multimode model.

2. Problem description.

Let $\Omega \subset \mathbb{R}^d$ be some bounded Lipschitz domain having boundary $\partial\Omega$. We consider inertialess, isothermal flow of an incompressible fluid with density ρ for which the momentum conservation and the continuity equations in Ω are

$$\rho \frac{\partial \mathbf{u}}{\partial t} - \eta_s \boldsymbol{\Delta} \mathbf{u} + \boldsymbol{\nabla} p = \boldsymbol{\nabla} . \boldsymbol{\tau} + \mathbf{f}, \tag{2.1}$$

$$\boldsymbol{\nabla} . \mathbf{u} = 0, \tag{2.2}$$

where $\mathbf{u}$ denotes the fluid velocity, p is the pressure, $\boldsymbol{\tau}$ is the polymeric contribution to the stress tensor, $\mathbf{f}$ is a body force, and η_s is the solvent viscosity. The extra-stress $\boldsymbol{\tau}$ needs an equation to close the system. For dilute solutions of polymers, this may be a differential or integral constitutive equation (macroscopic approach) or an expression derived from a macromolecular model (mesoscopic approach) in which case a stochastic differential equation needs to be solved. Some models such as the Oldroyd B model have both a constitutive equation and a macromolecular model representation. In the next two subsections, we will describe the two approaches for this model.

2.1. Macroscopic approach: Constitutive equation.

For an Oldroyd B fluid and a time dependent flow, the polymeric extra-stress tensor satisfies the following hyperbolic equation:

$$\boldsymbol{\tau} + \lambda \overset{\triangledown}{\boldsymbol{\tau}} = \eta_p \dot{\boldsymbol{\gamma}}(\mathbf{u}), \tag{2.3}$$

where λ is the relaxation time of the fluid, $\dot{\boldsymbol{\gamma}}(\mathbf{u}) = (\boldsymbol{\nabla}\mathbf{u} + \boldsymbol{\nabla}\mathbf{u}^T)$ is the rate of strain tensor, and η_p is the polymeric fluid viscosity. Given the velocity field $\mathbf{u}$, the upper-convected derivative $\triangledown$ in (2.3) is then defined by

$$\overset{\triangledown}{\boldsymbol{\tau}} = \frac{\partial \boldsymbol{\tau}}{\partial t} + \mathbf{u}.\nabla\boldsymbol{\tau} - \nabla\mathbf{u}.\boldsymbol{\tau} - (\nabla\mathbf{u}.\boldsymbol{\tau})^T. \tag{2.4}$$

Usually, a decoupled numerical scheme is used, i.e., for a given source term $\boldsymbol{\tau}$, (2.1) and (2.2) form a standard Stokes problem, and once the velocity has been computed, the extra-stress can be determined from (2.3). The advantage of constitutive equations is their simplicity: a differential equation just needs to be discretized with possibly some stabilizing techniques to account for its hyperbolic character. The CPU and memory requirements may be less than for its mesoscopic counterpart, and noise-free solutions can be obtained. Viscoelastic flows are characterized by a Deborah number (see (5.1)), which is a nondimensional measure of the elasticity of the fluid. The macroscopic approach is less robust (it is possible that only simulations at modest Deborah numbers can be performed), and a closed form equation may not be available for the most general models.

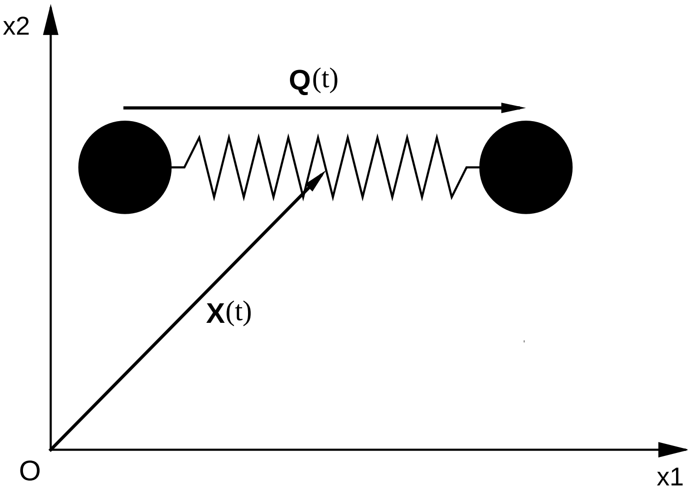


FIG. 1. *Description of a single dumbbell placed in the fluid in* ***X****(t).*

2.2. Mesoscopic approach. In this subsection, we briefly describe *the Brownian configuration field method* [12, 15, 20] first introduced in 1997 by Hulsen et al. The idea of this mesoscopic approach is to model the polymer chains by elastic dumbbells that consist of two beads connected by a spring (see Figure 1). Then, we model the polymer chains and their interactions with the solvent. The *configuration vector* $\mathbf{Q}(t)$ describes the configuration of a dumbbell, i.e., the orientation and elongation in space of the spring connecting the two beads as it moves along its trajectory $\mathbf{X}(t)$. Note that $\mathbf{X}(t)$ is nothing but the trajectory of the fluid particles. For the Oldroyd B model, the spring force $\mathbf{F}$ is proportional to the configuration vector

$$\mathbf{F} = H\mathbf{Q}(t), \tag{2.5}$$

where H is the spring constant.

In order to reduce the number of parameters, the configuration vector is scaled by a characteristic length to give $\widetilde{\mathbf{Q}}(t)$

$$\widetilde{\mathbf{Q}}(t) = \frac{\mathbf{Q}(t)}{\sqrt{\frac{k_B T}{H}}},$$

where k_B is Boltzmann's constant and T the absolute temperature. Adopting an Eulerian point of view and taking the same initial configuration and the same Brownian forces in time for ensembles of dumbbells leads to the configuration field method. Then, the configuration field $\widetilde{\mathbf{Q}}(\mathbf{x},t)$ satisfies the following stochastic differential equation (see [16] for details):

$$\begin{aligned} d\widetilde{\mathbf{Q}}(\mathbf{x},t) = \left(-\mathbf{u}(\mathbf{x},t).\nabla\widetilde{\mathbf{Q}}(\mathbf{x},t) + \nabla\mathbf{u}(\mathbf{x},t)\widetilde{\mathbf{Q}}(\mathbf{x},t) - \frac{1}{2\lambda}\widetilde{\mathbf{Q}}(\mathbf{x},t)\right) dt \\ + \sqrt{\frac{1}{\lambda}} d\mathbf{W}(t). \end{aligned} \tag{2.6}$$

In (2.6), $\mathbf{W}(t)$ is the so-called Wiener process which accounts for the Brownian forces acting on each bead. The Wiener process has some important properties that will be used in the next section to derive a simple expression for the extra-stress. It is a Gaussian process with zero mean and covariance $\langle \mathbf{W}(t)\mathbf{W}(t')\rangle = \min(t,t')\mathbf{I}$. It should also be noted that each component of $\mathbf{W}(t)$ depends only on time and has a constant spatial value all over the domain Ω.

The local value of the extra-stress $\boldsymbol{\tau}(\mathbf{x},t)$ (which is a macroscopic quantity) can be determined from an ensemble of N_{cf} configuration fields $\{\widetilde{\mathbf{Q}}_m(\mathbf{x},t)\}_{1\leq m\leq N_{cf}}$ that experience different uncorrelated Wiener processes $\{\mathbf{W}_m(t)\}_{1\leq m\leq N_{cf}}$. For the Oldroyd B model, $\boldsymbol{\tau}(\mathbf{x},t)$ is given by the *Kramers expression* (see page 69 of [1])

$$\boldsymbol{\tau}(\mathbf{x},t) = \frac{\eta_p}{\lambda}\left(-\mathbf{I} + \lim_{N_{cf}\to\infty}\left(\frac{1}{N_{cf}}\sum_{m=1}^{N_{cf}} \widetilde{\mathbf{Q}}_m(\mathbf{x},t)\otimes\widetilde{\mathbf{Q}}_m(\mathbf{x},t)\right)\right), \tag{2.7}$$

where the symbol $\otimes$ denotes the tensor product of two vectors. Initially, the fluid is at rest so that the equilibrium extra-stress tensor must be zero. Each component of the initial configuration fields $\widetilde{\mathbf{Q}}_m(\mathbf{x},t_0)$ satisfies a space-independent scalar normal law with zero mean and unity variance [2], and thus

$$\lim_{N_{cf}\to\infty}\left(\frac{1}{N_{cf}}\sum_{m=1}^{N_{cf}} \widetilde{\mathbf{Q}}_m(\mathbf{x},t_0)\otimes\widetilde{\mathbf{Q}}_m(\mathbf{x},t_0)\right) = \mathbf{I}. \tag{2.8}$$

In order to state more precisely the typical CPU and memory cost required by a mesoscopic computation, let us use a simple implicit Euler scheme for the time discretization of (2.6). For a given uniform partition $0 = t_0 < t_1 < \cdots < t_{\max}$ of the time range $[0, t_{\max}]$ with $t_i = i\Delta t$, $(i = 1,\ldots,\max)$, the mth configuration field to be used in (2.7) satisfies

$$\begin{aligned}\mathbf{Q}_m(\mathbf{x},t_i) + &\left(\mathbf{u}(\mathbf{x},t_i).\nabla\mathbf{Q}_m(\mathbf{x},t_i) - \nabla\mathbf{u}(\mathbf{x},t_i)\mathbf{Q}_m(\mathbf{x},t_i) + \frac{1}{2\lambda}\mathbf{Q}_m(\mathbf{x},t_i)\right)\Delta t\\ &= \mathbf{Q}_m(\mathbf{x},t_{i-1}) + \sqrt{\frac{\Delta t}{\lambda}}\Delta\mathbf{W}_m(t_i).\end{aligned} \tag{2.9}$$

The tildes of the configuration field vectors have been dropped for better readability and in what follows it is understood that $\mathbf{Q}_m(\mathbf{x},t)$ is a dimensionless quantity. The Wiener process has also been scaled by $\sqrt{\Delta t}$ so that we now have $\langle \Delta\mathbf{W}_m(t_i)^2\rangle = \langle(\mathbf{W}_m(t_i) - \mathbf{W}_m(t_{i-1}))^2\rangle = \mathbf{I}$. In practice, it is not possible to take an infinite number of configuration fields in order to evaluate the extra-stress with (2.7), and, instead, a finite number of them is used. According to the *central-limit theorem*, for N_{cf} sufficiently large, the statistical error of the extra-stress is proportional to $1/\sqrt{N_{cf}}$. Therefore, the numerical solution converges towards the noise-free solution at a slow rate of order $\mathcal{O}(1/\sqrt{N_{cf}})$. This means that in order to get solutions with low noise level, (2.9) has to be solved a great number of times (typically, $N_{cf} = \mathcal{O}(10^3)$) and explains why the mesoscopic approach is so CPU intensive. It also means that many spatial vectors $\{\mathbf{Q}_m(\mathbf{x},t_{i-1})\}_{1\leq m\leq N_{cf}}$ coming from the previous calculation at time t_{i-1} must be stored in order to evaluate the right-hand side of (2.9) at time t_i. Variance reduction techniques make it possible to reduce the noise level [3]; however, the efficiency of such techniques may be less effective as the Deborah number increases

[2]. In the next section, we will see how the properties of the Wiener process may be exploited in order to get noise-free extra-stresses at a relatively low CPU and memory cost.

3. A new method for noise-free extra-stresses.

We first introduce some notation that will be useful in what follows. We denote by $\mathbf{q}^{t_l}(\mathbf{x}, t_i)$ a d-dimensional vector whose value is updated at time $t_i = i\Delta t$. The superscript t_l indicates that this vector was originally created at time $t_l = l\Delta t$ (with $l \leq i$). In order to model the Wiener process of (2.9) at time t_i, it is convenient to introduce d sets of N_{cf} real random numbers $\{\Delta W_m^j(t_i)\}_{1\leq m\leq N_{cf}}$ ($j = 1, \ldots, d$) obeying a scalar normal law. Then, we introduce d constant d-dimensional vectors $\{\mathbf{b}_j\}_{1\leq j\leq d}$, where $\mathbf{b}_j$ has unit value at its jth component and zero elsewhere. With those definitions, the mth Wiener process of (2.9) can be written

$$\Delta\mathbf{W}_m(t_i) = \sum_{j=1}^{d} \mathbf{b}_j \Delta W_m^j(t_i). \tag{3.1}$$

Finally, we define a real function of the relaxation time λ for any given time step Δt; thus

$$\alpha(\lambda) = \sqrt{\frac{\Delta t}{\lambda}}. \tag{3.2}$$

For any kind of spatial discretization of (2.9) (e.g., a finite element method, finite difference method, or spectral element method), we are now in position to prove the following theorem that will be used to derive a new expression for the extra-stress.

THEOREM 3.1. *Let $\mathbf{Q}_m(\mathbf{x},t_i)$ be the discrete solution of* (2.9) *at time $t_i = i\Delta t$ ($i \geq 1$) for any given Wiener process $\Delta\mathbf{W}_m(t_i)$. Then, there exists $d \times i$ vectors independent of m, denoted by $\{\mathbf{q}_j^{t_{i-l}}(\mathbf{x}, t_i)\}_{0\leq l\leq i-1}^{1\leq j\leq d}$, such that*

$$\mathbf{Q}_m(\mathbf{x},t_i) = \sum_{j=1}^{d} \left\{ \left(\sum_{l=0}^{i-1} \alpha(\lambda)\mathbf{q}_j^{t_{i-l}}(\mathbf{x}, t_i)\Delta W_m^j(t_{i-l}) \right) + \mathbf{q}_j^{t_1}(\mathbf{x}, t_i)\Delta W_m^j(t_0) \right\}. \tag{3.3}$$

Proof. We will prove this theorem by induction on i. We first show that (3.3) is true for $i = 1$. At $t = t_0$, the configuration fields are at equilibrium, and, for the Oldroyd B model, they come from the same distribution as the Wiener processes so that we can write $\mathbf{Q}_m(\mathbf{x},t_0) = \Delta\mathbf{W}_m(t_0)$. The relation (3.1) can be applied at time t_0 to give

$$\mathbf{Q}_m(\mathbf{x},t_0) = \Delta\mathbf{W}_m(t_0) = \sum_{j=1}^{d} \mathbf{b}_j \Delta W_m^j(t_0). \tag{3.4}$$

Let $\mathbf{E}_\mathbf{u}(\mathbf{x},t_i)$ be the matrix resulting from the discretization of the left-hand side of (2.9) at time t_i for a given velocity field $\mathbf{u}$. Then, at time t_1, (2.9) can be formally rewritten as

$$\begin{aligned} \mathbf{E}_\mathbf{u}(\mathbf{x}, t_1)\mathbf{Q}_m(\mathbf{x},t_1) &= \alpha(\lambda)\Delta\mathbf{W}_m(t_1) + \mathbf{Q}_m(\mathbf{x},t_0) \\ &= \alpha(\lambda)\Delta\mathbf{W}_m(t_1) + \sum_{j=1}^{d} \mathbf{b}_j \Delta W_m^j(t_0), \end{aligned} \tag{3.5}$$

where we have replaced $\mathbf{Q}_m(\mathbf{x},t_0)$ by its equivalent expression (3.4).

Multiplying (3.5) throughout by $\mathbf{E}_{\mathbf{u}}^{-1}(\mathbf{x},t_1)$, and using equation (3.1) at time t_1 for $\Delta\mathbf{W}_m(t_1)$, we obtain

$$\mathbf{Q}_m(\mathbf{x},t_1) = \sum_{j=1}^{d} \left(\alpha(\lambda)\mathbf{E}_{\mathbf{u}}^{-1}(\mathbf{x},t_1)\mathbf{b}_j\Delta W_m^j(t_1) + \mathbf{E}_{\mathbf{u}}^{-1}(\mathbf{x},t_1)\mathbf{b}_j\Delta W_m^j(t_0)\right). \tag{3.6}$$

We can now define a new vector $\mathbf{q}_j^{t_1}(\mathbf{x},t_1)$ created at time t_1 by

$$\mathbf{q}_j^{t_1}(\mathbf{x},t_1) = \mathbf{E}_{\mathbf{u}}^{-1}(\mathbf{x},t_1)\mathbf{b}_j \tag{3.7}$$

so that we can rewrite (3.6) in the form

$$\mathbf{Q}_m(\mathbf{x},t_1) = \sum_{j=1}^{d} \left(\alpha(\lambda)\mathbf{q}_j^{t_1}(\mathbf{x},t_1)\Delta W_m^j(t_1) + \mathbf{q}_j^{t_1}(\mathbf{x},t_1)\Delta W_m^j(t_0)\right). \tag{3.8}$$

The above equation shows that the relation (3.3) is true for $i = 1$.

We now suppose that the relation (3.3) is true at time t_{i-1}, and we will prove that it is true at time t_i. At time t_i, (2.9) can be formally rewritten as

$$\mathbf{E}_{\mathbf{u}}(\mathbf{x},t_i)\mathbf{Q}_m(\mathbf{x},t_i) = \alpha(\lambda)\Delta\mathbf{W}_m(t_i) + \mathbf{Q}_m(\mathbf{x},t_{i-1}). \tag{3.9}$$

By the inductive hypothesis the relation (3.3) is true at time t_{i-1} so that we have

$$\begin{aligned}\mathbf{Q}_m(\mathbf{x},t_{i-1}) &= \sum_{j=1}^{d}\left\{\left(\sum_{l=0}^{i-2}\alpha(\lambda)\mathbf{q}_j^{t_{i-1-l}}(\mathbf{x},t_{i-1})\Delta W_m^j(t_{i-1-l})\right) + \mathbf{q}_j^{t_1}(\mathbf{x},t_{i-1})\Delta W_m^j(t_0)\right\}\\ &= \sum_{j=1}^{d}\left\{\left(\sum_{l=1}^{i-1}\alpha(\lambda)\mathbf{q}_j^{t_{i-l}}(\mathbf{x},t_{i-1})\Delta W_m^j(t_{i-l})\right) + \mathbf{q}_j^{t_1}(\mathbf{x},t_{i-1})\Delta W_m^j(t_0)\right\},\end{aligned} \tag{3.10}$$

where the last expression has been obtained after a change of indices for l.

Multiplying (3.9) throughout by $\mathbf{E}_{\mathbf{u}}^{-1}(\mathbf{x},t_i)$ and replacing $\mathbf{Q}_m(\mathbf{x},t_{i-1})$ by its expression above, we obtain

$$\begin{aligned}\mathbf{Q}_m(\mathbf{x},t_i) = \sum_{j=1}^{d}\Bigg\{&\left(\sum_{l=1}^{i-1}\alpha(\lambda)\mathbf{E}_{\mathbf{u}}^{-1}(\mathbf{x},t_i)\mathbf{q}_j^{t_{i-l}}(\mathbf{x},t_{i-1})\Delta W_m^j(t_{i-l})\right)\\ &+ \mathbf{E}_{\mathbf{u}}^{-1}(\mathbf{x},t_i)\mathbf{q}_j^{t_1}(\mathbf{x},t_{i-1})\Delta W_m^j(t_0)\Bigg\} + \alpha(\lambda)\mathbf{E}_{\mathbf{u}}^{-1}(\mathbf{x},t_i)\Delta\mathbf{W}_m(t_i).\end{aligned} \tag{3.11}$$

We define the vector $\mathbf{q}_j^{t_{i-l}}(\mathbf{x},t_i)$ created at time t_{i-l} and updated at time t_i as the function of the vector $\mathbf{q}_j^{t_{i-l}}(\mathbf{x},t_{i-1})$ created at the same time t_{i-l} and updated at the previous time step t_{i-1} by setting

$$\mathbf{q}_j^{t_{i-l}}(\mathbf{x},t_i) = \mathbf{E}_{\mathbf{u}}^{-1}(\mathbf{x},t_i)\mathbf{q}_j^{t_{i-l}}(\mathbf{x},t_{i-1}) \text{ for } l = 1,\dots,i-1. \tag{3.12}$$

Using the above expressions of $\mathbf{q}_j^{t_{i-l}}(\mathbf{x},t_i)$ and replacing the Wiener process $\Delta\mathbf{W}_m(t_i)$

by its equivalent expression (3.1), equation (3.11) becomes

$$\begin{aligned}(3.13)\quad \mathbf{Q}_m(\mathbf{x},t_i) = \sum_{j=1}^{d} \Bigg\{ \Bigg(\sum_{l=1}^{i-1} \alpha(\lambda) \mathbf{q}_j^{t_{i-l}}(\mathbf{x},t_i) \Delta W_m^j(t_{i-l}) \Bigg) \\ + \mathbf{q}_j^{t_1}(\mathbf{x},t_i)\Delta W_m^j(t_0) \Bigg\} + \sum_{j=1}^{d} \alpha(\lambda) \mathbf{E}_{\mathbf{u}}^{-1}(\mathbf{x},t_i)\mathbf{b}_j \Delta W_m^j(t_i).\end{aligned}$$

We can now define the newly created vectors $\mathbf{q}_j^{t_i}(\mathbf{x},t_i)$ at time t_i by

$$(3.14)\qquad \mathbf{q}_j^{t_i}(\mathbf{x},t_i) = \mathbf{E}_{\mathbf{u}}^{-1}(\mathbf{x},t_i)\mathbf{b}_j;$$

then (3.13) can be written

$$\begin{aligned}\mathbf{Q}_m(\mathbf{x},t_i) &= \sum_{j=1}^{d} \Bigg\{ \Bigg(\sum_{l=1}^{i-1} \alpha(\lambda) \mathbf{q}_j^{t_{i-l}}(\mathbf{x},t_i) \Delta W_m^j(t_{i-l}) \Bigg) \\ (3.15)\qquad &\quad + \alpha(\lambda)\mathbf{q}_j^{t_i}(\mathbf{x},t_i)\Delta W_m^j(t_i) + \mathbf{q}_j^{t_1}(\mathbf{x},t_i)\Delta W_m^j(t_0) \Bigg\} \\ &= \sum_{j=1}^{d} \Bigg\{ \Bigg(\sum_{l=0}^{i-1} \alpha(\lambda) \mathbf{q}_j^{t_{i-l}}(\mathbf{x},t_i) \Delta W_m^j(t_{i-l}) \Bigg) + \mathbf{q}_j^{t_1}(\mathbf{x},t_i)\Delta W_m^j(t_0) \Bigg\}.\end{aligned}$$

The above relation shows that (3.3) is true at time t_i as long as it is satisfied at time t_{i-1}. Since (3.3) is true at the initial time t_1, by induction, the relation is valid for any time $t_i = i\Delta t$ ($i \geq 1$), and this concludes the proof. □

The proof of Theorem 3.1 gives an explicit expression for the vectors $\mathbf{q}_j^{t_{i-l}}(\mathbf{x},t_i)$ which can be obtained with (3.14) at their time of creation and subsequently updated according to (3.12). The next theorem shows how the extra-stress can be computed from the vectors $\{\mathbf{q}_j^{t_{i-l}}(\mathbf{x},t_i)\}_{0\leq l\leq i-1}^{1\leq j\leq d}$.

THEOREM 3.2. *The extra-stress tensor $\boldsymbol{\tau}(\mathbf{x},t_i)$ at time t_i can be computed from the vectors $\{\mathbf{q}_j^{t_{i-l}}(\mathbf{x},t_i)\}_{0\leq l\leq i-1}^{1\leq j\leq d}$ in a deterministic way by*

$$\boldsymbol{\tau}(\mathbf{x},t_i) = \frac{\eta_p}{\lambda}\Bigg(-\mathbf{I} + \sum_{j=1}^{d} \Bigg\{ \Bigg(\sum_{l=0}^{i-1} \alpha(\lambda)^2 \mathbf{q}_j^{t_{i-l}}(\mathbf{x},t_i) \otimes \mathbf{q}_j^{t_{i-l}}(\mathbf{x},t_i) \Bigg) + \mathbf{q}_j^{t_1}(\mathbf{x},t_i) \otimes \mathbf{q}_j^{t_1}(\mathbf{x},t_i) \Bigg\} \Bigg).$$
(3.16)

Proof. According to (2.7), the extra-stress tensor $\boldsymbol{\tau}(\mathbf{x},t_i)$ at time t_i is given by

$$(3.17)\quad \boldsymbol{\tau}(\mathbf{x},t_i) = \frac{\eta_p}{\lambda}\left(-\mathbf{I} + \lim_{N_{cf}\to\infty} \left(\frac{1}{N_{cf}} \sum_{m=1}^{N_{cf}} \mathbf{Q}_m(\mathbf{x},t_i) \otimes \mathbf{Q}_m(\mathbf{x},t_i) \right) \right).$$

Before passing to the limit when $N_{cf} \to \infty$, we rewrite $\mathbf{Q}_m(\mathbf{x},t_i) \otimes \mathbf{Q}_m(\mathbf{x},t_i)$ by replacing $\mathbf{Q}_m(\mathbf{x},t_i)$ by its expression (3.3) established in Theorem 3.1, and, using the distributivity of the tensor product, we have

$$
\begin{aligned}
\mathbf{Q}_m(\mathbf{x},t_i) \otimes \mathbf{Q}_m(\mathbf{x},t_i) &= \sum_{j=1}^{d} \left\{ \left(\sum_{l=0}^{i-1} \alpha(\lambda) \mathbf{q}_j^{t_{i-l}}(\mathbf{x},t_i) \Delta W_m^j(t_{i-l}) \right) + \mathbf{q}_j^{t_1}(\mathbf{x},t_i) \Delta W_m^j(t_0) \right\} \\
&\otimes \sum_{j'=1}^{d} \left\{ \left(\sum_{l'=0}^{i-1} \alpha(\lambda) \mathbf{q}_{j'}^{t_{i-l'}}(\mathbf{x},t_i) \Delta W_m^{j'}(t_{i-l'}) \right) + \mathbf{q}_{j'}^{t_1}(\mathbf{x},t_i) \Delta W_m^{j'}(t_0) \right\} \\
&= \sum_{j=1}^{d} \sum_{l=0}^{i-1} \sum_{j'=1}^{d} \sum_{l'=0}^{i-1} \alpha(\lambda)^2 \Delta W_m^j(t_{i-l}) \Delta W_m^{j'}(t_{i-l'}) \mathbf{q}_j^{t_{i-l}}(\mathbf{x},t_i) \otimes \mathbf{q}_{j'}^{t_{i-l'}}(\mathbf{x},t_i) \\
&+ \sum_{j=1}^{d} \sum_{l=0}^{i-1} \sum_{j'=1}^{d} \alpha(\lambda) \Delta W_m^j(t_{i-l}) \Delta W_m^{j'}(t_0) \mathbf{q}_j^{t_{i-l}}(\mathbf{x},t_i) \otimes \mathbf{q}_{j'}^{t_1}(\mathbf{x},t_i) \\
&+ \sum_{j=1}^{d} \sum_{j'=1}^{d} \sum_{l'=0}^{i-1} \alpha(\lambda) \Delta W_m^j(t_0) \Delta W_m^{j'}(t_{i-l'}) \mathbf{q}_j^{t_1}(\mathbf{x},t_i) \otimes \mathbf{q}_{j'}^{t_{i-l'}}(\mathbf{x},t_i) \\
&+ \sum_{j=1}^{d} \sum_{j'=1}^{d} \Delta W_m^j(t_0) \Delta W_m^{j'}(t_0) \mathbf{q}_j^{t_1}(\mathbf{x},t_i) \otimes \mathbf{q}_{j'}^{t_1}(\mathbf{x},t_i).
\end{aligned} \tag{3.18}
$$

In order to simplify this expression, we note that $\{\Delta W_m^j(t_l)\}_{1 \leq m \leq N_{cf}}$ and $\{\Delta W_m^{j'}(t_{l'})\}_{1 \leq m \leq N_{cf}}$ are two sets of random numbers following the scalar normal law. If they are generated independently, i.e., they come from two different times $t_l \neq t_{l'}$ or different components $j \neq j'$, their covariance is zero. Those properties can be summarized as follows:

$$
\lim_{N_{cf} \to \infty} \frac{1}{N_{cf}} \sum_{m=1}^{N_{cf}} \Delta W_m^j(t_l) \Delta W_m^{j'}(t_{l'}) = \delta_{jj'} \delta_{ll'}, \tag{3.19}
$$

where δ is the Kronecker delta. Using (3.18) and (3.19), we can now simplify the expression $\frac{1}{N_{cf}} \sum_{m=1}^{N_{cf}} \mathbf{Q}_m(\mathbf{x},t_i) \otimes \mathbf{Q}_m(\mathbf{x},t_i)$ when $N_{cf} \to \infty$:

$$
\begin{aligned}
&\lim_{N_{cf} \to \infty} \left(\frac{1}{N_{cf}} \sum_{m=1}^{N_{cf}} \mathbf{Q}_m(\mathbf{x},t_i) \otimes \mathbf{Q}_m(\mathbf{x},t_i) \right) \\
&= \sum_{j=1}^{d} \sum_{l=0}^{i-1} \sum_{j'=1}^{d} \sum_{l'=0}^{i-1} \alpha(\lambda)^2 \delta_{jj'} \delta_{i-l,i-l'} \mathbf{q}_j^{t_{i-l}}(\mathbf{x},t_i) \otimes \mathbf{q}_{j'}^{t_{i-l'}}(\mathbf{x},t_i) \\
&+ \sum_{j=1}^{d} \sum_{l=0}^{i-1} \sum_{j'=1}^{d} \alpha(\lambda) \delta_{jj'} \delta_{i-l,0} \mathbf{q}_j^{t_{i-l}}(\mathbf{x},t_i) \otimes \mathbf{q}_{j'}^{t_1}(\mathbf{x},t_i) \\
&+ \sum_{j=1}^{d} \sum_{j'=1}^{d} \sum_{l'=0}^{i-1} \alpha(\lambda) \delta_{jj'} \delta_{0,i-l'} \mathbf{q}_j^{t_1}(\mathbf{x},t_i) \otimes \mathbf{q}_{j'}^{t_{i-l'}}(\mathbf{x},t_i) \\
&+ \sum_{j=1}^{d} \sum_{j'=1}^{d} \delta_{jj'} \delta_{00} \mathbf{q}_j^{t_1}(\mathbf{x},t_i) \otimes \mathbf{q}_{j'}^{t_1}(\mathbf{x},t_i) \\
&= \sum_{j=1}^{d} \left(\sum_{l=0}^{i-1} \alpha(\lambda)^2 \mathbf{q}_j^{t_{i-l}}(\mathbf{x},t_i) \otimes \mathbf{q}_j^{t_{i-l}}(\mathbf{x},t_i) \right) + \sum_{j=1}^{d} \mathbf{q}_j^{t_1}(\mathbf{x},t_i) \otimes \mathbf{q}_j^{t_1}(\mathbf{x},t_i).
\end{aligned} \tag{3.20}
$$

With the result above and (3.17), we find the required relation for the extra-stress $\boldsymbol{\tau}(\mathbf{x},t_i)$

$$\boldsymbol{\tau}(\mathbf{x},t_i) = \frac{\eta_p}{\lambda}\Bigg(-\mathbf{I}+\sum_{j=1}^{d}\Bigg\{\Bigg(\sum_{l=0}^{i-1}\alpha(\lambda)^2\mathbf{q}_j^{t_{i-l}}(\mathbf{x},t_i)\otimes\mathbf{q}_j^{t_{i-l}}(\mathbf{x},t_i)\Bigg) + \mathbf{q}_j^{t_1}(\mathbf{x},t_i)\otimes\mathbf{q}_j^{t_1}(\mathbf{x},t_i)\Bigg\}\Bigg). \qquad \square \tag{3.21}$$

The above equation gives a nice interpretation of the so-called *memory effects* which are of great importance for viscoelastic fluids. Indeed, (3.21) shows that in order to compute the extra-stress $\boldsymbol{\tau}(\mathbf{x},t_i)$ at time t_i, we need to know the history of the fluid at times $\{t_l\}_{1\leq l\leq i}$ which is contained in the vectors $\{\mathbf{q}_j^{t_{i-l}}(\mathbf{x},t_i)\}_{0\leq l\leq i-1}$. Furthermore, this relation is entirely deterministic and gives noise-free solutions. We also see that at every new time step, d d-dimensional vectors have to be created and stored. Therefore, the computation becomes more and more expensive as we advance in time. This is why for the time discretization of (2.6), we have adopted an implicit Euler scheme which allows us to take larger time steps and to reach higher Deborah numbers than its explicit counterpart. In (3.21), the vectors $\{\mathbf{q}_j^{t_{i-l}}(\mathbf{x},t_i)\}_{0\leq l\leq i-1}$ receive equal weighting. If, for efficiency purposes, the "old" vectors $\mathbf{q}_j^{t_i}$ are removed before the steady state is reached, then information is lost and the variables $(\mathbf{u},p,\boldsymbol{\tau})$ will not converge to the right solution. In the next section, we apply this new method using a spectral element method for spatial discretization of (2.1), (2.2), and (2.9).

4. Weak formulation and spectral element discretization.

4.1. Functional spaces and variational formulation. We now introduce the following linear spaces over the flow domain $\Omega \subset \mathbb{R}$ for the velocity, pressure, and extra-stress, respectively:

$$V = \left\{\mathbf{u}\in\left(H^1(\Omega)\right)^2 : \mathbf{u}=\mathbf{0} \text{ on } \partial\Omega_0 \neq \emptyset\right\}, \tag{4.1}$$

$$P = L_0^2(\Omega), \tag{4.2}$$

$$\Sigma = \left\{\boldsymbol{\tau} : \boldsymbol{\tau}\in[L^2(\Omega)]_{\mathrm{s}}^{2\times 2}\right\}. \tag{4.3}$$

Given the velocity field **u,** the *inflow boundary* $\partial\Omega^-$ of Ω having boundary $\partial\Omega$ is defined in the usual way by

$$\partial\Omega^- = \{\mathbf{x}\in\partial\Omega : \mathbf{u}(\mathbf{x}).\mathbf{n}(\mathbf{x}) < 0\}, \tag{4.4}$$

where Ω has outward pointing normal vector $\mathbf{n}$.

With that definition of $\partial\Omega^-$, we can now introduce the following space for the configuration field:

$$Q = \left\{\mathbf{q} : \mathbf{q}\in[L^2(\Omega)]^2,\ \mathbf{u}.\nabla\mathbf{q}\in[L^2(\Omega)]^2\ \forall\mathbf{u}\in V\text{and } \mathbf{q}=\mathbf{q}_{\text{inflow}} \text{ on } \partial\Omega^-\right\}. \tag{4.5}$$

Equipped with these functional spaces, we introduce the following bilinear forms: $A, M : V\times V \longrightarrow \mathbb{R}$, $B : P\times V \longrightarrow \mathbb{R}$, $C : \Sigma\times V \longrightarrow \mathbb{R}$; thus

$$M(\mathbf{u},\mathbf{v}) = \frac{\rho}{\Delta t}\int_\Omega \mathbf{u}.\mathbf{v}d\mathbf{x} \quad \forall\mathbf{u},\mathbf{v}\in V, \tag{4.6}$$

$$A(\mathbf{u},\mathbf{v}) = \eta_s\int_\Omega \boldsymbol{\nabla}\mathbf{u}^T : \boldsymbol{\nabla}\mathbf{v}d\mathbf{x} \quad \forall\mathbf{u},\mathbf{v}\in V, \tag{4.7}$$

$$B(p,\mathbf{v}) = \int_\Omega p\boldsymbol{\nabla}.\mathbf{v}d\mathbf{x} \quad \forall p \in P,\ \mathbf{v} \in V, \tag{4.8}$$

$$C(\mathbf{T},\mathbf{v}) = \int_\Omega \mathbf{T} : \boldsymbol{\nabla}\mathbf{v}d\mathbf{x} \quad \forall \mathbf{T} \in \Sigma,\ \mathbf{v} \in V. \tag{4.9}$$

Using an implicit Euler scheme for the time discretization and the extra-stress being computed at the previous time step, the weak formulation of the Stokes problem may be formally written as follows: find $(\mathbf{u}(t_i), p(t_i)) \in V \times P$ for any given extra-stress $\boldsymbol{\tau}(t_{i-1}) \in \Sigma$ such that

$$\begin{aligned} &M(\mathbf{u}(t_i),\mathbf{v}) + A(\mathbf{u}(t_i),\mathbf{v}) - B(p(t_i),\mathbf{v}) \\ &\quad = M(\mathbf{u}(t_{i-1}),\mathbf{v}) - C(\boldsymbol{\tau}(t_{i-1}),\mathbf{v}) \quad \forall \mathbf{v} \in V, \end{aligned} \tag{4.10}$$

$$B(w,\mathbf{u}(t_i)) = 0 \quad \forall w \in P. \tag{4.11}$$

We now define one trilinear and one bilinear form $E : V \times Q \times Q \longrightarrow \mathbb{R}$ and $b : Q \times Q \longrightarrow \mathbb{R}$ to be used for the configuration field equation

$$E(\mathbf{u},\mathbf{q},\mathbf{S}) = \int_\Omega \left(\mathbf{q} + \left(\mathbf{u}.\nabla\mathbf{q} - \nabla\mathbf{u}.\mathbf{q} + \frac{\mathbf{q}}{2\lambda}\right)\Delta t\right).\mathbf{S}d\mathbf{x} \quad \forall \mathbf{u} \in V,\ \mathbf{q},\mathbf{S} \in Q, \tag{4.12}$$

$$b(\mathbf{b},\mathbf{S}) = \int_\Omega \mathbf{b}.\mathbf{S}d\mathbf{x} \quad \forall \mathbf{b},\mathbf{S} \in Q. \tag{4.13}$$

With these definitions of $E(\cdot,\cdot,\cdot)$ and $b(\cdot,\cdot)$, the Galerkin weak formulation of the configuration field equation becomes the following: find $\{\mathbf{q}_j^{t_{i-l}}(t_i)\}_{1\le j\le d}^{0\le l\le i-1} \in Q$ such that

$$E(\mathbf{u}(t_i), \mathbf{q}_j^{t_{i-l}}(t_i), \mathbf{S}) = b(\mathbf{q}_j^{t_{i-l}}(t_{i-1}), \mathbf{S}) \quad \forall \mathbf{S} \in \Sigma. \tag{4.14}$$

Note that on the right-hand side of (4.14) for $l = 0$, we have used the convention that $\mathbf{q}_j^{t_i}(t_{i-1}) = \mathbf{b}_j$, where $\mathbf{b}_j$ has been introduced in section 3. Once the vectors $\mathbf{q}_j^{t_{i-l}}(t_i)$ have been computed, the extra-stress $\boldsymbol{\tau}(t_i)$ can be easily determinated with the relation (3.21). It should be noted that the choice of spaces V, P, Σ, and Q are chosen to ensure that the integrals in the Galerkin weak formulation are bounded. However, the function spaces required for establishing the well-posedness of the problem may well have to be smaller than V, P, Σ, and Q.

4.2. Spectral element discretization. The Legendre spectral element method [14, 17] may be used for the discretization of the continuous problem (4.10), (4.11), and (4.14). The domain Ω is partitioned into K (say) nonoverlapping spectral elements $\{\Omega_k\}_{1\le k\le K}$. Then, letting $\mathcal{P}_N$ denote the space of polynomials of degree less than or equal to N in both Cartesian directions x and y within each spectral element Ω_k, our choice of finite-dimensional subspaces $V^N \subset V$, $P^N \subset P$, $Q^N \subset Q$, and $\Sigma^N \subset \Sigma$ for the velocity, pressure, configuration field, and extra-stress, respectively, are

$$\begin{aligned} V^N &= V \cap (\mathcal{P}_N)^2, \\ P^N &= P \cap \mathcal{P}_{N-2}, \\ Q^N &= Q \cap (\mathcal{P}_N)^2, \\ \Sigma^N &= \Sigma \cap (\mathcal{P}_N)^{2\times 2}. \end{aligned}$$

Note that the pressure approximation space was thus chosen in order to satisfy the Babuška–Brezzi condition for the velocity/pressure compatibility. However, due to the complexity of (4.10), (4.11), and (4.14), and, in particular, their nonlinearity, no available analysis indicating appropriate choices of compatible approximation spaces presently exists.

Mapping of each of the spectral elements $\{\Omega_k\}_{k=1}^K$ onto a parent element,

$$\widehat{\Omega} = \{(\xi, \eta) : -1 \le \xi, \eta \le 1\},$$

is achieved using the transfinite mapping technique of Gordon and Hall [11]. We may then write down discrete representations $\mathbf{u}_N(t_i)$ and $\mathbf{q}_N(t_i)$ for the velocity vector and configuration field at time t_i, respectively, as follows:

$$\mathbf{u}_N(t_i)|_{\Omega_k} \equiv \mathbf{u}_N^k(t_i, \xi, \eta) = \sum_{l=1}^{N+1} \sum_{m=1}^{N+1} \mathbf{u}(t_i)_{l,m}^k h_l(\xi) h_m(\eta) \in V_N, \tag{4.15}$$

$$\mathbf{q}_N(t_i)|_{\Omega_k} \equiv \mathbf{q}_N^k(t_i, \xi, \eta) = \sum_{l=1}^{N+1} \sum_{m=1}^{N+1} \mathbf{q}(t_i)_{l,m}^k h_l(\xi) h_m(\eta) \in Q_N. \tag{4.16}$$

In (4.15) and (4.16), $h_l(\xi)$, $1 \le l \le N+1$, denotes the lth degree N Lagrange interpolating polynomial, having the property that

$$h_l(\xi_m) = \delta_{lm}, \tag{4.17}$$

where ξ_m is the mth Gauss–Lobatto–Legendre point. The spectral representation of the pressure in Ω_k is taken as

$$p_N^k(t_i, \xi, \eta) = \sum_{l=1}^{N-1} \sum_{m=1}^{N-1} p(t_i)_{l,m}^k \tilde{h}_l(\xi) \tilde{h}_m(\eta), \tag{4.18}$$

where $\tilde{h}_l(\xi)$, $1 \le l \le N-1$, is the lth Lagrange interpolating polynomial of degree $N-2$ based on the interior Gauss–Lobatto–Legendre points. The integrals appearing in (4.10), (4.11), and (4.14) are determined numerically using Gauss–Lobatto quadrature rules, and the discretized equations can now be written in the following matrix-vector product form:

$$(\mathbf{M} + \mathbf{A})\mathbf{u}(t_i) - \mathbf{B}\mathbf{p}(t_i) = -\mathbf{C}\boldsymbol{\tau}(t_{i-1}) + \mathbf{M}\mathbf{u}(t_{i-1}), \tag{4.19}$$

$$\mathbf{B}^T \mathbf{u}(t_i) = \mathbf{0}, \tag{4.20}$$

$$\mathbf{E}_{\mathbf{u}(t_i)} \mathbf{q}_j^{t_{i-l}}(t_i) = \mathbf{q}_j^{t_{i-l}}(t_{i-1}) \text{ for } l = 0, \dots, i-1 \text{ and } j = 1, 2, \tag{4.21}$$

where $\mathbf{u}, \mathbf{p}$, and $\mathbf{q}_j^{t_{i-l}}$ in (4.19)–(4.21) are vectors containing, in an obvious way, the nodal values of the velocity, pressure, and configuration field variables. Once the vectors $\mathbf{q}_j^{t_{i-l}}$ have been computed with (4.21), the extra-stress to be used into (4.19) is determined with the formula (3.16) of Theorem 3.2. For a given velocity field $\mathbf{u}(t_i)$, the configuration field equation (4.21) constitutes a hyperbolic system of first-order partial differential equations. The Galerkin spectral element method may produce highly oscillatory numerical solutions for such equations. To treat this problem, we use the SUPG element-by-element method which was found to be an efficient method for the discretization of a steady state Oldroyd B flow [5]. The main idea of that method is to order the spectral elements according to their location along the streamlines. Then,

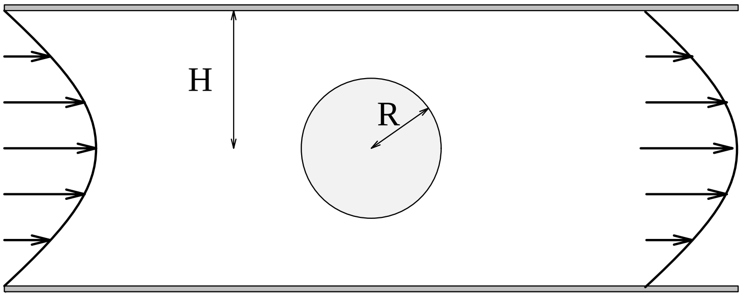


Fig. 2. *Cylinder radius R placed symmetrically in a 2D channel of half width H.*

the configuration field equation is solved sequentially on each spectral element, which makes possible the use of a direct solver for (4.21). The inflow boundary conditions are obtained from the upstream elements or the inflow boundary conditions. Note that the matrix $\mathbf{E}_{\mathbf{u}(t_i)}$ needs to be LU decomposed only once every time step since it is independent of $\mathbf{q}_j^{t_{i-l}}$.

5. Numerical example: Flow around a cylinder in a channel.

5.1. Description of the benchmark problem. The problem of planar viscoelastic flow around a cylinder has become popular recently as evidenced by the number of publications dealing with this benchmark problem over the last three years (e.g., [7, 10, 18, 19]). To be able to compare our results with those of others, we choose the aspect ratio $\Lambda = R/H = 1/2$, where H is the width of the channel and R is the radius of the cylinder (see Figure 2). A global Deborah number for this problem may be defined by

$$De = \frac{\lambda \overline{U}}{R}, \tag{5.1}$$

where $\overline{U}$ is the average velocity of the fluid in the channel at entry. The computational region extends a distance $20R$ upstream and downstream of the cylinder so that the assumption of fully developed flow conditions at entry and exit is valid. The ratio of the solvent viscosity η_s to the total viscosity $\eta = (\eta_s + \eta_p)$ was taken equal to 0.59 so as to match that used by other authors. The most popular quantity used for the comparison of numerical results is the drag factor F^* on the cylinder

$$F^* = \frac{F}{4\pi\eta\overline{U}},$$

where F is the drag on the cylinder

$$F = \int_0^\pi \left\{ \left(-p + 2\eta_s \frac{\partial u_x}{\partial x} + \tau_{xx} \right) \cos\theta + \left(\eta_s \left(\frac{\partial u_y}{\partial x} + \frac{\partial u_x}{\partial y} \right) + \tau_{xy} \right) \sin\theta \right\} R d\theta.$$

The problem is solved by dividing the flow domain into 22 conforming spectral elements (see Figure 3), and polynomial degrees ranging from $N = 8$ to $N = 14$ are used in the two spatial directions. The time steps are chosen equal to $\Delta t = 0.05$.

5.2. Numerical results. Two types of calculations are performed: the first one uses the constitutive equation (2.3), and the second one uses the new mesoscopic

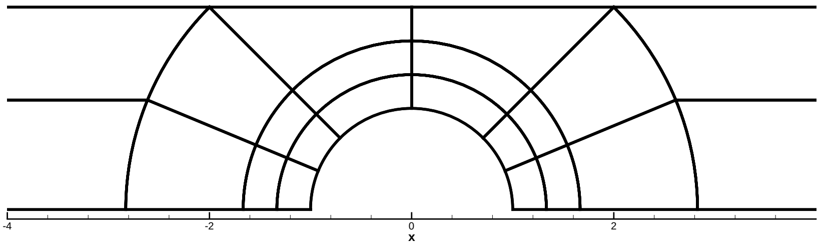


FIG. 3. *Flow domain divided into* 22 *spectral elements.*

approach described in section 3. For the two cases, the parameters $(De, \eta_s, \eta_p, \Delta t)$ and the numerical scheme (spectral element method with an implicit Euler scheme for the time discretization) are the same. The iterations are stopped when the following convergence criterion is fulfilled for all collocation points $\mathbf{x} \in \Omega$:

$$\frac{\|\mathbf{u}(\mathbf{x}, t_{i+1}) - \mathbf{u}(\mathbf{x}, t_i)\|}{\Delta t} \leq 10^{-3}.$$

At time t_i, the proposed mesoscopic method requires $d \times i$ vectors $\mathbf{q}_j^{t_{i-l}}(t_i)$ to be computed with (3.12) and (3.14). The total number of solves from $t = t_0$ to $t = t_i$ is therefore $\sum_{k=0}^{i} d \times k = d \times i \times (i+1)/2$. In practice, a maximum of $200 = \mathcal{O}(10^2)$ time steps was found to be sufficient to reach the steady solution. In this case the total number of solves is $\mathcal{O}(10^4)$. Since the computation of the extra-stress is deterministic, the solution is noise-free. On the other hand, for the traditional configuration field method, it is usual to take $3000 = \mathcal{O}(10^3)$ configuration fields. Therefore, we would have to solve equation (2.9) $\mathcal{O}(10^2) \times \mathcal{O}(10^3) = \mathcal{O}(10^5)$ times from $t = t_0$ to $t = t_i$. However, in this case, inevitably, the solution would be subject to noise. The present approach is therefore very competitive in terms of cost and quality of the solution.

Tables 1 and 2 give the values of the drag factor for two types of computations (the first one uses the formulation presented in this paper, and the second uses the classical Oldroyd-B constitutive equation) for different level of discretizations ($N = 8, \dots, 14$) and different Deborah numbers. The mesoscopic calculation is clearly more robust since Deborah numbers up to $De = 0.8$ can be reached, whereas the constitutive equation allows computation to proceed only up to $De = 0.6$. The convergence of the drag factor with mesh refinement is also better: the difference between the two finest meshes at $De = 0.6$ is 0.004% for the mesoscopic computation and falls to 0.2% for its macroscopic counterpart. The superiority of the method can be seen clearly by comparing Figures 4 and 5 which show the profiles of τ_{xx} for three levels of discretization ($N = 8, 10$, and 12) at the same Deborah number $De = 0.6$ for the macroscopic and mesoscopic computation, respectively. For high Deborah numbers, and as already pointed out in numerous papers [4, 8, 9, 10], the drag factor is not a good indicator of the quality of the solution because the critical part of the flow lies in the wake of the cylinder where the τ_{xx} component of the extra-stress is exceedingly difficult to capture. This is why, although there are published results for the drag factors up to $De = 1.85$ [19], evidence of convergence in the wake of the cylinder is available only up to $De = 0.7$ [5, 10]. Figures 6 and 7 show the profiles of τ_{xx} for three level of discretization ($N = 8, 10$, and 12) for the mesoscopic computation at $De = 0.7$ and $De = 0.8$, respectively. The convergence with mesh refinement is still

Table 1
Drag factor F^* computed on uniform meshes ($N = 8, \ldots, 14$). Mesoscopic simulation.

De	**N=8**	**N=9**	**N=10**	**N=11**	**N=12**	**N=13**	**N=14**
0.3	9.7735	9.7741	9.7740	9.7739	9.7740	9.7739	9.7739
0.4	9.5623	9.5655	9.5655	9.5660	9.5658	9.5655	9.5656
0.5	9.4136	9.4178	9.4200	9.4226	9.4213	9.4207	9.4210
0.6	9.3226	9.3189	9.3293	9.3341	9.3317	9.3317	9.3313
0.7	9.2904	9.2591	9.2886	9.2882	9.2895	9.2898	-
0.8	9.3328	9.2352	9.3037	9.2743	9.2958	-	-

Table 2
Drag factor F^* computed on uniform meshes ($N = 8, \ldots, 14$). Macroscopic simulation.

De	**N=8**	**N=9**	**N=10**	**N=11**	**N=12**	**N=13**	**N=14**
0.3	9.8022	9.8034	9.8030	9.8029	9.8032	9.8031	9.8032
0.4	9.5897	9.5969	9.5953	9.5969	9.5960	9.5952	9.5963
0.5	9.4317	9.4428	9.4510	9.4633	9.4540	9.4547	9.4551
0.6	9.2801	9.2872	9.3450	9.3807	9.3591	9.3803	9.3611

good both on the cylinder surface ($-1 \leq x \leq 1$) and in the wake of the cylinder ($x > 1$), even at the highest Deborah number.

6. Extension of the method to multimode Oldroyd B fluids.

In practice, fluids have a continuous spectrum of relaxation times. This is approximated by taking several (say p) relaxation times $\{\lambda_r\}_{1\leq r\leq p}$ associated with polymeric viscosities $\{\eta_{pr}\}_{1\leq r\leq p}$. The extra-stress that now enters the linear momentum equation is the sum of the contribution of each mode (λ_r, η_{pr}), i.e., for a p-mode model

$$\boldsymbol{\tau}(\mathbf{x},t_i) = \sum_{r=1}^{p} \boldsymbol{\tau}_r(\mathbf{x},t_i). \tag{6.1}$$

For the Oldroyd B multimode model, the equation satisfied by the rth mode of the configuration field $\mathbf{Q}_m^r(\mathbf{x}, t_i)$ is

$$\begin{aligned}\mathbf{Q}_m^r(\mathbf{x}, t_i) + \Big(\mathbf{u}(\mathbf{x},t_i).\nabla\mathbf{Q}_m^r(\mathbf{x}, t_i) - \nabla\mathbf{u}(\mathbf{x},t_i)\mathbf{Q}_m^r(\mathbf{x}, t_i)\Big)\Delta t \\ = \left(1 - \frac{\Delta t}{2\lambda_r}\right)\mathbf{Q}_m^r(\mathbf{x}, t_{i-1}) + \sqrt{\frac{\Delta t}{\lambda_r}}\Delta\mathbf{W}_m^r(t_i),\end{aligned} \tag{6.2}$$

where all the terms depending on λ_r have been taken explicitly so that the matrix $\mathbf{E_u}(\mathbf{x}, t_i)$ resulting from the discretization of the left-hand side is now independent of the modes and needs to be LU decomposed only once every time step.

Introducing a real function of the relaxation time λ for any given time step Δt thus

$$\beta(\lambda) = \left(1 - \frac{\Delta t}{2\lambda}\right), \tag{6.3}$$

the following result can be shown by induction on i in the same way as Theorem 3.1.

Theorem 6.1. *Let $\mathbf{Q}_m^r(\mathbf{x}, t_i)$ be the rth mode discrete solution of (6.2) at time $t_i = i\Delta t$ ($i \geq 1$) for any given Wiener process $\Delta\mathbf{W}_m^r(t_i)$. Then, there exists $d \times i$ vectors independent of m and independent of the mode r, noted $\{\mathbf{q}_j^{t_{i-l}}(\mathbf{x}, t_i)\}_{0\leq l\leq i-1}^{1\leq j\leq d}$*

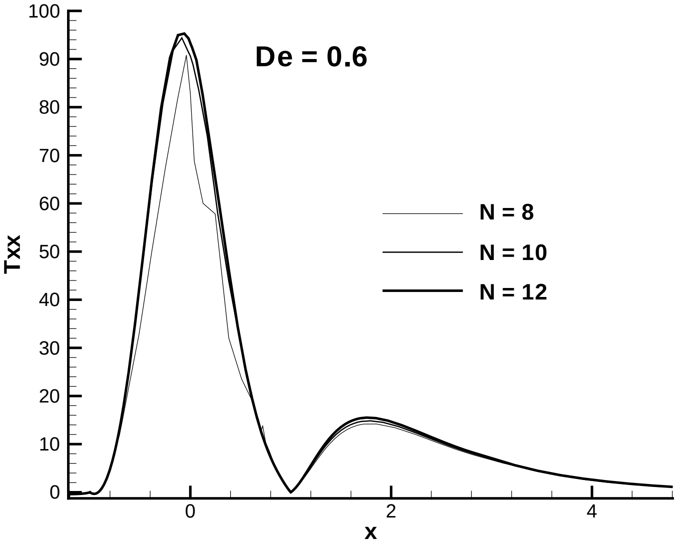


FIG. 4. *Profiles of τ_{xx} on the cylinder surface and in the wake of the cylinder for different meshes at $De = 0.6$. Macroscopic simulation.*

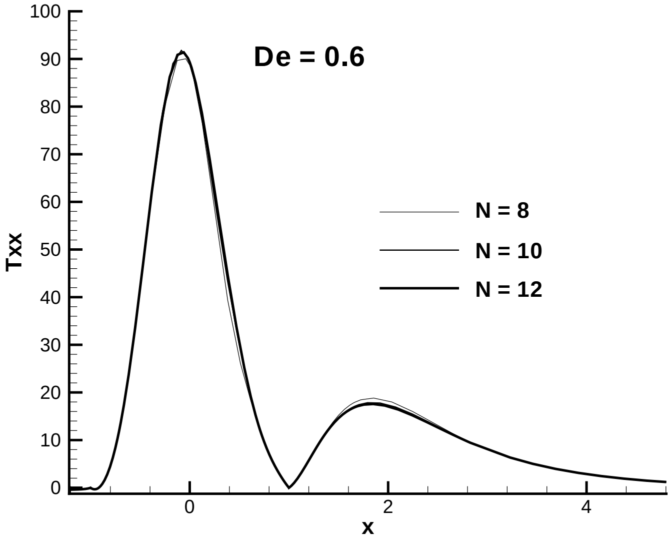


FIG. 5. *Profiles of τ_{xx} on the cylinder surface and in the wake of the cylinder for different meshes at $De = 0.6$. Mesoscopic simulation.*

such that

$$\mathbf{Q}^r_m(\mathbf{x},t_i)=\sum_{j=1}^{d}\left\{\left(\sum_{l=0}^{i-1}\alpha(\lambda_r)\beta(\lambda_r)^l\mathbf{q}_j^{t_{i-l}}(\mathbf{x},t_i)\Delta W_m^{rj}(t_{i-l})\right)+\beta(\lambda_r)^i\mathbf{q}_j^{t_1}(\mathbf{x},t_i)\Delta W_m^{rj}(t_0)\right\}.$$

The vectors $\mathbf{q}_j^{t_{i-l}}(\mathbf{x},t_i)$ are created and updated similarly to the single mode case. In the same fashion as in Theorem 3.2, we can derive a simple expression for the rth mode contribution to the extra-stress

$$\begin{aligned}(6.4)\quad \boldsymbol{\tau}_r(\mathbf{x},t_i) = \frac{\eta_{pr}}{\lambda_r}\Bigg(-\mathbf{I}+\sum_{j=1}^{d}\Bigg\{\sum_{l=0}^{i-1}\Big(\alpha(\lambda_r)^2\beta(\lambda_r)^{2l}\mathbf{q}_j^{t_{i-l}}(\mathbf{x},t_i)\otimes\mathbf{q}_j^{t_{i-l}}(\mathbf{x},t_i)\Big)\\ +\beta(\lambda_r)^{2i}\mathbf{q}_j^{t_1}(\mathbf{x},t_i)\otimes\mathbf{q}_j^{t_1}(\mathbf{x},t_i)\Bigg\}\Bigg).\end{aligned}$$

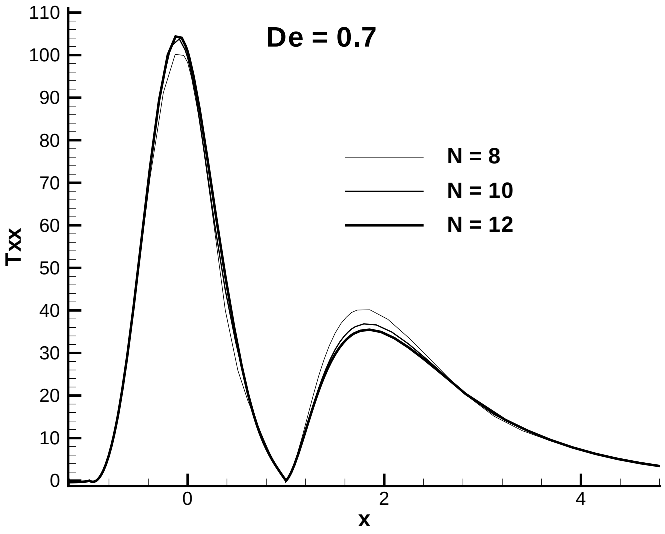


FIG. 6. *Profiles of τ_{xx} on the cylinder surface and in the wake of the cylinder for different meshes at $De = 0.7$. Mesoscopic simulation.*

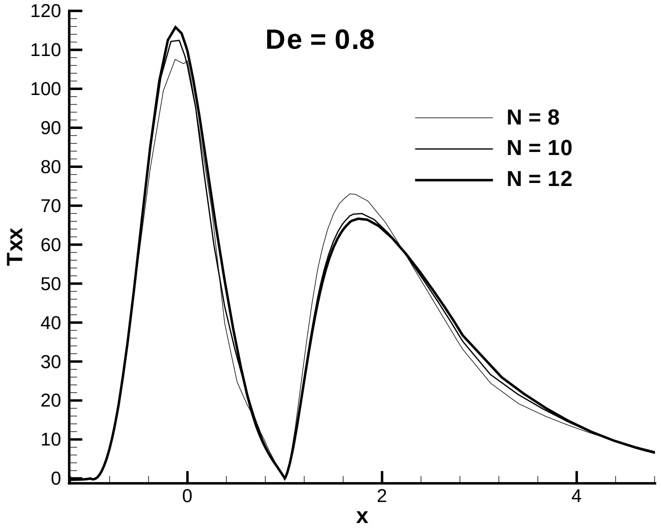


FIG. 7. *Profiles of τ_{xx} on the cylinder surface and in the wake of the cylinder for different meshes at $De = 0.8$. Mesoscopic simulation.*

In order to write the total extra-stress in a condensed way, we define i constants $\{\chi_l\}_{0\leq l\leq i-1}$ at time t_i; thus

$$\chi_l = \sum_{r=1}^{p} \frac{\eta_{pr}}{\lambda_r}\alpha(\lambda_r)^2\beta(\lambda_r)^{2l} \text{ for } l = 0, \dots, i-2, \tag{6.5}$$

$$\chi_{i-1} = \sum_{r=1}^{p} \frac{\eta_{pr}}{\lambda_r}\left(\alpha(\lambda_r)^2\beta(\lambda_r)^{2(i-1)} + \beta(\lambda_r)^{2i}\right). \tag{6.6}$$

With those definitions and the help of (6.1) and (6.4), the total extra-stress can be written

$$\boldsymbol{\tau}(\mathbf{x},t_i) = -\left(\sum_{r=1}^{p} \frac{\eta_{pr}}{\lambda_r}\right)\mathbf{I} + \sum_{j=1}^{d}\sum_{l=0}^{i-1} \chi_l \mathbf{q}_j^{t_{i-l}}(\mathbf{x},t_i) \otimes \mathbf{q}_j^{t_{i-l}}(\mathbf{x},t_i). \tag{6.7}$$

The equation above shows that the difference between the single mode Oldroyd B model and the multimode model lies in the computation of $\{\chi_l\}_{0\leq l\leq i-1}$, which is an inexpensive calculation. This is a big advantage over simulations using constitutive equations, where an equation similar to (2.3) has to be solved for each mode.

7. Conclusion. The principal idea in the proposed method lies in the fact that the random part appearing in the discretized stochastic equation follows the normal law, which enables a simple expression for the extra-stress to be derived for the Oldroyd B model. The multimode case was shown to be a simple extension to the presented approach. The method has been tested numerically on the benchmark problem of an Oldroyd B fluid past a cylinder in a channel. The results have shown that the method is superior in terms of stability to using a constitutive equation. It is also cheaper than the original Brownian configuration field method. The present approach was extended to dumbbell models having finitely extensible elastic spring forces such as the FENE-P model in [6].

Acknowledgment. The author would like to thank Robert Owens for helpful discussions and for proofreading the manuscript.

REFERENCES

[1] R.B. Bird, R.C. Armstrong, and O. Hassager, *Dynamics of Polymeric Liquids*, Vol. 2, John Wiley and Sons, New York, 1987.

[2] J. Bonvin, *Numerical Simulation of Viscoelastic Fluids With Mesoscopic Models*, Ph.D. thesis, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, 2000.

[3] J. Bonvin and M. Picasso, *Variance reduction methods for CONNFFESSIT-like simulations*, J. Non-Newtonian Fluid Mech., 84 (1999), pp. 191–215.

[4] C. Chauviere and R.G. Owens, *How accurate is your solution? Error indicators for viscoelastic flow calculations*, J. Non-Newtonian Fluid Mech., 95 (2000), pp. 1–33.

[5] C. Chauviere and R.G. Owens, *A new spectral element method for the reliable computation of viscoelastic flow*, Comput. Meth. Appl. Mech. Engrg., 190 (2001), pp. 3999–4018.

[6] C. Chauvière and R.G. Owens, *A robust spectral element method for simulations of time-dependent viscoelastic flows, derived from the Brownian configuration field method*, J. Sci. Comput., to appear.

[7] H.S. Dou and N. Phan-Thien, *The flow of an Oldroyd-B fluid past a cylinder in a channel: Adaptive viscosity vorticity (DAVSS-ω) formulation*, J. Non-Newtonian Fluid Mech., 87 (1999), pp. 47–73.

[8] Y.R. Fan, *A comparative study of the discontinuous Galerkin and continuous SUPG finite element methods for computation of viscoelastic flows*, Comput. Meth. Appl. Mech. Engrg., 141 (1997), pp. 47–65.

[9] Y.R. Fan and M.J. Crochet, *High-order finite element methods for steady viscoelastic flows*, J. Non-Newtonian Fluid Mech., 57 (1995), pp. 283–311.

[10] Y.R. Fan, R.I. Tanner, and N. Phan-Thien, *Galerkin/least-squares finite-element methods for steady viscoelastic flows*, J. Non-Newtonian Fluid Mech., 84 (1999), pp. 233–256.

[11] W.J. Gordon and C.A. Hall, *Transfinite element methods: Blending function interpolation over arbitrary curved element domains*, Numer. Math., 21 (1973), pp. 109–129.

[12] M.A. Hulsen, A.P.G. Van Heel, and B.H.A.A. Van Den Brule, *Simulation of viscoelastic flows using Brownian configuration fields*, J. Non-Newtonian Fluid Mech., 70 (1997), pp. 79–101.

[13] M. Laso and H.C. Öttinger, *Calculation of viscoelastic flow using molecular models: The CONNFFESSIT approach*, J. Non-Newtonian Fluid Mech., 47 (1993), pp. 1–20.

[14] Y. Maday and A.T. Patera, *Spectral element methods for the incompressible Navier-Stokes equations*, in State of the Art Surveys in Computational Mechanics, ASME, New York, 1989, pp. 71–143.

[15] H.C. Öttinger, B.H.A.A. Van Den Brule, and M.A. Hulsen, *Brownian configuration fields and variance reduced CONNFFESSIT*, J. Non-Newtonian Fluid Mech., 70 (1997), pp. 255–261.

[16] H.C. Öttinger, *Stochastic Processes in Polymeric Fluids*, Springer-Verlag, Berlin, 1996.

[17] A.T. Patera, *A spectral element method for fluid dynamics: Laminar flow in a channel expansion*, J. Comput. Phys., 54 (1984), pp. 468–488.

[18] N. Phan-Thien and H.S. Dou, *Viscoelastic flow past a cylinder: Drag coefficient*, Comput. Meth. Appl. Mech. Engrg., 180 (1999), pp. 243–266.

[19] J. Sun, M.D. Smith, R.C. Armstrong, and R.A. Brown, *Finite element method for viscoelastic flows based on the discrete adaptive viscoelastic stress splitting and the discontinuous Galerkin method: DAVSS-G/DG*, J. Non-Newtonian Fluid Mech., 86 (1999), pp. 281–307.

[20] A.P.G. Van Heel, M.A. Hulsen, and B.H.A.A. Van Den Brule, *On the selection of parameters in the FENE-P model*, J. Non-Newtonian Fluid Mech., 75 (1998), pp. 253–271.

SIAM J. Sci. Comput.
Vol. 23, No. 6, pp. 2141–2164

MULTILEVEL METHOD FOR MIXED EIGENPROBLEMS[*]

R. HIPTMAIR[†] AND K. NEYMEYR[‡]

Abstract. For a Lipschitz-polyhedron $\Omega \subset \mathbb{R}^3$ we consider eigenvalue problems $\mathbf{curl}\,\alpha\,\mathbf{curl}\,\mathbf{u} = \lambda\mathbf{u}$ and $\operatorname{grad}\alpha\operatorname{div}\mathbf{u} = \lambda\mathbf{u}$, $\lambda > 0$, set in $\mathsf{H}(\mathbf{curl};\Omega)$ and $\mathsf{H}(\operatorname{div};\Omega)$. They are discretized by means of the conforming finite elements introduced by Nédélec. The preconditioned inverse iteration in its subspace variant is adapted to these problems. A standard multigrid scheme serves as the preconditioner. The main challenge arises from the large kernels of the operators **curl** and div. However, thanks to the choice of finite element spaces these kernels have a direct representation through the gradients/rotations of discrete potentials. This makes it possible to use a multigrid iteration in potential space to obtain approximate projections onto the orthogonal complements of the kernels. There is ample evidence that this will lead to an asymptotically optimal method. Numerical experiments confirm the excellent performance of the method even on very fine grids.

Key words. mixed eigenvalue problems, edge elements, Raviart–Thomas elements, mixed finite elements, preconditioned inverse iteration, multigrid methods

AMS subject classifications. 65F15, 65N25, 65N30, 78A40

PII. S1064827501385001

1. Introduction.

Let $\Omega \subset \mathbb{R}^3$ be a Lipschitz-polyhedron [36], whose boundary is partitioned into Γ_D and Γ_N. Our focus is on the vector-valued eigenvalue problems

$$\begin{array}{rclll} \mathbf{curl}\,\alpha\,\mathbf{curl}\,\mathbf{u} &=& \lambda\mathbf{u} & \text{in } \Omega\,, & \mathbf{u}\times\mathbf{n} = 0 \text{ on } \Gamma_D\,, \\ \operatorname{div}\mathbf{u} &=& 0 & \text{in } \Omega\,, & \alpha\,\mathbf{curl}\,\mathbf{u}\times\mathbf{n} = 0 \text{ on } \Gamma_N\,, \end{array} \tag{1.1}$$

and

$$\begin{array}{rclll} \operatorname{grad}\alpha\operatorname{div}\mathbf{u} &=& \lambda\mathbf{u} & \text{in } \Omega\,, & \mathbf{u}\cdot\mathbf{n} = 0 \text{ on } \Gamma_D\,, \\ \mathbf{curl}\,\mathbf{u} &=& 0 & \text{in } \Omega\,, & \alpha\operatorname{div}\mathbf{u}\cdot\mathbf{n} = 0 \text{ on } \Gamma_N\,. \end{array} \tag{1.2}$$

Here, the vectorfields $\mathbf{u}$ is an eigenfunction, $\lambda \geq 0$ stands for the eigenvalue, and $\alpha \in L^\infty(\Omega)$ is a uniformly positive coefficient.

We seek approximations of a few of the smallest nonzero eigenvalues and corresponding eigenfunctions. This problem is of considerable relevance in several areas of scientific computing. For instance, (1.1) describes so-called electromagnetic resonators if $\mathbf{u}$ is regarded as the (scaled) electric field. We refer to [1, sect. 1] for more detailed explanations. When we want to determine a few of the lowest resonant modes for a given cavity Ω, we exactly encounter the eigenvalue problem (1.1). Beyond the calculation of resonant modes, approximations of the lowest eigenmodes are the basis for modal [28] approaches: A set of dominant modes is computed once, and the fields at other frequencies are then approximated by a superposition of these modes. This can be used to extract lumped parameters for electromagnetic devices in the frequency domain. A completely different application emerges in the study of coupled

[*] Received by the editors February 14, 2001; accepted for publication (in revised form) November 2, 2001; published electronically May 15, 2002.

http://www.siam.org/journals/sisc/23-6/38500.html

[†] SFB 382, Universität Tübingen, Auf der Morgenstelle 10, D-72076 Tübingen, Germany (ralf@hiptmair.de). The work of this author was supported by Deutsche Forschungsgemeinschaft as part of SFB 382.

[‡] Mathematisches Institut, Universität Tübingen, Auf der Morgenstelle 10, D-72076 Tübingen, Germany (klaus@na.uni-tuebingen.de).

solid-fluid systems. When one tries to find their eigenmodes, the eigenvalue problem (1.2) appears [9].

Of course, there is a close relationship between (1.1) and (1.2) and eigenvalue problems for second order elliptic differential operators. For the latter case, which amounts to a generalized eigenvalue problem for large sparse symmetric *positive definite* matrices, a huge body of work about numerical solution methods has been compiled over the years [4, 5, 22, 38, 48, 54]. The driving force was the sheer size of the eigenproblems arising from discretized PDEs. Millions of unknowns rule out the use of methods that rely on dense matrices or factorizations. In addition, it is highly desirable to avoid a deterioration of the convergence of the iterative schemes for large problems. As far as the solution of discretized elliptic boundary value problems is concerned, multigrid methods meet this requirement. It turned out that the multigrid idea can be grafted onto solution methods for discrete elliptic eigenproblems in several ways, resulting in eigensolvers with optimal or quasi-optimal computational complexity. For instance, Hackbusch [38, 39] applies multigrid principles directly to the nonlinear eigenvalue problem to compute eigenvalue/vector approximations on the final grid by combining a multigrid iteration and nested iteration. Let us also mention the multigrid minimization technique of Mandel and McCormick [54], its extension by Deuflhard, Friese, and Schmidt [27], as well as the class of methods which apply multigrid as a linear solver. Essentially, the idea underlying this last class is to linearize the discrete eigenvalue problem by methods like inverse iteration [64] and to solve the associated system of linear equations approximately by multigrid [5]. Representing the application of the multigrid procedure by a multigrid preconditioner and taking inverse iteration (without a shift) as an outer iteration defines *preconditioned inverse iteration* (PINVIT). Recently, a new convergence theory for PINVIT has been devised, providing sharp convergence estimates and substantial insight into the underlying geometry [50, 61, 62].

The scheme of preconditioned inverse iteration is also known in the literature as the preconditioned gradient method for the eigenvalue problem. The idea behind this term is to compute a sequence of iterates with decreasing Rayleigh quotients by successively correcting the iterates in the direction of the negative preconditioned gradient of the Rayleigh quotient. By doing so, one expects that the sequence of iterates converges to an eigenvector while the Rayleigh quotients tend to the smallest eigenvalue. Preconditioned gradient methods have been studied predominantly by Russian authors: see, for instance, Samokish [66], Petryshyn [65], Godunov, Ogneva, and Prokopov [35], D'yakonov [29], D'yakonov and Orekhov [31], and Knyazev [47,48], as well as the monograph of D'yakonov [30], including an extensive bibliography. Recently, Ovtchinnikov and Xanthis [63] introduced a new variant. Knyazev in [48] gives a survey on preconditioned eigensolvers.

Preconditioned inverse iteration has been generalized to a subspace algorithm for computing some of the smallest eigenvalues together with the eigenvectors by emulating the subspace variant of inverse iteration [64]. Once again, the associated matrix equation is solved approximately. After each subspace correction step the Rayleigh–Ritz procedure is applied. It provides the Ritz values and Ritz vectors spanning the approximating subspace. Convergence estimates have been presented in [20, 59]. In sum, the resulting preconditioned eigensolver inherits the typical asymptotic multigrid efficiency [51] from the multigrid procedure used to solve the associated linear equations.

On a smaller scale, researchers have also investigated ways to compute solutions to (1.1) and (1.2) [1, 68]. It is obvious that the large kernels of the differential operators **curl** and div pose the main challenge: A straightforward application of iterative techniques developed for the symmetric positive definite case is doomed because these methods single out the smallest eigenvalues and will invariably churn out kernel vectors in the end. However, as $\lambda > 0$ is requested, these are not the desired answer. We are left with the task of steering the iterations away from the kernels.

One option is regularization, i.e., adding a term corresponding to a weak version of $\operatorname{grad}\operatorname{div}\cdot$ for (1.1) and $\mathbf{curl}\,\mathbf{curl}\cdot$ for (1.2) to the differential operator (cf. [1, sect. 4.1] and [10]). This will make the kernel "visible" and convert the problem into a standard positive definite one. Thus it becomes amenable to "shift-and-invert" techniques combined with, e.g., an implicitly restarted Lanczos method. The resulting indefinite linear systems of equations can be solved by means of Krylov-subspace methods, whose convergence will degrade, however, for very large problems.

An alternative option is projection of approximate eigenvectors onto a complement of the kernels. This is the gist of our method, which we call *projected preconditioned inverse iteration* (PPINVIT). The idea to forgo regularization in favor of projections is fairly natural. For instance, it is used in [67] for two-dimensional problems arising in waveguide design. Yet, little is gained unless a fast projection and good preconditioners are at our disposal.

Recently, multilevel methods for the solution of $\boldsymbol{H}(\mathbf{curl};\Omega)$- and $\boldsymbol{H}(\operatorname{div};\Omega)$-elliptic boundary value problems have become available [3, 41, 43] if discretization is based on special conforming finite elements. The goal of this paper is to demonstrate how they can be forged into eigenproblem solvers featuring multigrid efficiency. The key idea is to combine the subspace variant of PINVIT [59] with an inexact multigrid projection onto the orthogonal complements of the kernels.

Since the approach crucially hinges on particular properties of the finite elements, those are reviewed in the next section. Then we give a detailed description of the algorithm, complete with projection control and termination criteria. The fourth section is dedicated to some theoretical investigations of the convergence of the method. Yet, we have not succeeded in providing a comprehensive theoretical analysis. To compensate for this, we report some numerical experiments in the final section. They give evidence of the efficacy and satisfactory performance of the method for some typical large eigenvalue problems.

2. Discrete eigenvalue problems.

The Galerkin-discretization starts from the weak form of the eigenvalue problems: In the case of (1.1) we seek $\mathbf{u} \in \boldsymbol{H}_{\Gamma_D}(\mathbf{curl};\Omega)$, $\lambda > 0$ such that

$$(\alpha\,\mathbf{curl}\,\mathbf{u}, \mathbf{curl}\,\mathbf{v})_0 = \lambda\,(\mathbf{u},\mathbf{v})_0 \quad \forall \mathbf{v} \in \boldsymbol{H}_{\Gamma_D}(\mathbf{curl};\Omega)\ . \tag{2.1}$$

If (1.2) is of concern the weak form reads: Seek $\mathbf{u} \in \boldsymbol{H}_{\Gamma_D}(\operatorname{div};\Omega)$, $\lambda > 0$ such that

$$(\alpha\operatorname{div}\mathbf{u}, \operatorname{div}\mathbf{v})_0 = \lambda\,(\mathbf{u},\mathbf{v})_0 \quad \forall \mathbf{v} \in \boldsymbol{H}_{\Gamma_D}(\operatorname{div};\Omega)\ . \tag{2.2}$$

As usual, we adopt the notation $(\cdot,\cdot)_0$ for the $\boldsymbol{L}^2(\Omega)$-inner product. In addition, a subscript Γ_D tags spaces of functions satisfying zero (tangential/normal) traces on Γ_D.

By testing (2.1) with gradients and (2.2) with **curl**s we observe that solutions $\mathbf{u}$ are either weakly divergence-free or weakly **curl**-free. As $\boldsymbol{H}_{\Gamma_D}(\mathbf{curl};\Omega) \cap \boldsymbol{H}(\operatorname{div};\Omega)$ and $\boldsymbol{H}_{\Gamma_D}(\operatorname{div};\Omega) \cap \boldsymbol{H}(\mathbf{curl};\Omega)$ are both compactly embedded in $\boldsymbol{L}^2(\Omega)$ [46] and the bilinear forms on the left-hand sides of (2.1) and (2.2) are symmetric positive semidefinite, the Riesz–Schauder theory guarantees the existence of increasing sequences of real positive eigenvalues $0 < \lambda_1 \leq \lambda_2 \leq \cdots$. From symmetry we can also conclude that the corresponding eigenspaces are $\boldsymbol{L}^2(\Omega)$-orthogonal.

This carries over to the discrete eigenfunctions obtained through a Galerkin-discretization of (2.1) and (2.2). In particular, we use conforming finite elements based on a hexahedral or simplicial triangulation $\mathcal{T}_h = \{T_i\}_i$ of Ω. Then, using the constructions proposed by Nédélec in [58], we obtain the finite element spaces $\boldsymbol{\mathcal{W}}^1_p(\mathcal{T}_h) \subset \boldsymbol{H}(\mathbf{curl};\Omega)$ and $\boldsymbol{\mathcal{W}}^2_p(\mathcal{T}_h) \subset \boldsymbol{H}(\mathrm{div};\Omega)$ of any polynomial order $p \in \mathbb{N}_0$. Details and descriptions of the degrees of freedom are given in, e.g., [21,34,56,58]. Dirichlet boundary conditions can be enforced by setting the degrees of freedom (d.o.f.) on Γ_D to zero.

In the case of lowest polynomial order, $p = 0$, the finite elements are either known as Whitney-forms [16] or, in the engineering literature, as *edge elements* ($\boldsymbol{H}(\mathbf{curl};\Omega)$-conforming scheme) and *face elements* ($\boldsymbol{H}(\mathrm{div};\Omega)$-conforming scheme), respectively. They owe these names to the definition of their d.o.f., which are given by path integrals along edges of the mesh and flux integrals over its faces, respectively. The finite element spaces form affine equivalent families if special transformations are used [42]. This makes it possible to show approximation properties on shape regular families of meshes (cf. [26]). In addition, if quasi uniformity is assumed, the inverse inequalities

$$\begin{aligned} \|\mathbf{curl}\,\mathbf{u}_h\|_0 &\leq C h^{-1}\,\|\mathbf{u}_h\|_0 \quad \forall \mathbf{u}_h \in \boldsymbol{\mathcal{W}}^1_p(\mathcal{T}_h)\ , \\ \|\mathrm{div}\,\mathbf{u}_h\|_0 &\leq C h^{-1}\,\|\mathbf{u}_h\|_0 \quad \forall \mathbf{u}_h \in \boldsymbol{\mathcal{W}}^2_p(\mathcal{T}_h) \end{aligned} \tag{2.3}$$

hold, where $h := \max\{\mathrm{diam}\,T,\, T \in \mathcal{T}_h\}$ is the meshwidth and $C > 0$ are generic constants. By this terminology we mean that C may depend *only* on $\Omega, \Gamma_D, \alpha, p$, and the shape regularity of the finite element mesh. On the other hand, the value of generic constants may change between different occurrences.

Despite the glaring differences in their definitions, the finite element spaces for $\boldsymbol{H}(\mathbf{curl};\Omega)$ and $\boldsymbol{H}(\mathrm{div};\Omega)$ introduced above are closely related. As discussed in [18, 19, 42], they all can be viewed as spaces of *discrete differential forms.* This is the rationale behind our decision to treat both (1.1) and (1.2) in a common framework. In a sense, we will adopt the common notation $\boldsymbol{\mathcal{V}}_h$ for both $\boldsymbol{\mathcal{W}}^1_p(\mathcal{T}_h)$ or $\boldsymbol{\mathcal{W}}^2_p(\mathcal{T}_h)$ with suitable Dirichlet boundary conditions imposed.

Hitherto, discrete differential forms supply the only conforming finite element discretization of (2.1) and (2.2) that can steer clear of so-called *spurious modes.* For instance, if one uses $H^1(\Omega)$-conforming finite elements to discretize the Cartesian components of the vectorfields $\mathbf{u}$, the discrete spectrum may feature eigenvalues that are not related to an eigenvalue of the continuous problem [14,17,33]. On the contrary, in recent years rigorous arguments have been found about why discrete differential forms ensure a correct approximation of the spectrum [12, 15, 23, 24, 33, 57]. For shape-regular families of meshes convergence of the eigenvalues will be quadratic in the meshwidth [23] under mild assumptions on the smoothness of the eigenfunctions.

A key role in the convergence theory is played by *discrete potentials.* They refer to an exceptional property of discrete differential forms, namely that they give rise to analogues to de Rham's exact sequences in a purely discrete setting [13, 19]. In particular, for contractible Ω, $\Gamma_D = \partial\Omega$ or $\Gamma_N = \partial\Omega$,

$$\{\mathbf{u}_h \in \boldsymbol{\mathcal{W}}^1_p(\mathcal{T}_h),\ \mathbf{curl}\,\mathbf{u}_h = 0\} = \mathrm{grad}\,\mathcal{W}^0_p(\mathcal{T}_h)\ , \tag{2.4}$$

$$\{\mathbf{u}_h \in \boldsymbol{\mathcal{W}}^2_p(\mathcal{T}_h),\ \mathrm{div}\,\mathbf{u}_h = 0\} = \mathbf{curl}\,\boldsymbol{\mathcal{W}}^1_p(\mathcal{T}_h)\ , \tag{2.5}$$

where $\mathcal{W}^0_p(\mathcal{T}_h)$ stands for the space of continuous finite element functions, piecewise polynomial of degree $p+1$ over $\mathcal{T}_h$, the conventional Lagrangian finite elements (see [25]). A proof of these identities can be found in [42]. Now it is clear why $\mathcal{W}_p^0(\mathcal{T}_h)$ and $\boldsymbol{\mathcal{W}}_p^1(\mathcal{T}_h)$ have been dubbed spaces of discrete potentials. Those will be denoted by $\mathcal{S}_h$, and $G_h : \mathcal{S}_h \mapsto \boldsymbol{\mathcal{V}}_h$ is the related differential operator mapping into the kernel of the respective differential operator; that is, $G_h := \text{grad}$ or $G_h := \mathbf{curl}$.

In the case of complex topologies and Dirichlet boundary conditions on parts of $\partial\Omega$, the kernels of the differential operators are no longer completely given by suitable discrete potentials. What is still missing are low-dimensional spaces of discrete harmonic vectorfields, $\boldsymbol{\mathcal{H}}^1(\mathcal{T}_h) \subset \boldsymbol{\mathcal{W}}_0^1(\mathcal{T}_h)$ and $\boldsymbol{\mathcal{H}}_2(\mathcal{T}_h) \subset \boldsymbol{\mathcal{W}}_0^2(\mathcal{T}_h)$, whose dimensions depend on the topology of Ω and the arrangements of the connected components of Γ_D. For instance, if $\Gamma_D = \partial\Omega$ the dimension of $\boldsymbol{\mathcal{H}}^1(\mathcal{T}_h)$ is equal to the number of connected components of $\partial\Omega$. A basis for $\boldsymbol{\mathcal{H}}^1(\mathcal{T}_h)$ is given by the gradients of piecewise linear continuous functions that assume the value 1 on one connected component of Γ_D and vanish on the other. Evidently, this basis can be constructed with little effort. In the case of Neumann boundary conditions throughout, $\dim \boldsymbol{\mathcal{H}}^1(\mathcal{T}_h)$ is equal to the number of homology classes of boundary cycles that are bounding relative to Ω. To find a basis, we associate a cutting surface to each homology class and compute the gradient of a piecewise linear function that is continuous except for a jump of height 1 across the cutting surface [2]. The surfaces can be determined by means of graph-theoretic algorithms [37]. In the case of mixed boundary conditions the situation is more involved [32, 52], but for concrete geometries the harmonic vectorfields can usually be found easily. In what follows we will write $\boldsymbol{\mathcal{H}}_h$ for a space of harmonic vectorfields and will take for granted that a basis $\{\mathbf{h}_1, \dots, \mathbf{h}_q\}$ of $\boldsymbol{\mathcal{H}}_h$ has been computed.

Summing up, we face the following abstract discrete eigenvalue problem: Seek $\mathbf{u}_h \in \boldsymbol{\mathcal{V}}_h$ such that

$$a(\mathbf{u}_h, \mathbf{v}_h) = \lambda\, (\mathbf{u}_h, \mathbf{v}_h)_0 \quad \forall \mathbf{v}_h \in \boldsymbol{\mathcal{V}}_h\ , \tag{2.6}$$

where $a(\cdot,\cdot)$ stands for the positive semidefinite bilinear form from (2.1) or (2.2). We associate operators $A_h : \boldsymbol{\mathcal{V}}_h \mapsto \boldsymbol{\mathcal{V}}_h'$ and $M_h : \boldsymbol{\mathcal{V}}_h \mapsto \boldsymbol{\mathcal{V}}_h'$ with the bilinear forms in (2.6), which converts it into an operator equation

$$A_h \mathbf{u}_h = \lambda M_h \mathbf{u}_h\ . \tag{2.7}$$

The basis of $\boldsymbol{\mathcal{V}}_h$ dual to the set of d.o.f. is called the nodal basis $\{\mathbf{b}_\iota\}_{\iota \in J}$, with J a suitable index set. The basis functions are locally supported and satisfy

$$\|\mathbf{b}_\iota\|_0 \leq C \operatorname{diam}\operatorname{supp}(\mathbf{b}_\iota)\, \|\mathbf{b}_\iota\|_A\ , \qquad \iota \in J\ , \tag{2.8}$$

with $\|\cdot\|_A$ the *energy-seminorm* induced by $a(\cdot,\cdot)$. Given the nodal basis, (2.7) can also be read as a matrix equation, A_h being the stiffness matrix and M_h the mass matrix, which are both large and sparse.

We follow the convention that functions will be given Roman symbols, whereas Greek letters are used for functionals. Those related to the base space $\boldsymbol{\mathcal{V}}_h$ will be given bold tokens, whereas entities from the potential space $\mathcal{S}_h$ are printed in plain style.

3. PPINVIT.

Standard inverse iteration (without shift) for an eigenvalue problem $A_h \mathbf{u}_h = \lambda M_h \mathbf{u}_h$ with symmetric *positive definite* operators $A_h : \boldsymbol{\mathcal{V}}_h \mapsto \boldsymbol{\mathcal{V}}_h'$, $M_h := \boldsymbol{\mathcal{V}}_h \mapsto \boldsymbol{\mathcal{V}}_h'$ computes a new iterate $\mathbf{x}_h^{\text{new}} \in \boldsymbol{\mathcal{V}}_h$ from the old $\mathbf{x}_h \in \boldsymbol{\mathcal{V}}_h$ through

$$\mathbf{y}_h = \kappa A_h^{-1} M_h \mathbf{x}_h^{\text{old}}, \quad \mathbf{x}_h^{\text{new}} := \mathbf{y}_h / \|\mathbf{y}_h\|_0$$

for some $\kappa \neq 0$. First, observe that the choice of κ is immaterial. Therefore, we may set $\kappa = r(\mathbf{x}_h)$, where

$$r(\mathbf{x}_h) = \frac{\langle A_h \mathbf{x}_h, \mathbf{x}_h \rangle}{\langle M_h \mathbf{x}_h, \mathbf{x}_h \rangle} \tag{3.1}$$

denotes the Rayleigh quotient and $\langle \cdot, \cdot \rangle$ the duality pairing. This choice of κ has the effect that $\mathbf{y}_h - \mathbf{x}_h$ converges to zero when $r(\mathbf{x}_h)$ approaches the smallest eigenvalue. Thus we recover the typical situation, where a correction is determined by solving a linear system with a small residual as its right-hand side. This paves the way for the application of a *preconditioner* $B_h : \boldsymbol{\mathcal{V}}_h' \mapsto \boldsymbol{\mathcal{V}}_h$, an approximate inverse of A_h, to compute $\mathbf{y}_h$. We arrive at the update formula

$$\mathbf{y}_h = \mathbf{x}_h - B_h(A_h \mathbf{x}_h - r(\mathbf{x}_h) M_h \mathbf{x}_h), \quad \mathbf{x}_h^{\mathrm{new}} := \mathbf{y}_h / \left\| \mathbf{y}_h \right\|_0 \ , \tag{3.2}$$

which is the basic building block for the algorithm of PINVIT [61]. The iterates will converge linearly to an eigenvector belonging to the smallest eigenvalue. The theoretically possible but unlikely case that PINVIT gets stuck in a higher eigenvalue does not occur in practice due to rounding errors. If an invariant subspace corresponding to the s smallest eigenvalues is desired, we can resort to the subspace variant. After a Rayleigh–Ritz projection, it updates each of the s Ritz vectors $\mathbf{x}_h^1, \ldots, \mathbf{x}_h^s$ according to (3.2) with $r(\mathbf{x}_h)$ replaced by the Ritz values [20, 59].

Let us return to the actual setting, in which A_h is only *positive semidefinite*. Then it is natural to demand that $\mathbf{y}_h$ is contained in the $\boldsymbol{L}^2(\Omega)$-orthogonal complement of $\mathrm{Ker}(A_h)$, as this is satisfied for any eigenvector belonging to a nonzero eigenvalue. In other words, the (exact) inverse iteration should be based on the pseudoinverse $A_h^\dagger : \boldsymbol{\mathcal{V}}_h' \mapsto \boldsymbol{\mathcal{V}}_h$. Then $\mathbf{x}_h$ will converge to an eigenvector corresponding to λ_1 as long as the starting vector (for the case of exact arithmetic) is not orthogonal to that eigenvector.

The pseudoinverse $A^\dagger$ is elusive and has to be approximated. We suggest to do so by means of a plain multigrid method. It relies on a hierarchy of nested meshes $\mathcal{T}_0 \prec \mathcal{T}_1 \prec \cdots \prec \mathcal{T}_L := \mathcal{T}_h$ and the corresponding finite element spaces $\boldsymbol{\mathcal{V}}_0 \subset \boldsymbol{\mathcal{V}}_1 \subset \cdots \subset \boldsymbol{\mathcal{V}}_L := \boldsymbol{\mathcal{V}}_h$. The natural way to create such meshes is through successive refinement of an initial rather coarse mesh $\mathcal{T}_0$, as described in [6, 11] for tetrahedral meshes. The refinement strategies make sure that the shape regularity of $\mathcal{T}_0$ is almost preserved for all finer meshes.

We instantly get a sequence of operators $A_l : \boldsymbol{\mathcal{V}}_l \mapsto \boldsymbol{\mathcal{V}}_l'$ generated by the bilinear form $a(\cdot, \cdot)$ on $\boldsymbol{\mathcal{V}}_l$. The embedding of the spaces $\boldsymbol{\mathcal{V}}_{l-1} \subset \boldsymbol{\mathcal{V}}_l$ spawns the canonical prolongation operators $I_l : \boldsymbol{\mathcal{V}}_{l-1} \mapsto \boldsymbol{\mathcal{V}}_l$, $l = 1, \ldots, L$. Their adjoints $I_l^* : \boldsymbol{\mathcal{V}}_l' \mapsto \boldsymbol{\mathcal{V}}_{l-1}'$ are known as restrictions [40, sect. 3.6]. These operators are purely local and cheaply implemented [43].

The definition of the symmetric multigrid preconditioner is based on the recursive algorithm sketched in Figure 1. There R_l^T is defined by $\left\langle \boldsymbol{\rho}_l, R_l^T \boldsymbol{\phi}_l \right\rangle = \langle \boldsymbol{\phi}_l, R_l \boldsymbol{\rho}_l \rangle$, $\boldsymbol{\rho}_l, \boldsymbol{\phi}_l \in \boldsymbol{\mathcal{V}}_l'$. Based on the algorithm of Figure 1 the application of the multigrid preconditioner $B_h : \boldsymbol{\mathcal{V}}_h \mapsto \boldsymbol{\mathcal{V}}_h'$ can be realized as follows:

$$\mathbf{c}_h := B_h \boldsymbol{\rho}_h \quad \Longleftrightarrow \quad \mathbf{c}_h := 0; \quad \mathsf{mgcycle}(L, \mathbf{c}_h, \boldsymbol{\rho}_h) \ . \tag{3.3}$$

The operators $R_l : \boldsymbol{\mathcal{V}}_l' \mapsto \boldsymbol{\mathcal{V}}_l$ occurring in the algorithm are conventional smoothing operators on level l, $l = 1, \ldots, L$. We will consider only point smoothers of Jacobi- or

```
mgcycle< $A$ > (int $l$,reference $\mathbf{u}_l \in \boldsymbol{\mathcal{V}}_l$, const $\boldsymbol{\rho}_l \in \boldsymbol{\mathcal{V}}'_l$)
{
  if($l == 0$) {   $\mathbf{u}_0 = A_0^{\dagger}\boldsymbol{\rho}_0$   }
  else   {
    // Presmoothing
    for(int $i = 0$ ; $i < \mu_1$ ; $++i$) {   $\mathbf{u}_l \leftarrow \mathbf{u}_l + R_l(\boldsymbol{\rho}_l - A_l\mathbf{u}_l)$   }
    // Coarse grid correction
    $\boldsymbol{\sigma}_l := \boldsymbol{\rho}_l - A_h\mathbf{u}_l$;   $\boldsymbol{\rho}_{l-1} := I_l^*\boldsymbol{\sigma}_l$;   $\mathbf{c}_{l-1} := 0 \in \boldsymbol{\mathcal{V}}_{l-1}$
    for(int $i = 0$ ; $i < \nu$ ; $++i$) {   mgcycle< $A$ >($l-1$,$\mathbf{c}_{l-1}$,$\boldsymbol{\rho}_{l-1}$)   }
    $\mathbf{u}_l \leftarrow \mathbf{u}_l + I_l\mathbf{c}_{l-1}$
    // Postsmoothing
    for(int $i = 0$ ; $i < \mu_2$ ; $++i$) {   $\mathbf{u}_l \leftarrow \mathbf{u}_l + R_l^T(\boldsymbol{\rho}_l - A_l\mathbf{u}_l)$   }
  }
}
```

FIG. 1. *Multigrid algorithm defining the preconditioner B_h. The parameters $\mu_1, \mu_2, \nu \in \mathbb{N}$ define the type of the cycle. For $\nu = 1$ we get a $V(\mu_1, \mu_2)$-cycle and for $\nu = 2$ a $W(\mu_1, \mu_2)$-cycle.*

Gauß–Seidel-type. For the latter, one sweep on level l, $l = 1, \dots, L$, with initial guess $u_l \in \boldsymbol{\mathcal{V}}_l$ and right-hand side $\boldsymbol{\rho}_l \in \boldsymbol{\mathcal{V}}'_l$ reads

$$\textbf{foreach}(\iota \in J_l) \quad \{ \quad \mathbf{u}_l \leftarrow \mathbf{u}_l + \frac{\langle \boldsymbol{\rho}_l, \mathbf{u}_l \rangle}{a(\mathbf{b}_\iota, \mathbf{b}_\iota)} \cdot \mathbf{b}_\iota \quad \} \, .$$

Though A_h is singular, relaxation will go smoothly, as (2.8) guarantees $a(\mathbf{b}_\iota, \mathbf{b}_\iota) > 0$. However, $\mathbf{b}_\iota$ does not exactly belong to $\mathrm{Ker}(A_l)^\perp$. Thus, the action of B_h will invariably introduce components in $\mathrm{Ker}(A_h)$ into the iterates. Eventually the iterates might tumble into the kernel.

To prevent this, we have to weed out the kernel contributions as soon as they are introduced. Formally, this can be done by projecting $\mathbf{y}_h$ from (3.2) onto $\mathrm{Ker}(A_h)^\perp$. Fortunately, if Ω is contractible, the representation of $\mathrm{Ker}(A_h)$ through discrete potentials according to $\mathrm{Ker}(A_h) = G_h\mathcal{S}_h$ enables us to express the $\boldsymbol{L}^2(\Omega)$-orthogonal projection $P_h : \boldsymbol{\mathcal{V}} \mapsto \mathrm{Ker}(A)^\perp$ through

$$P_h := Id - G_h T_h^{\dagger} G_h^* M_h \;, \tag{3.4}$$

where $T_h : \mathcal{S}_h \mapsto \mathcal{S}'_h$ is the operator associated with the bilinear form

$$d : \mathcal{S} \times \mathcal{S} \mapsto \mathbb{R}, \quad d(u_h, v_h) = (G_h u_h, G_h v_h)_0 \,, \quad u_h, v_h \in \mathcal{S}_h \;. \tag{3.5}$$

Yet, the exact computation of $T_h^{\dagger}\rho_h$ for some $\rho_h \in \mathcal{S}'_h$ is hardly feasible. Just recall that in the case of the eigenvalue problem in $\boldsymbol{H}(\mathbf{curl}; \Omega)$ the operator T_h is the discrete Laplacian, i.e., in general described by a huge sparse stiffness matrix. Therefore, we cannot help using an approximate pseudoinverse also in this case. A multigrid scheme analogous to the one outlined in Figure 1 comes in handy, this time to be conducted in the potential space with the operators A_l replaced by their counterparts $T_l : \mathcal{S}_l \mapsto \mathcal{S}'_l$. This will yield an approximate projection $\widetilde{P}_h$

$$\widetilde{P}_h := Id - G_h C_h G_h^* M_h \;, \tag{3.6}$$

where C_h stands for the approximate (pseudo-)inverse of T_h furnished by the multigrid cycle. Reassuringly, we do not have to worry about pollution in $\mathrm{Ker}(G_h)$ this time

```
update(reference $\mathbf{x}_h \in \boldsymbol{\mathcal{V}}_h$,$\kappa \in \mathbb{R}$)
{
  project($\mathbf{x}_h$)
  $\boldsymbol{\phi}_h := A_h\mathbf{x}_h$ ; $\boldsymbol{\psi}_h := M_h\mathbf{x}_h$
  $\boldsymbol{\rho}_h := \boldsymbol{\phi}_h - \kappa \cdot \boldsymbol{\psi}_h$
  $\mathbf{c}_h := 0 \in \boldsymbol{\mathcal{V}}_h$
  mgcycle< $A$ >($L$,$\mathbf{c}_h$,$\boldsymbol{\rho}_h$)
  $\mathbf{x}_h \leftarrow \mathbf{x}_h - \mathbf{c}_h$
  project($\mathbf{x}_h$)
  $\mathbf{x}_h \leftarrow \mathbf{x}_h/|\mathbf{x}_h|$
}
```

```
project(reference $\mathbf{x}_h \in \boldsymbol{\mathcal{V}}_h$)
{
  // Treat harmonic vectorfields
  for(int $i = 1$; $i \leq q$; $++i$)
  { $\mathbf{x}_h \leftarrow \mathbf{x}_h - \left(\widetilde{\mathbf{h}}_i, \mathbf{x}_h\right)_0 \cdot \widetilde{\mathbf{h}}_i$ }
  $\boldsymbol{\eta}_h := M_h\mathbf{x}_h$
  $\phi_h := G_h^*\boldsymbol{\eta}_h$
  $c_h = 0 \in \mathcal{S}_h$
  mgcycle< $T$ >($L$,$c_h$,$\phi_h$)
  $\mathbf{x}_h \leftarrow \mathbf{x}_h - G_h c_h$
}
```

FIG. 2. *Update procedure for PPINVIT.*

because in (3.4) the operator G_h is applied to the result, suppressing any kernel component.

If we have to take into account discrete harmonic vectorfields in $\boldsymbol{\mathcal{H}}_h := \text{Span}\,\{\mathbf{h}_1, \ldots, \mathbf{h}_q\}$, their basis should be approximately orthogonalized to $G_h\mathcal{S}_h$. This can be done once and for all before the actual eigenvalue computations, utilizing a few steps of the approximate multigrid projection $\widetilde{P}_h$. For the sake of efficiency, a nested iteration approach should be employed. Eventually, the basis functions should be $\boldsymbol{L}^2(\Omega)$-orthonormalized to each other by solving a small linear system of equations. If $\widetilde{\mathbf{h}}_1, \ldots, \widetilde{\mathbf{h}}_q$ are the functions thus obtained, $\widetilde{\boldsymbol{\mathcal{H}}}_h := \text{Span}\{\widetilde{\mathbf{h}}_1, \ldots, \widetilde{\mathbf{h}}_q\}$ will be another suitable space of discrete harmonic vectorfields. Given this preprocessing, orthogonality to $\widetilde{\boldsymbol{\mathcal{H}}}_h$ can be easily enforced.

The final algorithm implementing the inexact projection is given in Figure 2 (right). We point out that G_h is a local operator, too, whose matrix representation can be derived from the embedding $G_h\mathcal{S}_h \subset \boldsymbol{\mathcal{V}}_h$ [43, sect. 6]. Let us elucidate this for edge elements: Assuming nodal bases of $\boldsymbol{\mathcal{V}}_h$ and $\mathcal{S}_h$ the evaluation of G_h boils down to simply distributing the nodal values from vertices (to which d.o.f. of $\mathcal{S}_h$ are associated) to edges, taking into account their orientations by means of weights $+1$ or -1.

In the end, incorporating the total action of project into $\tilde{P}_h$, we get the following update formula for an approximate eigenvector:

$$\mathbf{y}_h = \tilde{P}_h(Id - B_h(A_h - \kappa M_h))\tilde{P}_h\mathbf{x}_h, \quad \mathbf{x}_h^{\text{new}} = \mathbf{y}_h / \left\|\mathbf{y}_h\right\|_0 \ . \tag{3.7}$$

Cast into an algorithm, this yields the procedure update displayed in Figure 2 (left).

It is hazardous to replace κ in (3.7) by the plain Rayleigh quotient (3.1) because significant kernel components might remain after the inexact projection. If we set $\kappa = r(\mathbf{x}_h)$, we might encounter $\kappa \ll \lambda$ though $A_h\mathbf{x}_h = \lambda M_h P_h \mathbf{x}_h$; i.e., the components of $\mathbf{x}_h$ in $\text{Ker}(A_h)^{\perp}$ already provide the desired eigenvector. Guided by the idea that the scheme should come close to inverse iteration in the complement $\text{Ker}(A_h)^{\perp}$ we should choose $\kappa = r_{\perp}(\mathbf{x}_h) = \langle A_h\mathbf{x}_h, \mathbf{x}_h\rangle / (P_h\mathbf{x}_h, P_h\mathbf{x}_h)_{\boldsymbol{L}^2(\Omega)}$. In practice, we are denied this option as $P_h\mathbf{x}_h$ is not available. However, we still want a replacement for $r_{\perp}$ that is insensitive to kernel components. A promising candidate is the "two-step Rayleigh quotient"

$$r_Q(\mathbf{x}) = \frac{\langle A_h M_h^{-1} A_h \mathbf{x}_h, \mathbf{x}_h\rangle}{\langle A_h\mathbf{x}_h, \mathbf{x}_h\rangle} \ , \tag{3.8}$$

```
ppinvit_step (reference $(\theta_1, \dots, \theta_s)^T \in \mathbb{R}^s$, reference $(\mathbf{x}_h^1, \dots, \mathbf{x}_h^s) \in (\boldsymbol{\mathcal{V}}_h)^s$)
{
  // Ritz-projection
  for($i = 1$ ; $i \leq s$ ; $++i$)   {
    $\boldsymbol{\phi}_h^i := A_h \mathbf{x}_h^i$;   $\mathbf{z}_h^i = \lambda_i \mathbf{x}_h^i$;   $\mathsf{cg}^m < M_h > (\mathbf{z}_h^i, \boldsymbol{\phi}_h^i)$
    $a_{ii} := \langle \boldsymbol{\phi}_h^i, \mathbf{z}_h^i \rangle$;   $m_{ii} := \langle \boldsymbol{\phi}_h^i, \mathbf{x}_h^i \rangle$
    for($j = 1$ ; $j < i$ ; $++i$) { $a_{ij} = a_{ji} := \left\langle \boldsymbol{\phi}_h^j, \mathbf{z}_h^i \right\rangle$;   $m_{ij} = m_{ji} := \left\langle \boldsymbol{\phi}_h^j, \mathbf{x}_h^i \right\rangle$ }
  } // Rayleigh–Ritz procedure
  $\mathsf{A}_s := (a_{ij}) \in \mathbb{R}^{s,s}$;   $\mathsf{M}_s := (m_{ij}) \in \mathbb{R}^{s,s}$;
  Find $\mathsf{Y} \in \mathbb{R}^{s,s}$ and Ritz values $\Theta = \operatorname{diag}(\theta_1, \dots, \theta_s)$ such that   $\mathsf{A}_s \mathsf{Y} = \mathsf{M}_s \mathsf{Y} \Theta$
  // Ritz vectors
  $(\mathbf{x}_h^1, \dots, \mathbf{x}_h^s) \leftarrow (\mathbf{x}_h^1, \dots, \mathbf{x}_h^s) \cdot \mathsf{Y}$
  // Approximate projected inverse iteration
  for($i = 1$ ; $i \leq s$ ; $++i$)   { update($\mathbf{x}_h^i, \theta_i$)  }
}
```

FIG. 3. *One step of the subspace variant of the algorithm for PPINVIT.* $\mathsf{cg}^m < M_h > (\mathbf{z}_h, \text{Œ}_h)$ *refers to* $m \in \mathbb{N}$ *CG-steps for the solution of* $M_h \mathbf{z}_h = \text{Œ}_h$.

with $r_Q(\mathbf{x}) \geq r_\perp(\mathbf{x}_h) \geq r(\mathbf{x}_h)$. Obviously, it yields an eigenvalue if we have already hit an eigenvector in $\operatorname{Ker}(A_h)^\perp$. Two issues arise, nevertheless: First, there is a risk of breakdown if $A_h \mathbf{x}_h = 0$. This means that the current approximate eigenvector lies in $\operatorname{Ker}(A_h)$, which hints at inadequate approximate projections. A way to detect and cure this condition will be discussed in section 5. The second problem is that the evaluation of (3.8) entails the solution of a linear system $M_h \mathbf{z}_h = A_h \mathbf{x}_h$ for the mass matrix M_h. As M_h is well conditioned unless some elements are badly distorted, a few steps of an iterative method (CG,Gauß–Seidel) will usually give a reasonable approximate solution. Moreover, if $(\lambda, \mathbf{x}_h)$ is already close to an eigenvalue/eigenvector pair, $\lambda \mathbf{x}_h$ is an excellent initial guess.

With all building blocks in place, we can now state the crucial update step of the algorithm for the computation of the s, $s \in \mathbb{N}$, smallest nonzero eigenvalues and corresponding eigenvectors of (2.7). Its details are given in Figure 3. The procedure ppinvit_step is meant to improve on approximations θ_i and $\mathbf{x}_h^i$, $i = 1, \dots, s$, for eigenvalues and eigenvectors.

The discussion of termination criteria is postponed until section 5. Initial guesses for the eigenvectors can easily be obtained through nested iteration by prolongating approximate eigenfunction from coarser grids.

Remark. For positive definite operators the Rayleigh–Ritz method is often applied to a modified/enlarged subspace (consisting of the actual subspace, the actual search directions, and possibly the old iterates). This is known to improve convergence [48,49] if $A_h > 0$. Yet, this trick is not advisable for the semidefinite problem because a massive amplification of kernel components might occur.

4. Convergence.

The theoretical examination of the algorithm starts with the $\boldsymbol{L}^2(\Omega)$-orthogonal decomposition and dual polar decomposition

$$\boldsymbol{\mathcal{V}}_h = \boldsymbol{\mathcal{X}}_h \otimes \boldsymbol{\mathcal{Z}}_h \,,\; \boldsymbol{\mathcal{Z}}_h := \operatorname{Ker}(A_h), \quad \boldsymbol{\mathcal{V}}_h' = \boldsymbol{\mathcal{X}}_h' \otimes \boldsymbol{\mathcal{Z}}_h' \,. \tag{4.1}$$

With respect to the splittings (4.1) the operators can be written in block form. For a symmetric preconditioner it reads

$$B_h = \begin{pmatrix} B_{\perp\!\!\perp} & B_{0\perp} \\ B_{0\perp}^T & B_{00} \end{pmatrix} : \boldsymbol{\mathcal{X}}_h' \otimes \boldsymbol{\mathcal{Z}}_h' \mapsto \boldsymbol{\mathcal{X}}_h \otimes \boldsymbol{\mathcal{Z}}_h \ , \tag{4.2}$$

and for the other operators

$$A_h = \begin{pmatrix} A_\perp & 0 \\ 0 & 0 \end{pmatrix}, \ M_h = \begin{pmatrix} M_\perp & 0 \\ 0 & M_0 \end{pmatrix}, \ A_h^\dagger = \begin{pmatrix} A_\perp^{-1} & 0 \\ 0 & 0 \end{pmatrix} \ \widetilde{P}_h = \begin{pmatrix} Id_\perp & 0 \\ 0 & P_0 \end{pmatrix}.$$

These formulas are immediate from the definition of the operators and the properties of the splittings. Be aware that $B_{0\perp} \neq 0$ causes the pollution by kernel components, and $P_0 \neq 0$ hints at an inexact projection.

Using $A_h A_h^\dagger + Q_h^* = Id_h^*$, where $Q_h : \boldsymbol{\mathcal{V}}_h \mapsto \boldsymbol{\mathcal{Z}}_h$ is the $\boldsymbol{L}^2(\Omega)$-orthogonal projection, we obtain from (3.7) with $\kappa = r_Q(\mathbf{x}_h)$

$$\mathbf{y}_h = \widetilde{P}_h \left((I_h - B_h A_h)(I_h - \kappa A_h^\dagger M_h) + \kappa B_h Q_h^* M_h + \kappa A_h^\dagger M_h \right) \widetilde{P}_h \mathbf{x}_h \ .$$

Splitting $\mathbf{y}_h = \mathbf{y}^0 + \mathbf{y}^\perp$, $\mathbf{x}_h = \mathbf{x}^0 + \mathbf{x}^\perp$, $\mathbf{x}^0, \mathbf{y}^0 \in \boldsymbol{\mathcal{Z}}_h$, $\mathbf{y}^\perp, \mathbf{x}^\perp \in \boldsymbol{\mathcal{X}}_h$ and plugging in the block forms of the operators leads to

$$\begin{pmatrix} \mathbf{y}^\perp - \mathbf{z}_h \\ \mathbf{y}^0 \end{pmatrix} = \begin{pmatrix} Id_\perp & 0 \\ 0 & P_0 \end{pmatrix} \left(\begin{pmatrix} Id_\perp - B_{\perp\!\!\perp} A_\perp & 0 \\ -B_{0\perp}^T A_\perp & Id_0 \end{pmatrix} \begin{pmatrix} Id_\perp - \kappa A_\perp^{-1} M_\perp & 0 \\ 0 & Id_0 \end{pmatrix} \right.$$
$$\left. + \begin{pmatrix} 0 & \kappa B_{0\perp} M_0 \\ 0 & \kappa B_{00} M_0 \end{pmatrix} \right) \begin{pmatrix} Id_\perp & 0 \\ 0 & P_0 \end{pmatrix} \begin{pmatrix} \mathbf{x}^\perp \\ \mathbf{x}^0 \end{pmatrix}$$

with $\mathbf{z}_h := \kappa A_\perp^{-1} M_\perp \mathbf{x}^\perp$. This results in a kind of *error propagation equation*

$$\begin{pmatrix} \mathbf{y}^\perp - \mathbf{z}_h \\ \mathbf{y}^0 \end{pmatrix} = \begin{pmatrix} Id_\perp - B_{\perp\!\!\perp} A_\perp & \kappa B_{0\perp} M_0 P_0 \\ -P_0 B_{0\perp}^T A_\perp & P_0 (Id_0 + \kappa B_{00} M_0) P_0 \end{pmatrix} \begin{pmatrix} \mathbf{x}^\perp - \mathbf{z}_h \\ \mathbf{x}^0 \end{pmatrix} . \tag{4.3}$$

Note that $(\mathbf{z}_h, 0)^T$ is what an exact inverse iteration for the pseudoinverse would give us before scaling. Thus (4.3) reflects how much PPINVIT differs from an exact inverse iteration. Next, we aim at quantitative estimates of this deviation. To this end we seek bounds for norms of the block-operators in (4.3).

Various norms need to be considered for operators $X_h : \boldsymbol{\mathcal{V}}_h \mapsto \boldsymbol{\mathcal{V}}_h$:

$$\|X_h\|_{0\to 0} := \sup_{\mathbf{v}_h \in \boldsymbol{\mathcal{V}}_h} \frac{\|X_h \mathbf{v}_h\|_0}{\|\mathbf{v}_h\|_0}, \quad \|X_h\|_{Z\to 0} := \sup_{\mathbf{v}_h \in \boldsymbol{\mathcal{Z}}_h} \frac{\|X_h \mathbf{v}_h\|_0}{\|\mathbf{v}_h\|_0},$$
$$\|X_h\|_{A\to A} := \sup_{\mathbf{v}_h \in \boldsymbol{\mathcal{V}}_h} \frac{\|X_h \mathbf{v}_h\|_A}{\|\mathbf{v}_h\|_A}, \quad \|X_h\|_{A\to 0} := \sup_{\mathbf{v}_h \in \boldsymbol{\mathcal{V}}_h} \frac{\|X_h \mathbf{v}_h\|_0}{\|\mathbf{v}_h\|_A},$$
$$\|X_h\|_{Z\to A} := \sup_{\mathbf{v}_h \in \boldsymbol{\mathcal{Z}}_h} \frac{\|X_h \mathbf{v}_h\|_A}{\|\mathbf{v}_h\|_0}.$$

In order to bound the operator norm $\|I - B_{\perp\!\!\perp} A_\perp\|_{A\to A}$ we remember that $\|\cdot\|_A$ is the energy-seminorm in $\boldsymbol{H}(\mathbf{curl};\Omega)$ and $\boldsymbol{H}(\mathrm{div};\Omega)$, respectively. In other words, this norm agrees with the convergence rate of the multigrid method in the energy-seminorm. In [41, 43, 44] it was shown that this convergence rate is bounded away from 1 independently of the number L of grid levels involved in the multigrid scheme. This justifies the assumption

$$\textbf{(A1)} \qquad \|Id_\perp - B_{\perp\!\!\!\perp} A_\perp\|_{A\to A} \le \gamma < 1\ .$$

In fact, numerical experiments give evidence that we can expect γ to be smaller than 0.5, at worst [43].

Next, we have to gauge the impact of the inexact projection. Again, we can rely on theoretical results and practical experience with multigrid methods to justify

$$\textbf{(A2)} \qquad \|G_h(Id_h - C_hT_h)u_h\|_0 \le \beta\,\|G_hu_h\|_0 \quad \forall u_h \in \mathcal{S}_h$$

for $\beta < 1$ uniformly in L. Note that β is the convergence rate of the iterative solver in potential space. The practical range for β will be the same as for γ. From (**A2**) and $T_h = G_h^* M_h G_h$ we conclude

$$\lambda_{\max}(G_h(T_h^\dagger - C_h)G_h^*M_h) = \lambda_{\max}(G_h^*M_hG_h(T_h^\dagger - C_h)) = \lambda_{\max}(Id_h - T_hC_h) = \beta\ .$$

Because of $(Id_h - G_hT_h^\dagger G_h^* M_h)\mathbf{x}_h^0 = 0$ for $\mathbf{x}_h^0 \in \boldsymbol{\mathcal{Z}}_h$, this teaches us that

$$\begin{aligned}\|P_0\mathbf{x}_h^0\|_0 &= \|\mathbf{x}_h^0 - G_hC_hG_h^*M_h\mathbf{x}_h^0\|_0\\ &\le \left\|(Id_h - G_hT_h^\dagger G_h^*M_h + G_h(T_h^\dagger - C_h)G_h^*M_h)\mathbf{x}_h^0\right\|_0\\ &\le \left\|G_h(T_h^\dagger - C_h)G_h^*M_h\mathbf{x}_h^0\right\|_0 \le \beta\,\|\mathbf{x}_h^0\|_0\ .\end{aligned} \tag{4.4}$$

The remaining terms involving the multigrid preconditioner will be tackled under the restrictive assumption of uniform refinement creating a quasi-uniform hierarchy of meshes. Hence, we have inverse estimates at our disposal. In addition, we take for granted a geometric decrease of the meshwidths and symmetric smoothing operators, i.e., $R_h = R_h^T$. Please note that these assumptions are needed only for the sake of theoretical treatment.

Under the above circumstances, the symmetric bilinear form $s_l : \boldsymbol{\mathcal{V}}_l \times \boldsymbol{\mathcal{V}}_l \mapsto \mathbb{R}$ that defines the smoother R_l via

$$s_l(R_l\boldsymbol{\phi}_l, \mathbf{v}_l) = \boldsymbol{\phi}_l(\mathbf{v}_l) \quad \forall \mathbf{v}_l \in \boldsymbol{\mathcal{V}}_l,\ \boldsymbol{\phi}_l \in \boldsymbol{\mathcal{V}}_l'$$

fulfills

$$Ch_l^{-2}\,(\mathbf{u}_l, \mathbf{u}_l)_0 \le s(\mathbf{u}_l, \mathbf{u}_l) \le Ch_l^{-2}\,(\mathbf{u}_l, \mathbf{u}_l)_0 \quad \forall \mathbf{u}_l \in \boldsymbol{\mathcal{V}}_l\ .$$

For the point smoothers that we have in mind, this is a consequence of (2.8). In particular, $s(\cdot,\cdot)$ turns out to be positive definite. Then the Cauchy–Schwarz inequality gives for $\mathbf{x}_l^0 \in \boldsymbol{\mathcal{Z}}_l$

$$\begin{aligned}\|R_lM_l\mathbf{x}_l^0\|_0^2 &\le Ch_l^2 s(R_lM_l\mathbf{x}_l^0, R_lM_l\mathbf{x}_l^0) = Ch_l^2 \sup_{\mathbf{w}_l\in\boldsymbol{\mathcal{V}}_l} \frac{s(R_lM_l\mathbf{x}_l^0, \mathbf{w}_l)^2}{s(\mathbf{w}_l, \mathbf{w}_l)}\\ &\le Ch_l^4 \sup_{\mathbf{w}_l\in\boldsymbol{\mathcal{V}}_l} \frac{\langle M_l\mathbf{x}_l^0, \mathbf{w}_l\rangle^2}{\|\mathbf{w}_l\|_0^2} \le Ch_l^4\,\|\mathbf{x}_l^0\|_0^2\ ,\end{aligned}$$

from which we infer

$$\|R_lM_l\|_{Z\to 0} \le Ch_l^2 \quad \text{and} \quad \|R_lM_l\|_{Z\to A} \le Ch_l\ . \tag{4.5}$$

The latter estimate is a consequence of the inverse inequalities (2.3) that involve

$$a(\mathbf{u}_l, \mathbf{u}_l) \le Ch_l^{-2} \left\| \mathbf{u}_l \right\|_0^2 \ . \tag{4.6}$$

The same arguments reveal

$$\begin{aligned} \left\| R_l A_l \mathbf{x}_l \right\|_0^2 &\le Ch_l^2 \, s(R_l A_l \mathbf{x}_l, R_l A_l \mathbf{x}_l) = Ch_l^2 \sup_{\mathbf{w}_l \in \boldsymbol{\mathcal{V}}_l} \frac{s(R_l A_l \mathbf{x}_l, \mathbf{w}_l)^2}{s(\mathbf{w}_l, \mathbf{w}_l)} \\ &\le Ch_l^4 \sup_{\mathbf{w}_l \in \boldsymbol{\mathcal{X}}_l} \frac{\langle A_l \mathbf{x}_l, \mathbf{w}_l \rangle^2}{\left\| \mathbf{w}_l \right\|_0^2} \le Ch_l^2 \left\| \mathbf{x}_l \right\|_A^2 \ . \end{aligned}$$

The inverse inequality in $\boldsymbol{\mathcal{V}}_l$ is concealed in the final estimate. Eventually,

$$\left\| R_l A_l \right\|_{A \to 0} \le Ch_l \quad \text{and} \quad \left\| R_l A_l \right\|_{A \to A} \le C \ . \tag{4.7}$$

Now we are in a position to examine the full multigrid cycle. For the sake of simplicity the analysis is confined to a V(1,1)-cycle.

THEOREM 4.1. *Assume a hierarchy of shape-regular, quasi-uniform meshes with geometrically decreasing meshwidths. In addition, the smoothers have to be symmetric and are to provide convergent linear iterations in the* $\left\| \cdot \right\|_A$*-seminorm. Then the preconditioners* B_l *spawned by* $V(1,1)$*-cycles satisfy*

$$\left\| B_l A_l \right\|_{A \to 0} \le K_A, \quad \left\| B_l M_l \right\|_{Z \to 0} \le K_0, \quad \left\| B_l M_l \right\|_{Z \to A} \le K_\perp \ ,$$

with constants $K_A > 0$*,* $K_0 > 0$*, and* $K_\perp > 0$ *that depend on the shape regularity of the meshes* $\mathcal{T}_0, \dots, \mathcal{T}_L$*, but not on* l*.*

Proof. The recursive nature of the multigrid algorithm suggests that we study two subsequent levels l and $l-1$. For ease of notation, we will use a subscript h to refer to level l (fine grid), and H will tag entities associated with level $l-1$ (coarse grid).

We retrace the single steps of the algorithm of Figure 1 and start with $\boldsymbol{\rho}_h := A_h \mathbf{x}_h$ for some $\mathbf{x}_h \in \boldsymbol{\mathcal{V}}_h$. Presmoothing takes it to $\mathbf{w}_h := R_h A_h \mathbf{x}_h$ since a zero initial guess has to be used. Afterwards, the coarse grid correction will result in $\mathbf{c}_H := B_H I_h^* A_h (\mathbf{x}_h - \mathbf{w}_h)$. Then, with $P_h^H : \boldsymbol{\mathcal{V}}_h \mapsto \boldsymbol{\mathcal{X}}_H$ denoting the $a(\cdot,\cdot)$-orthogonal projection, we infer from $I_h^* A_h = A_H P_h^H$ that $\mathbf{c}_H = B_H A_H P_h^H (Id_h - R_h A_h) \mathbf{x}_h$. As the multigrid method is supposed to converge in the $\left\| \cdot \right\|_A$-seminorm, $\left\| Id_H - B_H A_H \right\|_{A \to A} < 1$ is guaranteed, so that

$$\left\| B_H A_H \right\|_{A \to A} \le 2 \ . \tag{4.8}$$

The smoother alone also provides a convergent iteration, i.e., $\left\| Id_h - R_h A_h \right\|_{A \to A} < 1 \Rightarrow \left\| R_h A_h \right\|_{A \to A} \le 2$, such that

$$\left\| \mathbf{c}_H \right\|_0 \le \left\| B_H A_H \right\|_{A \to 0} \left\| \mathbf{x}_h \right\|_A , \quad \left\| \mathbf{c}_H \right\|_A \le 2 \left\| \mathbf{x}_h \right\|_A \ .$$

With $\mathbf{u}_h := \mathbf{w}_h + I_h \mathbf{c}_H$, which, due to (4.7), fulfills

$$\left\| \mathbf{u}_h \right\|_0 \le (\left\| B_H A_H \right\|_{A \to 0} + Ch) \left\| \mathbf{x} \right\|_A , \quad \left\| \mathbf{u}_h \right\|_A \le 4 \left\| \mathbf{x}_h \right\|_A \ ,$$

we can express the result of postsmoothing as

$$B_h A_h \mathbf{x}_h = \mathbf{u}_h + R_h (A_h \mathbf{x}_h - A_h \mathbf{u}_h) = \mathbf{w}_h + \mathbf{u}_h - R_h A_h \mathbf{u}_h \ .$$

Again, we invoke (4.7) and see

$$\|B_h A_h \mathbf{x}_h\|_0 \leq (\|B_H A_H\|_{A\to 0} + Ch)\,\|\mathbf{x}\|_A + Ch\,\|\mathbf{u}_h\|_A \leq (\|B_H A_H\|_{A\to 0} + Ch)\,\|\mathbf{x}\|_A\,.$$

Consequently, $\|B_h A_h\|_{A\to 0} \leq \|B_H A_H\|_{A\to 0} + Ch$. Taking into account that $B_0 = A_0^\dagger$, i.e., $\|B_0 A_0\|_{A\to 0} = 0$, and the geometric decrease of the meshwidths, this ensures $\|B_h A_h\|_{A\to 0} \leq K_A$, for $K_A > 0$ independent of the level.

Analogous considerations can be carried out with $\boldsymbol{\rho}_h := M_h \mathbf{x}_h^0$ for some $\mathbf{x}_h^0 \in \boldsymbol{\mathcal{Z}}_h$. Presmoothing yields $\mathbf{w}_h := R_h \boldsymbol{\rho}_h = R_h M_h \mathbf{x}_h^0$, and after the cycle on the coarse grid we end up with

$$\mathbf{c}_H = \mathbf{c}_1 + \mathbf{c}_2 := B_H I_h^* M_h \mathbf{x}_h^0 + B_H I_h^* A_h \mathbf{w}_h\,. \tag{4.9}$$

As $I_h^* M_h = M_H Q_h^H$, where $Q_h^H : \boldsymbol{\mathcal{V}}_h \mapsto \boldsymbol{\mathcal{V}}_H$ is the $\boldsymbol{L}^2(\Omega)$-orthogonal projection, we get for the first contribution to $\mathbf{c}_H$

$$\|\mathbf{c}_1\|_0 \leq \|B_H M_H\|_{Z\to 0}\,\|\mathbf{x}_h^0\|_0\,, \quad \|\mathbf{c}_1\|_A \leq \|B_H M_H\|_{Z\to A}\,\|\mathbf{x}_h^0\|_0\,. \tag{4.10}$$

Similarly, from $I_h^* A_h = A_H P_h^H$, (4.5), and (4.8), it follows that

$$\begin{aligned}\|\mathbf{c}_2\|_0 &\leq \|B_H A_H P_h^H \mathbf{w}_h\|_0 \leq \|B_H A_H\|_{A\to 0}\,\|\mathbf{w}_h\|_A \leq Ch\,\|B_H A_H\|_{A\to 0}\,\|\mathbf{x}_h^0\|_0\,,\\ \|\mathbf{c}_2\|_A &\leq \|B_H A_H P_h^H \mathbf{w}_h\|_A \leq \|B_H A_H\|_{A\to A}\,\|\mathbf{w}_h\|_A \leq Ch\,\|\mathbf{x}_h^0\|_0\,.\end{aligned}$$

Combining this with (4.9), (4.10), and the result of the first part of the proof yields

$$\|\mathbf{c}_H\|_0 \leq (\|B_H M_H\|_{Z\to 0} + ChK_A)\,\|\mathbf{x}_h^0\|_0\,, \tag{4.11}$$

$$\|\mathbf{c}_H\|_A \leq (\|B_H M_H\|_{Z\to A} + Ch)\,\|\mathbf{x}_h^0\|_0\,. \tag{4.12}$$

Next, we consider the coarse grid correction $\mathbf{u}_h = \mathbf{w}_h + I_h \mathbf{c}_H$. As the prolongation is an identity mapping in disguise, the following estimates are straightforward from (4.11), (4.12), and (4.5):

$$\|\mathbf{u}_h\|_0 \leq \left(\|B_H M_H\|_{Z\to 0} + ChK_A + Ch^2\right)\|\mathbf{x}_h^0\|_0\,, \tag{4.13}$$

$$\|\mathbf{u}_h\|_A \leq (\|B_H M_H\|_{Z\to A} + Ch)\,\|\mathbf{x}_h^0\|_0\,. \tag{4.14}$$

The postsmoothing results in $B_h M_h \mathbf{x}_h^0 = \mathbf{u}_h + \mathbf{w}_h - R_h A_h \mathbf{u}_h$. By (4.7) and (4.8) we know

$$\|R_h A_h \mathbf{u}_h\|_0 \leq Ch\,\|\mathbf{u}_h\|_A\,, \quad \|R_h A_h \mathbf{u}_h\|_A \leq C\,\|\mathbf{u}_h\|_A\,. \tag{4.15}$$

First, appealing to (4.14), (4.5), and the assumed $\|\cdot\|_A$-convergence of the smoothing iterations, we find

$$\|B_h M_h \mathbf{x}_h^0\|_A \leq (\|B_H M_H\|_{Z\to A} + Ch)\,\|\mathbf{x}_h^0\|_0\,.$$

On the coarsest grid $l = 0$, we have $B_{00} M_0 = 0$ and $B_{0\perp} M_0 = 0$. Then the geometric decrease of the meshwidth leads to the assertion of the lemma for $\|B_l M_l\|_{Z\to A}$. The same argument can be applied to $\|B_l M_l\|_{Z\to 0}$ because from (4.13), (4.5), (4.15), and (4.14) we can infer

$$\|B_h M_h \mathbf{x}_h^0\|_0 \leq \left(\|B_H M_H\|_{Z\to 0} + Ch(K_A + K_\perp) + Ch^2\right)\|\mathbf{x}_h^0\|_0\,. \qquad \square$$

Now, we can convert (4.3) into the estimates

$$\begin{pmatrix} \left\|\mathbf{y}_h^\perp - \kappa\mathbf{z}_h\right\|_A \\ \left\|\mathbf{y}_h^0\right\|_0 \end{pmatrix} \leq \begin{pmatrix} \gamma & \kappa K_\perp \beta \\ K_A\beta & \beta^2(1+\kappa K_0) \end{pmatrix} \begin{pmatrix} \left\|\mathbf{x}_h^\perp - \kappa\mathbf{z}_h\right\|_A \\ \left\|\mathbf{x}_h^0\right\|_0 \end{pmatrix} . \tag{4.16}$$

All the constants are basically independent of the meshwidth and the number L of levels involved in the multigrid solvers. A quantitative conclusion can be rigorously drawn from (4.16) (see [45] for the technical proof).

THEOREM 4.2. *If $\kappa := \kappa_Q < \kappa^*$ for all steps of the iteration and the initial iterate $\mathbf{x}_h$ satisfies*

$$\frac{\left\|\mathbf{x}_h^0\right\|_0}{\left\|\mathbf{x}_h\right\|_A} \leq \frac{(1-\gamma)}{2\kappa^* K_\perp},$$

then this will hold for all other iterates, provided that β is below a threshold depending on $K_A, K_\perp, K_0$, and γ only.

This guarantees that the iterates cannot plunge into the kernel if a sufficient damping of kernel components is achieved by the projection.

Heuristic insights into the significance of (4.16) can be gained from the theory of PINVIT in the positive definite case [50, 61, 62]. If, with $\kappa = r_\perp$, for some positive $\Gamma < 1$

$$\left\|\mathbf{y}_h^\perp - \kappa A_\perp^{-1} M_\perp \mathbf{x}_h^\perp\right\|_A \leq \Gamma \left\|\mathbf{x}_h^\perp - \kappa A_\perp^{-1} M_\perp \mathbf{x}_h^\perp\right\|_A , \tag{4.17}$$

the PINVIT convergence theory in [50] gives a simple sharp estimate for the Rayleigh quotient $r_\perp$ of the new iterate $\mathbf{y}_h^\perp$ (cf. Theorem 1 in [50]), demonstrating that PINVIT converges at least linearly to the eigenvalue λ_1.

In the semidefinite case the theory applies to PINVIT in $\mathrm{Ker}(A)^\perp$ only for the unrealistic choice of a perfect projection, i.e., $\beta = 0$. Such an exact projection is prohibitively expensive. Hence, we prefer to deal with moderately small $\beta > 0$. Consequently, we have to accept that the iterates in $\mathrm{Ker}(A)^\perp$ will inevitably be perturbed by the term $\kappa B_{0\perp} M_0 P_0 \mathbf{x}^0$ in (4.3). Nevertheless, (4.17) may hold with $\Gamma < 1$ throughout the iteration. Then convergence according to the PINVIT theory is guaranteed (apart from the minor modification of replacing $r_\perp$ by r_Q). Unfortunately, if $\left\|\mathbf{x}_h^\perp - \kappa A_\perp^{-1} M_\perp \mathbf{x}_h^\perp\right\|_A \ll \left\|\mathbf{x}_h^0\right\|_0$ the constant in (4.17) may blow up. However, (4.16) teaches that in this case a significant reduction of the kernel component will be achieved, provided that β is sufficiently small. Hence, $\Gamma > 1$ might happen in a single step, but in the next step (4.17) is likely to hold with a rather small Γ. In other words, for a $\beta \ll 1$ the kernel components are damped out in the course of the iteration. Therefore, the perturbations become more and more insignificant. This effect is elusive, and we have not succeeded in giving a rigorous analysis.

Let us study as a low-dimensional model system the eigenvalue problem for $A_h = \mathrm{diag}(2, 5, 8, 0)$ and $M_h = Id$ with $\gamma = 0.5$. Then the preconditioner B_h is a 4×4 matrix. We take $b_{\max}$ as the bound for the absolute value of B_{00} and for the A-norm of $B_{0\perp}$.[1] Figure 4 illustrates the relative damping of kernel components by PPINVIT. Therefore, the maximal ratio $|x_h^0|/|x_h|$ after 10 steps of PPINVIT using the two-step

[1] Note that the low dimension of the model problem is motivated by the fact that PINVIT takes its extremal convergence in a two-dimensional space which is spanned by those eigenvectors whose corresponding eigenvalues enclose the Rayleigh quotient of the actual iterate. Moreover, as a result of Lemma 3.1 in [61] the assumption that all eigenvalues are of the algebraic multiplicity 1 appears nonrestrictive.

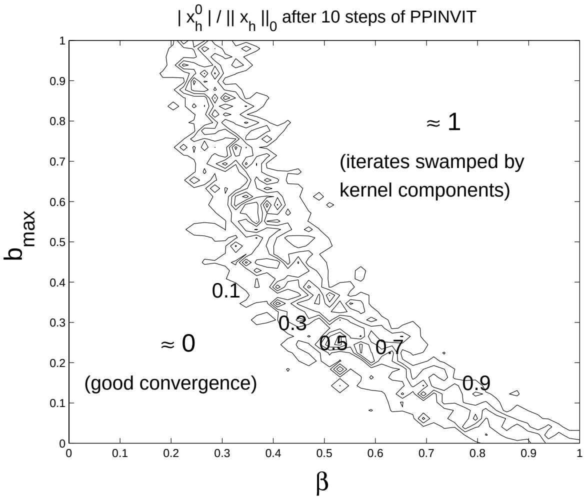


FIG. 4. *Relative damping of the kernel component for the model problem.*

Rayleigh quotient r_Q is displayed in a contour plot for $\beta \in [0,1]$ and $b_{\max} \in [0,1]$. The maximal ratio has been determined for each point of the underlying 50×50 grid among 150000 combinations of random preconditioners and random start vectors with a fixed initial kernel component. For $\beta < 0.2$ the kernel is damped out very well independently of the choice of $b_{\max}$. We conclude that a sufficiently good projection can weed out the kernel components after a modest number of iterations.

The same conclusion can be drawn from a second experiment: We recorded the components of those iterates that display the poorest convergence of r_Q in the first and fourth step of PPINVITT (Figure 5). For 10^7 combinations of random preconditioners ($\gamma = 0.5$, $\beta = 0.25$, $b_{\max} = 0.2$) the components are plotted against the two-step Rayleigh quotients r_Q of those iterates. In the first step of PPINVIT the kernel components appear as the dominating part. In the fourth step we identify the ith and $i+1$th component as the prevailing ones if $r_Q \in [\lambda_i, \lambda_{i+1}]$. The kernel has all but disappeared. Thus, Figure 5 highlights a key trait of PPINVIT: *Convergence is brought about by the subtle interaction of multiple steps.*

5. Projection control and termination criteria. The theoretical considerations highlight the importance of a good projection: It goes without saying that the method will fail if the projection is too weak to reign in kernel components. Taking the cue from Theorem 4.2, we aim to force the ratio $\|\mathbf{x}_h^0\|_0 : \|\mathbf{x}_h\|_A$ below a threshold $\delta > 0$ for all iterates.

From the properties of the inexact projection $\widetilde{P}_h$ and (4.4) we learn that

$$\frac{\beta}{1-\beta} \cdot \frac{\|\mathbf{x}_h - \widetilde{P}\mathbf{x}_h\|_0}{\|\mathbf{x}_h\|_A} \le \delta \quad \Longrightarrow \quad \frac{\|P_0\mathbf{x}_h\|_0}{\|\mathbf{x}_h\|_A} \le \delta\,. \tag{5.1}$$

Of course, good bounds for β are hard to get. We take a crude estimate based on the decrease of the $L^2(\Omega)$-norm of the residual during a multigrid sweep. It is computed whenever a projection is carried out, and β is chosen to be the maximum of all estimates thus obtained. The final adaptive projection is depicted in Figure 6. There, $\sigma \in]0,1[$ is a safety factor intended to prevent gross underestimation of β.

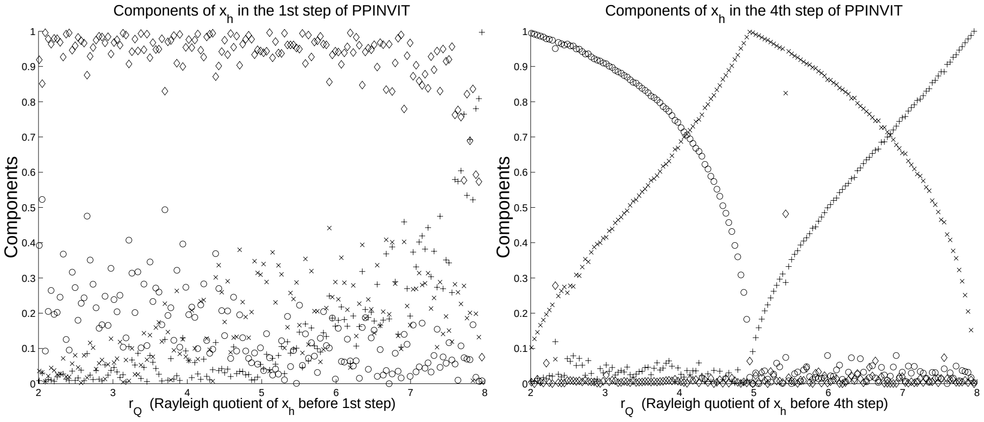


FIG. 5. *Components of those iterates x_h showing poorest convergence (i.e., the smallest decrease of the Rayleigh quotient in one PPINVIT step) for the model problem. Components in the directions of eigenvectors to $\lambda_1, \lambda_2, \lambda_3$, and* 0 *correspond to symbols* $(\circ, \times, +, \diamond)$. *Left: first step of PPINVIT. Right: fourth step of PPINVIT.*

```
project(reference $\mathbf{x}_h \in \boldsymbol{\mathcal{V}}_h$,$\delta > 0$,$\sigma \in [0,1[$)
{
    for(int $i = 1$; $i \le q$; $++i$)   { $\mathbf{x}_h \leftarrow \mathbf{x}_h - (\widetilde{\mathbf{h}}_i\mathbf{x}_h \cdot \widetilde{\mathbf{h}}_i)$ }
    $\nu := \langle A_h\mathbf{x}_h, \mathbf{x}_h\rangle$
    do   {
        $\phi_h := G_h^* M_h \mathbf{x}_h$;   $c_h = 0 \in \mathcal{S}_h$
        mgcycle< $T$ >($L$,$c_h$,$\phi_h$);   $\mathbf{z}_h := G_h c_h$;   $\mathbf{x}_h \leftarrow \mathbf{x}_h - \mathbf{z}_h$
        $\rho_h := \phi_h - T_h c_h$;   $\beta := \langle\phi_h, \phi_h\rangle \,/\, \langle\rho_h, \rho_h\rangle$;   $\bar{\beta} \leftarrow \max\{\bar{\beta}, (1-\sigma)\beta + \sigma\}$
        $\mu := \dfrac{\bar{\beta}}{(1-\bar{\beta})} \cdot \dfrac{\langle M_h\mathbf{z}_h, \mathbf{z}_h\rangle}{\nu}$
    } while ($\mu > \delta$);
}
```

FIG. 6. *Enhanced projection with adaptive control. The global variable $\bar{\beta}$ is set to* 0 *initially.*

Our next concern is the termination of the iteration. After the completion of the Rayleigh–Ritz procedure (cf. Figure 3) there are on hand the Ritz values θ_i and the Ritz vectors $\mathbf{x}_h^i$ with $\|\mathbf{x}_h^i\|_0 = 1$, $i = 1, \dots, s$. The M_h^{-1}-norm of the residual $\mathbf{r}_h^i = A_h\mathbf{x}_h^i - \theta_i M_h \mathbf{x}_h^i$ provides a simple residual bound [64] for the quality of the Ritz value θ_i. It is guaranteed that in each interval $[\theta_i - \|\mathbf{r}_h^i\|_{M_h^{-1}}, \theta_i + \|\mathbf{r}_h^i\|_{M_h^{-1}}]$ an eigenvalue of (A_h, M_h) is contained. For disjoint intervals the θ_i provide s approximations to s different eigenvalues of (A_h, M_h). In practice, the inverse of the mass matrix may be approximated through one Gauß–Seidel step. This yields a quantity that is equivalent to the M_h^{-1}-norm independent of the meshwidth.

Beyond, we suggest that the ratio r/r_Q of Rayleigh-quotients of approximate eigenfunctions is used to judge whether the iteration has been successful. Only if it is very close to 1 can the results be trusted.

Remark. It is not a moot point that δ should be reduced during the iteration as the approximate eigenvectors get closer and closer to the exact eigenvectors. However, we failed to find a strategy with heuristic, let alone rigorous, underpinning.

6. Numerical experiments. For all numerical experiments covered in this section we relied on lowest order edge/face elements on uniform Cartesian grids. Ritz projections and eigenvalues/eigenfunctions on the coarsest grids were determined by means of suitable LAPACK routines. All computations were carried out in double precision arithmetic, whereas the matrices were stored in single precision format. We computed seven eigenfunction/eigenvalue pairs in each case. Unless stated otherwise, the eigenvalue problem (1.1) is investigated.

For the tests we resorted to a setting marked by discontinuous coefficients and a reentrant corner, more precisely $\Omega :=]0,1[^3 \backslash [0,\frac{1}{3}]^3$, $\Gamma_N := \bar{\Omega} \cap \{x_1 = 1\}$, and

$$\alpha(\mathbf{x}) = \begin{cases} 1 & \text{if } x_2 < \frac{2}{3} \;, \\ 100 & \text{if } x_2 \geq \frac{2}{3} \text{ and } \frac{1}{3} \leq x_1, x_3 \leq \frac{2}{3} \;, \\ 10 & \text{elsewhere .} \end{cases}$$

This creates challenging conditions for multigrid methods. The uniform grid on level l, $l = 0, \ldots, 5$, consisted of $26 \cdot 8^l$ equal cubes. This means that for $l = 5$ the discretized problems (1.1) and (1.2) feature 2599200 and 2626560 d.o.f., respectively.

Experiment 1. To begin with, we monitored the behavior of the "two-step" Rayleigh quotients r_Q from (3.8) and the M_l^{-1}-norms $\left\langle M_l^{-1} \boldsymbol{\rho}_l, \boldsymbol{\rho}_l \right\rangle^{1/2}$ of the residuals $\boldsymbol{\rho}_l := A_l \mathbf{x}_l - r_Q(\mathbf{x}_l) M_l \mathbf{x}_l$ of approximate eigenfunctions $\mathbf{x}_l \in \boldsymbol{\mathcal{V}}_l$, $l = 3, 4, 5$. Of course, $M_l^{-1} \boldsymbol{\rho}_l$ could not be computed exactly but was realized by two Gauß–Seidel (GS)-sweeps. Both quantities were tracked for 3 eigenfunctions (belonging to eigenvalues #1, #3, and #5) during 15 iterations of PPINVIT. Interpolants of solenoidal polynomial vectorfields $\mathbf{f}_i := x_m^r x_l^r \cdot \vec{e}_k$, $\{m, l, k\} = \{1, 2, 3\}$, $k = i \mod 3$, $r = i \operatorname{div} 3$, served as initial guesses for the ith eigenfunction, $i = 1, \ldots, 7$. By and large, the effect of different initial guesses quickly abates during the iterations.

Single symmetric multigrid V(1,1)-cycles with lexicographic GS-smoothers were used both in $\boldsymbol{\mathcal{V}}_l$ and potential space. The inverse mass matrix required for the calculation of the two-step Rayleigh quotient was approximated by three steps of the preconditioned conjugate gradient (PCG) method with a symmetric GS-preconditioner.

The values of r_Q were considered as useful approximations of eigenvalues and thus it makes sense to examine their relative errors with respect to "exact discrete eigenvalues" (computed by nested iteration up to a relative error of 10^{-6}). The results for problem (1.1) are plotted in Figure 7.

First of all, a rather uniform decrease of the errors/residual norms takes place. Next, we note that the M_l^{-1}-norm of the residual permits us to assess the accuracy of the approximate eigenvalue very well. As expected, the larger the eigenvalue the poorer the convergence (up to a total failure to converge for the seventh eigenvalue). One should follow the customary advice that dimension of the subspace should be chosen somewhat larger than the number of eigenvalue one is interested in.

Experiment 2. Retaining most of the setting of the previous experiment we studied the impact of choices of different multigrid cycles for both the update and projection step. In Table 1 we report the rate

$$\rho = \left((r_Q(\mathbf{x}_h^{(10)}) - \lambda_{\text{exact}}) / (r_Q(\mathbf{x}_h^{(2)}) - \lambda_{\text{exact}}) \right)^{\frac{1}{8}} \tag{6.1}$$

of convergence of the eigenvalue approximations between the second and tenth step of the iteration.

The effect of very accurate preconditioners/projections seems to be limited, as predicted by the theory of PINVIT: We cannot be better than exact inverse iteration

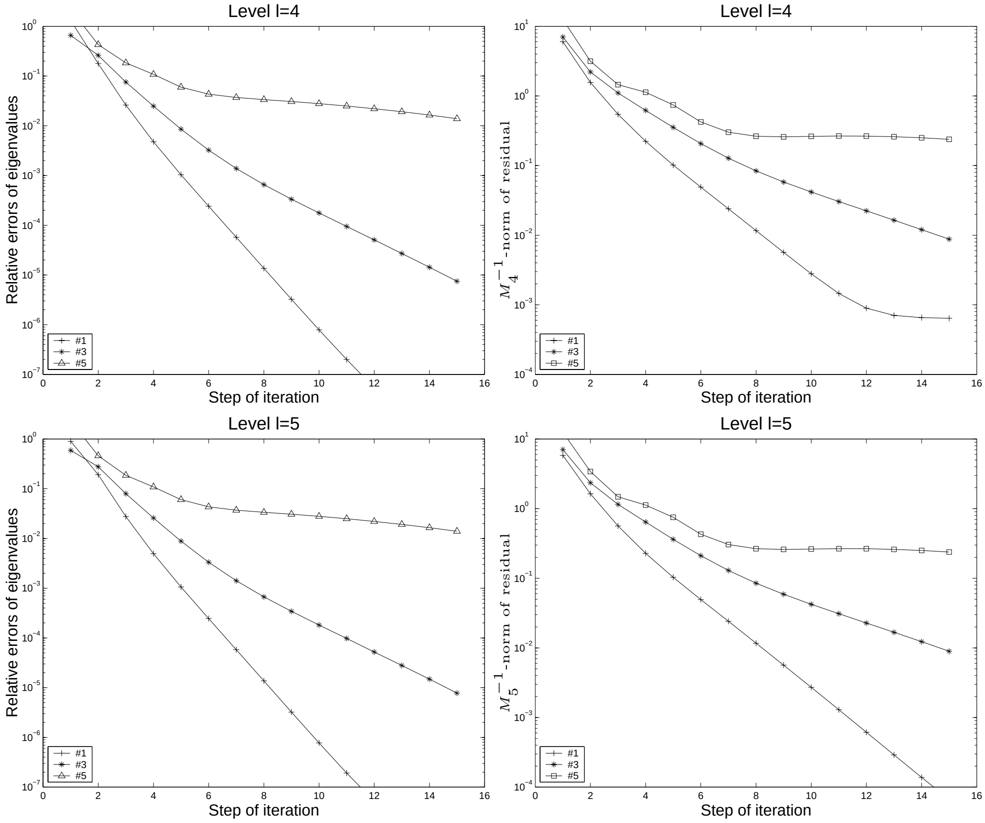


FIG. 7. *Experiment* 1: *Left: Relative errors of two-step Rayleigh quotients when compared with "exact discrete eigenvalues". Right: Approximate M_l^{-1}-norms of residuals.*

and improving the preconditioner has little impact if the Rayleigh quotient is already close to the exact eigenvalue. Moreover, the rates highlight the danger of too many smoothing steps on edge d.o.f. This will make the constants K_A, $K_\perp$, and K_0 from (4.16) soar and has to be offset by improved projection.

Experiment 3. Of course, choosing random initial guesses is foolish, in particular, as a nested iteration approach will do much better in a multilevel environment. The behavior of the approximate M_l^{-1}-norms of the eigenfunction residuals during nested iteration was recorded for the various settings. On each level l the iteration was terminated if $\left\langle M_l^{-1}\boldsymbol{\rho}_l, \boldsymbol{\rho}_l\right\rangle^{\frac{1}{2}} \leq \tau$ for eigenfunction #1 through #5, where $\tau > 0$ is a prescribed threshold.

The same multigrid cycles as before were employed. Moreover, we chose $\tau = 0.01$ and $\tau = 0.1$. In the latter case the evaluation of M_h^{-1} in the computation of r_Q was based on only one symmetric GS-sweep, which is much cheaper than the three PCG-steps used for the former case. The results can be looked at in Figure 8.

As before, the data strikingly confirm that the convergence of multigrid-PPINVIT is independent of the depth of refinement: About the same number of iterations is required on each level to achieve the prescribed reduction of the norm of the residuals.

Experiment 4. In experiment 3 we boldly relied on a single symmetric GS-sweep to get an approximation for r_Q. Now, we aim to investigate how different approximations

TABLE 1

Experiment 2: *Error reduction factors ρ for eigenvalues* #1 *through* #7 *and different choices of cycles for approximate inverses B_h and C_h.*

Cycles	l	#1	#2	#3	#4	#5	#6	#7
B_h: V(1,1) C_h: V(1,1)	2	0.19	0.25	0.40	0.65	0.72	0.62	0.66
	3	0.21	0.24	0.40	0.64	0.72	0.63	0.66
	4	0.21	0.24	0.40	0.64	0.71	0.63	0.66
	5	0.21	0.23	0.40	0.64	0.70	0.63	0.66
B_h: V(1,1) C_h: W(2,2)	2	0.19	0.25	0.40	0.65	0.72	0.62	0.66
	3	0.21	0.24	0.40	0.64	0.72	0.63	0.66
	4	0.21	0.24	0.40	0.64	0.71	0.63	0.66
	5	0.21	0.23	0.40	0.64	0.70	0.63	0.66
B_h: W(2,2) C_h: V(1,1)	2	0.18	0.16	0.34	0.62	0.71	0.63	0.67
	3	0.14	0.17	0.36	0.63	0.71	0.64	0.67
	4	0.10	0.18	0.37	0.63	0.71	0.64	0.67
	5	0.12	0.19	0.37	0.63	0.72	0.64	0.70
B_h: W(2,2) C_h: W(2,2)	2	0.17	0.16	0.35	0.62	0.71	0.63	0.67
	3	0.14	0.17	0.36	0.63	0.71	0.64	0.67
	4	0.11	0.18	0.37	0.63	0.71	0.64	0.67
	5	0.13	0.18	0.37	0.63	0.71	0.64	0.67
B_h: W(4,4) C_h: V(1,1)	2	0.18	0.16	0.35	0.62	0.72	0.63	0.67
	3	0.15	0.17	0.36	0.63	0.71	0.64	0.67
	4	0.13	0.18	0.37	0.63	0.72	0.64	0.75
	5	0.15	0.32	0.56	0.78	**1.01**	**1.24**	**1.21**

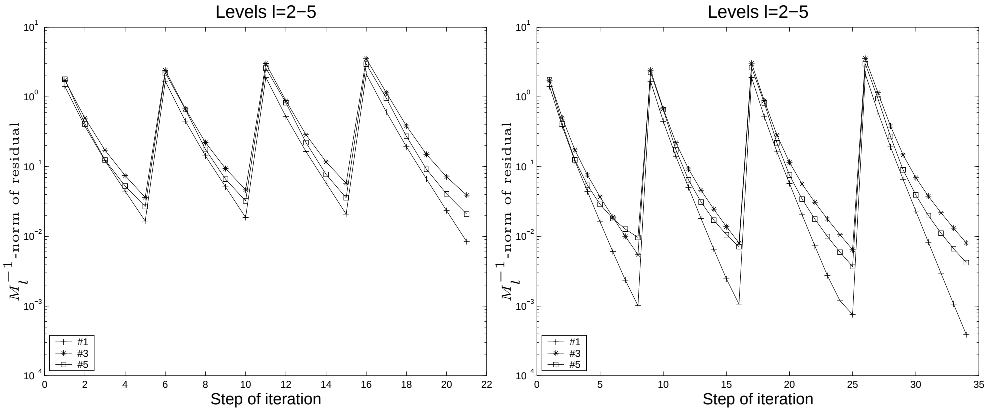


FIG. 8. *Experiment* 3: *Nested iteration, M_l^{-1} realized by* 3 *PCG-steps. Left: $\tau = 0.1$. Right: $\tau = 0.01$.*

of M_h^{-1} perform. In particular, we used either one, two, or three steps of SGS or PCG. In any other respect the setting is just borrowed from experiment 1. The decrease of the errors in the eigenvalue approximations according to (6.1) are listed in Table 2. The picture is blurred, but the general message is that spending much effort on M_h^{-1} does not pay off in terms of asymptotic convergence rates, but can boost convergence during the first few steps of the iteration, when eigenvalue approximations are still poor.

Experiment 5. We repeated experiment 3 with face elements instead of edge elements. The (level-dependent) M_l^{-1}-norms of some residuals are plotted in Figure 9.

TABLE 2
Experiment 4*: Behavior of different approximations of* M_5^{-5} *on level* 5*,* B_h*:* $V(1,1)$*,* C_h*:* $V(1,1)$*.*

M_l^{-1} by	#1	#2	#3	#4	#5	#6	#7
Relative error after step #8 of PPINVIT							
3 PCG	$1.4{\cdot}10^{-5}$	$1.1{\cdot}10^{-5}$	6.71e-04	0.012	0.033	0.033	0.10
2 PCG	$1.9{\cdot}10^{-5}$	$1.2{\cdot}10^{-5}$	0.00064	0.011	0.033	0.033	0.10
1 PCG	0.035	0.0044	0.00071	0.013	0.015	0.0074	0.039
3 SGS	$2.0{\cdot}10^{-5}$	$1.2{\cdot}10^{-5}$	0.00066	0.012	0.033	0.033	0.10
2 SGS	$4.4{\cdot}10^{-5}$	$6.1{\cdot}10^{-5}$	0.00073	0.012	0.033	0.032	0.10
1 SGS	0.0074	0.015	0.018	0.026	0.054	0.045	0.13
Decrease ρ of relative error according to (6.1)							
3 PCG	0.21	0.23	0.40	0.64	0.70	0.63	0.66
2 PCG	0.22	0.21	0.38	0.61	0.70	0.63	0.66
1 PCG	0.41	0.35	0.29	0.40	0.43	0.36	0.46
3 SGS	0.20	0.19	0.36	0.57	0.65	0.61	0.63
2 SGS	0.17	0.15	0.27	0.40	0.45	0.42	0.46
1 SGS	0.19	0.21	0.22	0.24	0.28	0.26	0.29

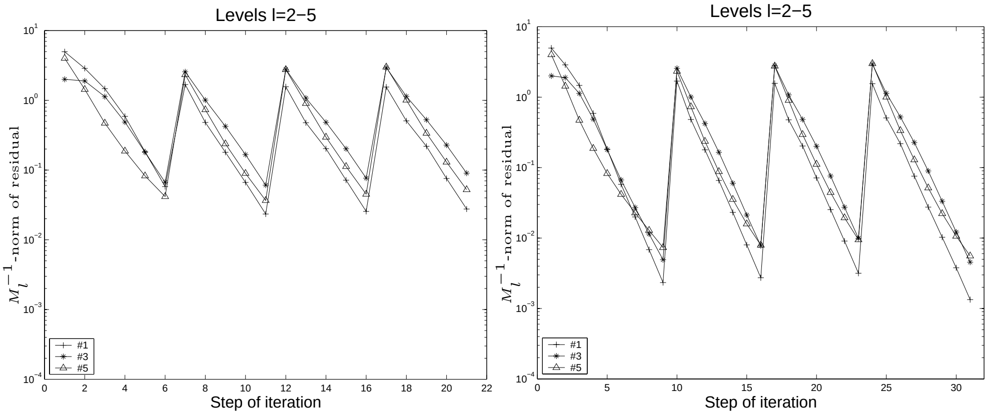


FIG. 9. *Experiment* 5*: Nested iteration for* face elements*,* M_h^{-1} *realized by* 3 *PCG-steps. Left:* $\tau = 0.1$*. Right:* $\tau = 0.01$*.*

Qualitatively, the residual norms behave as for edge elements. If we had reported all experiments for face elements, too, this statement would have been appropriate for any other case.

Experiment 6. The final experiment scrutinizes whether projection control as discussed in section 5 can really offset poor projections. To that end we used a plain symmetric GS-sweep for C_h, which yields an outrageously bad $\widetilde{P}_h$ on fine grids. Otherwise, the algorithm of the first experiment was retained and we focused on level 4.

Projection control with $\delta = 0.1$, $\delta = 0.01$, and a safety factor $\sigma = \frac{1}{4}$ was enabled. In addition, as we observed wild fluctuations of the number of GS-steps suggested by the projection control, we imposed that this number could not shrink by more than a factor of two between subsequent projections (zig-zag-evasion). In Figure 10 the behavior of relative errors of eigenvalues is logged. Some ratios $r(\mathbf{x}_h)/r_Q(\mathbf{x}_h)$ are recorded in Figure 11.

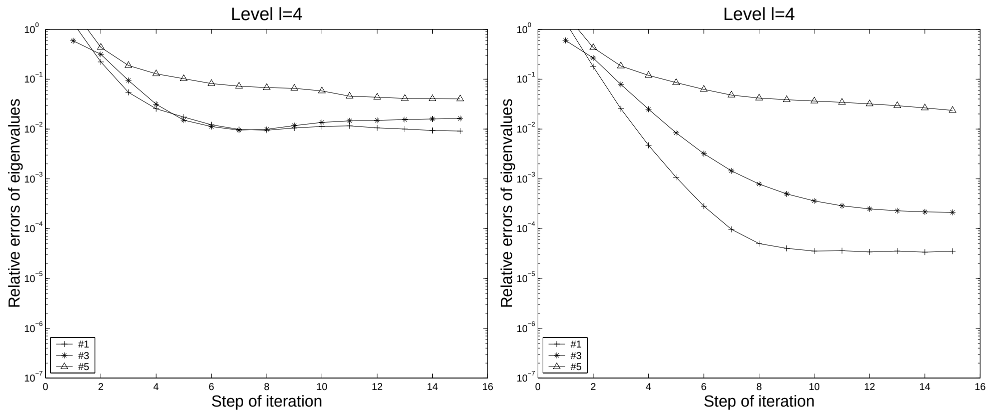


FIG. 10. *Experiment* 6: *Projection control with* $\delta = 0.1$ *(left),* $\delta = 0.01$ *(right).*

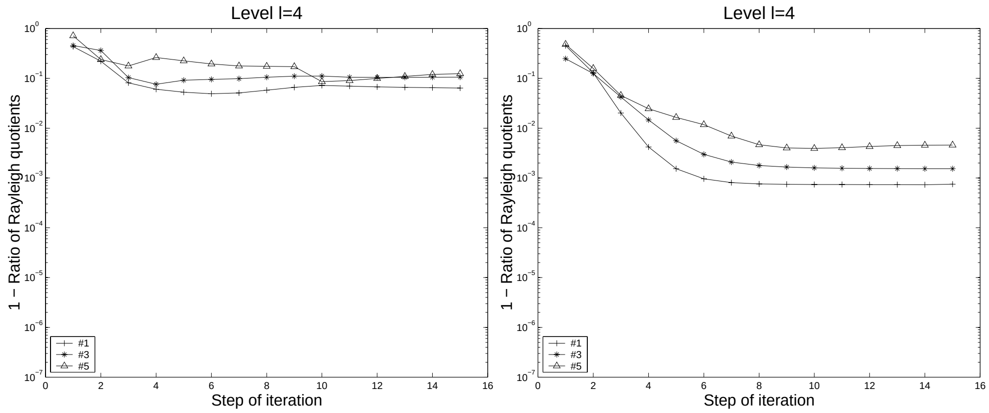


FIG. 11. *Experiment* 6: $1 - r(\mathbf{x}_h)/r_Q(\mathbf{x}_h)$ *for different projection controls with* $\delta = 0.1$ *(left),* $\delta = 0.01$ *(right).*

Sufficiently tight projection control ensures convergence. However, the results also highlight the need for an adaptive choice of δ because it seems hard to determine in advance when δ will be sufficiently small. This experiment also hints that the ratio of Rayleigh quotients, which is to tend to 1, can help detect ineffective projections.

Remark. In our experimental setting singularities of eigenfunctions at reentrant edges and coefficient discontinuities can be expected. Therefore, adaptive refinement would be advisable. It could be done along the lines of [53, 60], taking into account the techniques of [8]. Numerical evidence hints that no deterioration of multigrid convergence will occur on locally refined meshes [7, 55]. Thus, PPINVIT will probably not be affected much.

7. Conclusion.

We presented a multigrid-preconditioned inverse iteration method for the solution of large discrete semidefinite eigenvalue problems in $\boldsymbol{H}(\mathbf{curl};\Omega)$ and $\boldsymbol{H}(\mathrm{div};\Omega)$. Though a complete theoretical analysis is still missing, there is strong numerical evidence that the method inherits the efficiency of multigrid based iterative solution procedures. Besides a thorough theoretical understanding many issues remain to be examined: among others, improved adaptive projection control, detection of topological complications, and potential acceleration by means of preconditioned steepest descent or various kinds of subspace enlargements and, last but not least, the benefit of shift strategies.

REFERENCES

[1] S. Adam, P. Arbenz, and R. Geus, *Eigenvalue Solvers for Electromagnetic Fields in Cavities*, Tech. Rep. 275, Institute of Scientific Computing, ETH Zürich, Zürich, Switzerland, 1997.

[2] C. Amrouche, C. Bernardi, M. Dauge, and V. Girault, *Vector potentials in three–dimensional nonsmooth domains*, Math. Methods Appl. Sci., 21 (1998), pp. 823–864.

[3] D. Arnold, R. Falk, and R. Winther, *Multigrid in* $H(\mathrm{div})$ *and* $H(\mathbf{curl})$, Numer. Math., 85 (2000), pp. 175–195.

[4] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, eds., *Templates for the solution of algebraic eigenvalue problems: A practical guide*, SIAM, Philadelphia, 2000.

[5] R. E. Bank, *Analysis of a multilevel inverse iteration procedure for eigenvalue problems*, SIAM J. Numer. Anal., 19 (1982), pp. 886–898.

[6] E. Bänsch, *Local mesh refinement in* 2 *and* 3 *dimensions*, IMPACT Comput. Sci. Engrg., 3 (1991), pp. 181–191.

[7] R. Beck, P. Deuflhard, R. Hiptmair, R. Hoppe, and B. Wohlmuth, *Adaptive multilevel methods for edge element discretizations of Maxwell's equations*, Surveys Math. Indust., 8 (1998), pp. 271–312.

[8] R. Beck, R. Hiptmair, and B. Wohlmuth, *A hierarchical error estimator for eddy current computation*, in Proceedings of the 2nd European Conference on Numerical Mathematics and Advanced Applications, Jyväskylä, Finland, 1999, P. Neittaanmäki and T. Tiihonen, eds., World Scientific, Singapore, 2000, pp. 110–120.

[9] A. Bermúdez, R. Durán, M. A. Muschietti, R. Rodríguez, and J. Solomin, *Finite element vibration analysis of fluid-solid systems without spurious modes*, SIAM J. Numer. Anal., 32 (1995), pp. 1280–1295.

[10] A. Bespalov, *Finite element method for the eigenmode problem of a RF cavity resonator*, Soviet J. Numer. Anal. Math. Model., 3 (1988), pp. 163–178.

[11] J. Bey, *Tetrahedral grid refinement*, Computing, 55 (1995), pp. 355–378.

[12] D. Boffi, *Discrete compactness and Fortin operator for edge elements*, Numer. Math., 87 (2000), pp. 229–246.

[13] D. Boffi, *A note on the discrete compactness property and the de Rham complex*, Appl. Math. Lett., 14 (2001), pp. 33–38.

[14] D. Boffi, F. Brezzi, and L. Gastaldi, *On the problem of spurious eigenvalues in the approximation of linear elliptic problems in mixed form*, Math. Comp., 69 (2000), pp. 121–140.

[15] D. Boffi, P. Fernandes, L. Gastaldi, and I. Perugia, *Computational models of electromagnetic resonators: Analysis of edge element approximation*, SIAM J. Numer. Anal., 36 (1999), pp. 1264–1290.

[16] A. Bossavit, *Mixed finite elements and the complex of Whitney forms*, in The Mathematics of Finite Elements and Applications VI, J. Whiteman, ed., Academic Press, London, 1988, pp. 137–144.

[17] A. Bossavit, *A rationale for edge elements in* 3*D field computations*, IEEE Trans. Mag., 24 (1988), pp. 74–79.

[18] A. Bossavit, *Whitney forms: A class of finite elements for three–dimensional computations in electromagnetism*, IEE Proc. A, 135 (1988), pp. 493–500.

[19] A. Bossavit, *A new viewpoint on mixed elements*, Meccanica, 27 (1992), pp. 3–11.

[20] J. Bramble, J. Pasciak, and A. Knyazev, *A subspace preconditioning algorithm for eigenvector/eigenvalue computation*, Adv. Comput. Math., 6 (1996), pp. 159–189.

[21] F. Brezzi and M. Fortin, *Mixed and Hybrid Finite Element Methods*, Springer-Verlag, New York, 1991.

[22] Z. Cai, J. Mandel, and S. McCormick, *Multigrid methods for nearly singular linear equations and eigenvalue problems*, SIAM J. Numer. Anal., 34 (1997), pp. 178–200.

[23] S. Caorsi, P. Fernandes, and M. Raffetto, *Approximations of Electromagnetic Eigenproblems: A General Proof of Convergence for Edge Finite Elements of Any Order of Both Nédélec's Families*, Tech. Rep. 16/99, CNR-IMA Genoa, Genoa, Italy, 1999.

[24] S. Caorsi, P. Fernandes, and M. Raffetto, *On the convergence of Galerkin finite element approximations of electromagnetic eigenproblems*, SIAM J. Numer. Anal., 38 (2000), pp. 580–607.

[25] P. Ciarlet, *The Finite Element Method for Elliptic Problems*, Stud. Math. Appl. 4, North-Holland, Amsterdam, 1978.

[26] P. Ciarlet, Jr., and J. Zou, *Fully discrete finite element approaches for time–dependent Maxwell equations*, Numer. Math., 82 (1999), pp. 193–219.

[27] P. Deuflhard, T. Friese, and F. Schmidt, *A Nonlinear Multigrid Eigenproblem Solver for the Complex Helmholtz Equation*, Tech. Rep. SC 97–55, ZIB, Berlin, Berlin, Germany, 1997.

[28] M. Dohlus, R. Schuhmann, and T. Weiland, *Calculation of frequency domain parameters using* 3*D eigensolutions*, Internat. J. Numer. Modelling, 12 (1999), pp. 41–68.

[29] E. D'yakonov, *Iteration methods in eigenvalue problems*, Math. Notes, 34 (1983), pp. 945–953.

[30] E. D'yakonov, *Optimization in Solving Elliptic Problems*, CRC Press, Boca Raton, Florida, 1996.

[31] E. D'yakonov and M. Orekhov, *Minimization of the computational labor in determining the first eigenvalues of differential operators*, Math. Notes, 27 (1980), pp. 382–391.

[32] P. Fernandes and G. Gilardi, *Magnetostatic and electrostatic problems in inhomogeneous anisotropic media with irregular boundary and mixed boundary conditions*, Math. Models Meth. Appl. Sci., 7 (1997), pp. 957–991.

[33] P. Fernandes and M. Raffetto, *Recent Developments in the (Spurious-Free) Approximation of Electromagnetic Eigenproblems by the Finite Element Method*, Tech. Rep. 06/00, CNR-IMA Genoa, Genoa, Italy, 2000.

[34] V. Girault and P. Raviart, *Finite Element Methods for Navier–Stokes Equations*, Springer-Verlag, Berlin, 1986.

[35] S. Godunov, V. Ogneva, and G. Prokopov, *On the convergence of the modified method of steepest descent in the calculation of eigenvalues*, Amer. Math. Soc. Transl. Ser. 2, 105 (1976), pp. 111–116.

[36] P. Grisvard, *Singularities in Boundary Value Problems*, Res. Notes Appl. Math. 22, Springer-Verlag, New York, 1992.

[37] P. Gross, *Efficient Finite Element-Based Algorithms for Topological Aspects of* 3*-Dimensional Magnetoquasistatic Problems*, Ph.D. thesis, College of Engineering, Boston University, Boston, MA, 1998.

[38] W. Hackbusch, *On the computation of approximate eigenvalues and eigenfunctions of elliptic operators by means of a multi-grid method*, SIAM J. Numer. Anal., 16 (1979), pp. 201–215.

[39] W. Hackbusch, *Multi-grid methods and applications*, Springer Ser. Comput. Math. 4, Springer-Verlag, Berlin, 1985.

[40] W. Hackbusch, *Theorie und Numerik elliptischer Differentialgleichungen*, B.G. Teubner–Verlag, Stuttgart, Germany, 1986.

[41] R. Hiptmair, *Multigrid method for $H(\mathrm{div})$ in three dimensions*, ETNA, 6 (1997), pp. 133–152.

[42] R. Hiptmair, *Canonical construction of finite elements*, Math. Comp., 68 (1999), pp. 1325–1346.

[43] R. Hiptmair, *Multigrid method for Maxwell's equations*, SIAM J. Numer. Anal., 36 (1998), pp. 204–225.

[44] R. Hiptmair, *Multigrid for Eddy Current Computation*, Report 154, SFB 382, Universität Tübingen, Tübingen, Germany, 2000, Math. Comp., to appear.

[45] R. Hiptmair and K. Neymeyr, *Multilevel Method for Mixed Eigenproblems*, Report 159, SFB 382, Universität Tübingen, Tübingen, Germany, 2001.

[46] F. Jochmann, *A compactness result for vector fields with divergence and curl in $L^q(\Omega)$ involving mixed boundary conditions*, Appl. Anal., 66 (1997), pp. 189–203.

[47] A. Knyazev, *Convergence rate estimates for iterative methods for a mesh symmetric eigenvalue problem*, Russian J. Numer. Anal. Math. Modelling, 2 (1987), pp. 371–396.

[48] A. Knyazev, *Preconditioned eigensolvers—an oxymoron?*, Electron. Trans. Numer. Anal., 7 (1998), pp. 104–123.

[49] A. V. Knyazev, *Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method*, SIAM J. Sci. Comput., 23 (2001), pp. 517–541.

[50] A. Knyazev and K. Neymeyr, *A geometric theory for preconditioned inverse iteration.* III: *A short and sharp convergence estimate for generalized eigenvalue problems*, Linear Algebra Appl., to appear.

[51] A. Knyazev and K. Neymeyr, *Efficient Solution of Symmetric Eigenvalue Problems Using Multigrid Preconditioners in the Locally Optimal Block Conjugate Gradient Method*, Tech. Rep. 163, Sonderforschungsbereich 382, Universität Tübingen, Tübingen, Germany, 2001, Electron. Trans. Numer. Anal., to appear.

[52] R. Kress, *Ein kombiniertes Dirichlet-Neumannsches Randwertproblem bei harmonischen Vek-*

torfeldern, Arch. Rational Mech. Anal., 42 (1971), pp. 40–49.

[53] P. Leinen, W. Lembach, and K. Neymeyr, *An Adaptive Subspace Method for Elliptic Eigenproblems with Hierarchical Basis Preconditioning*, Tech. Rep., Sonderforschungsbereich 382, Universität Tübingen, Tübingen, Germany, 1997.

[54] J. Mandel and S. McCormick, *A multilevel variational method for $Au = \lambda Bu$ on composite grids*, J. Comput. Phys., 80 (1989), pp. 442–452.

[55] W. F. Mitchell, *Optimal multilevel iterative methods for adaptive grids*, SIAM J. Sci. Statist. Comput, 13 (1992), pp. 146–167.

[56] P. Monk, *Analysis of a finite element method for Maxwell's equations*, SIAM J. Numer. Anal., 29 (1992), pp. 714–729.

[57] P. Monk and L. Demkowicz, *Discrete compactness and the approximation of Maxwell's equations in $\mathbb{R}^3$*, Math. Comp., 70 (2001), pp. 507–523.

[58] J. Nédélec, *Mixed finite elements in R^3*, Numer. Math., 35 (1980), pp. 315–341.

[59] K. Neymeyr, *A geometric theory for preconditioned inverse iteration applied to a subspace*, Math. Comp., 71 (2002), pp. 197–216.

[60] K. Neymeyr, *A Posteriori Error Estimation for Elliptic Eigenproblems.* Report 132, Sonderforschungsbereich 382, Universität Tübingen, Tübingen, Germany, 1999, J. Numer. Linear Algebra Appl., to appear.

[61] K. Neymeyr, *A geometric theory for preconditioned inverse iteration.* I: *Extrema of the Rayleigh quotient*, Linear Algebra Appl., 322 (2001), pp. 61–85.

[62] K. Neymeyr, *A geometric theory for preconditioned inverse iteration.* II: *Convergence estimates*, Linear Algebra Appl., 322 (2001), pp. 87–104.

[63] E. Ovtchinnikov and L. Xanthis, *Successive eigenvalue relaxation: A new method for the generalized eigenvalue problem and convergence estimates*, Proc. Roy. Soc. London Ser. A, 457 (2001), pp. 441–451.

[64] B. Parlett, *The Symmetric Eigenvalue Problem*, Prentice Hall, Englewood Cliffs, NJ, 1980.

[65] W. Petryshyn, *On the eigenvalue problem $Tu - \lambda Su = 0$ with unbounded and non-symmetric operators T and S*, Philos. Trans. Roy. Soc. Math. Phys. Sci., 262 (1968), pp. 413–458.

[66] B. Samokish, *The steepest descent method for an eigenvalue problem with semi-bounded operators*, Izv. Vyssh. Uchebn. Zaved. Mat., 5 (1958), pp. 105–114 (in Russian).

[67] F. Schmidt, T. Friese, L. Zschiederich, and P. Deuflhard, *Adaptive Multigrid Methods for the Vectorial Maxwell Eigenvalue Problem for Optical Waveguide Design*, Tech. Rep. 00-54, ZIB Berlin, Berlin, Germany, 2000.

[68] D. White and J. Koning, *Efficient Solution of Large-Scale Electromagnetic Eigenvalue Problems Using the Implicitly Restarted Arnoldi Method*, Preprint UCRL-JC-129188, Lawrence Livermore National Laboratory, Livermore, CA, 1997.

SIAM J. Sci. Comput.
Vol. 23, No. 6, pp. 2165–2182

A BLOCK ORTHOGONALIZATION PROCEDURE WITH CONSTANT SYNCHRONIZATION REQUIREMENTS*

ANDREAS STATHOPOULOS† AND KESHENG WU‡

Abstract. First, we consider the problem of orthonormalizing skinny (long) matrices. We propose an alternative orthonormalization method that computes the orthonormal basis from the right singular vectors of a matrix. Its advantages are that (a) all operations are matrix-matrix multiplications and thus cache efficient, (b) only one synchronization point is required in parallel implementations, and (c) it is typically more stable than classical Gram–Schmidt (GS). Second, we consider the problem of orthonormalizing a block of vectors against a previously orthonormal set of vectors and among itself. We solve this problem by alternating iteratively between a phase of GS and a phase of the new method. We provide error analysis and use it to derive bounds on how accurately the two successive orthonormalization phases should be performed to minimize total work performed. Our experiments confirm the favorable numerical behavior of the new method and its effectiveness on modern parallel computers.

Key words. Gram–Schmidt, orthogonalization, Householder, QR factorization, singular value decomposition, Poincaré

AMS subject classification. 65F15

PII. S1064827500370883

1. Introduction.

Computing an orthonormal basis from a given set of vectors is a basic computation, common to most scientific applications. Often, it is also one of the most computationally demanding procedures because the vectors are of large dimension, and because the computation scales as the square of the number of vectors involved. Further, among several orthonormalization techniques the ones that ensure high accuracy are the more expensive ones.

Skinny (or long) matrices, whose row dimension far exceeds their column dimension, arise naturally in various scientific contexts. Examples include statistical analysis, where there are many more observations than variables, and iterative methods that use a small subspace of vectors to span the required solutions. For a variety of reasons, an orthonormal basis of these vectors must be computed. Traditionally, this is obtained through the QR factorization, even though quite often the matrix R is not of primary interest, but rather the orthonormal basis Q. The QR factorization is computed through Householder transformations (we call this method simply QR) or through classical Gram–Schmidt (GS) or modified Gram–Schmidt (MGS). Although less stable numerically, GS with reorthogonalization is usually preferred to the QR method because of better computational properties.

Yet, all of these methods have performance limitations on modern, cache based processors and parallel computers. Their implementations are based on level 1 or level 2 BLAS operations [8, 9, 15], which have low cache reuse. Level 3 BLAS implementations are possible, but they are not suitable for skinny matrices. On parallel

*Received by the editors April 17, 2000; accepted for publication (in revised form) December 4, 2001; published electronically May 15, 2002. This research was supported by NERSC and performed using computational facilities at NERSC and the College of William and Mary. The latter facility was supported by grants from the National Science Foundation (EIA-9977030) and Sun Microsystems (SAR EDU00-03-793).

http://www.siam.org/journals/sisc/23-6/37088.html

†Department of Computer Science, College of William and Mary, Williamsburg, VA 23187-8795 (andreas@cs.wm.edu).

‡NERSC, Lawrence Berkeley National Laboratory, Berkeley, CA 94720 (kwu@lbl.gov).

platforms, such as the increasingly popular clusters of workstations, reduction in communication and synchronization overheads has not kept up with the explosive growth of network bandwidth and processor speed [21]. As a result, the global synchronization required by frequent inner products does not scale with the number of processors. A method is still needed that operates only on blocks of vectors and requires a number of synchronizations that is independent of the size of the matrix.

In many applications, orthonormalization occurs in an incremental fashion, where a new set of vectors (we call this *internal set*) is orthogonalized against a previously orthonormal set of vectors (we call this *external*), and then among themselves. This computation is typical in block Krylov methods, where the Krylov basis is expanded by a block of vectors [11, 12]. It is also typical when certain external orthogonalization constraints have to be applied to the vectors of an iterative method. Locking of converged eigenvectors in eigenvalue iterative methods is such an example [19, 22]. The nature of these applications suggests that the internal set is usually a skinny matrix with fewer vectors than the external set.

Conceptually, this problem can be viewed as an update of a QR factorization that has already produced the orthonormal set of external vectors which should not be modified. Computationally, however, the problem is usually tackled as a two phase process; first, orthogonalizing the internal vectors against the external ones (external phase), and second, orthogonalizing the internal vectors among themselves (internal phase). For the external phase, block GS and MGS are the most competitive choices, while for the internal phase an efficient orthonormalization procedure for skinny matrices is needed.

Most of the previous efforts to address the above two problems considered blocks of vectors, and used hybrids of the more scalable GS across blocks, and the more accurate MGS within blocks [2, 14]. Performance improves, but the number of synchronization points is still linear to the number of vectors, and BLAS level 2 kernels are still dominant despite blocking. Interestingly, such efforts have focused on a full QR factorization of a set of vectors, rather than on the two phase problem.

In this paper, we introduce a method based on the singular value decomposition (SVD) that uses the right singular vectors to produce an orthonormal basis for a given skinny matrix. The idea itself is not new, dating back at least to Poincaré, and it is sometimes encountered in chemistry and wavelet literature [5, 6, 16, 17, 20]. However, it has not received any attention as a computationally viable orthogonalization method, and to our knowledge there is no analysis of its numerical properties. The method, which we call SVQB, uses exclusively level 3 BLAS kernels, and it has a constant number of synchronization points. We show that it is not as accurate numerically as MGS, but it is better than GS in the absence of special sparsity structure. More interestingly, we show that more stable alternatives, such as MGS or Householder, are an overkill for our two phase problem. Coupling the SVQB method for the internal phase with a block GS with reorthogonalization for the external phase results in a method also with constant synchronization requirements.

The paper is organized as follows. First we describe the SVQB method for skinny matrices, we analyze its numerical stability, and we confirm our theory through numerical experiments and comparisons with other methods. Second, we couple SVQB with a block GS for the two phase problem. We analyze the numerical interaction between the two methods, and based on this theory we tune the two phases to avoid unnecessary reorthogonalizations. Following this, we present timings from a series of experiments on the Cray T3E, IBM SP-2, and on a cluster of SUN workstations.

These verify that the block computations and the small number of synchronizations help the new method achieve accurate orthogonality faster than other competitive methods.

2. The problem(s). Let $V \in \Re^{n\times k}$ be a set of orthonormal vectors, and $W \in \Re^{n\times m}$ be a set of vectors, where $k + m \leq n$. In practice, we expect $m < k$ and $m \ll n$. When $m \ll n$, W is often referred to as a skinny matrix. The first problem we consider is that of obtaining an orthonormal set of vectors Q such that span(Q)=span(W). The second problem is again to obtain an orthonormal Q such that span([V W])=span([V Q]) and $Q \perp V$. In both problems Q can be any orthonormal basis, not necessarily from a QR factorization. In finite precision, the equality of the spans can be relaxed, but the orthonormality requirement must remain.

The distinction between the two problems is made for computational reasons. First, orthonormalizing skinny matrices is a problem important in itself, which allows for efficient solutions without a QR factorization. In the presence of an external matrix V, methods like the classical GS would orthogonalize each vector of W against all previous orthogonal vectors in both W and V. In that case, the difference between the two phases is blurred. However, such algorithms allow only for level 2 BLAS computational kernels and introduce at least $O(m)$ number of synchronization points on parallel computers. To improve computational performance we need to consider W as a block (or subblocks within W). A block GS method would orthogonalize the block against V (and other previous subblocks in W). In this case, the orthogonality among the vectors within the block must be resolved at a different time in a distinct internal phase. Finally, because V cannot be modified, distinguishing between the problems allows non-QR factorization methods to be used for the internal phase.

For the internal phase, GS and MGS are popular QR factorization methods which both incur the same number of arithmetic operations, they are based on level 2 BLAS kernels, and they can be implemented in parallel with a modest number of $m + 1$ synchronization points [11, 24]. MGS is more numerically stable with the error in the orthogonality of Q bounded by $\epsilon\kappa(W)$, where $\kappa(W)$ is the condition number of W [1, 3]. Householder reflections yield a matrix Q which is orthogonal to machine precision (ϵ) but require twice the arithmetic of (M)GS. In practice, for most matrices, a second orthogonalization with GS is typically enough for producing orthogonality to machine precision [7], and thus GS is often preferred over other methods.

In the context of the two phase problem, producing an internal set of vectors Q with orthogonality close to machine precision is unnecessary because of the interdependence of the phases. For example, external orthogonalization against V may spoil the internal orthogonality of W, and vice versa. Therefore, this two phase problem obviates the use of expensive but stable methods such as Householder.

3. The SVQB method. An especially interesting orthonormal basis of the span(W) is the one derived from the right singular vectors of W. Assume that the vectors in W are normalized. The singular values of W are the square roots of the eigenvalues of $S = W^TW$, and the right singular vectors are the corresponding eigenvectors of S. Let $SU = U\Lambda$ be the eigendecomposition of S and define $Q = WU\Lambda^{-1/2}$. Obviously, span(Q) = span(W) and $Q^TQ = I$. If the W vectors are not normalized, the diagonal of S, $D = \text{diag}(S)$, contains the squares of their norms ($S_{ii} = W_i^TW_i$). Therefore, we can implicitly work with the normalized $WD^{-1/2}$ by scaling the columns and rows of S. This is inexpensive and is as numerically stable as explicit normalization. The resulting factorization is not a QR but rather a "QB" factorization, where Q is orthonormal and B a full matrix. In exact arithmetic, the algorithm for this singular vector QB factorization (which we call SVQB) follows.

Algorithm 3.1. $Q = \mathrm{SVQB}(W)$.

```
1. $S' = W^T W$.
2. Scale $S = D^{-1/2} S' D^{-1/2}$, with $D = diag(S')$.
3. Solve $SU = U\Lambda$, for all eigenpairs.
4. Compute $Q = WD^{-1/2} U \Lambda^{-1/2}$.
```

When some of the vectors in W are linearly dependent, one or more of the eigenvalues and their corresponding vectors in Q are zero. In finite precision, a similar effect is caused by almost linearly dependent vectors and eigenvalues close to zero. Because of numerical noise, such eigenvalues cannot be bounded away from zero. To prevent normalization overflows, and to avoid an explicit computation of the norm of the vectors Q, we set a minimum threshold for eigenvalues. If ϵ is the machine precision, we insert the following two steps:

```
    3.1. $\tau = \epsilon \ \max_i(\Lambda_{ii})$.
    3.2. If $\Lambda_{ii} < \tau$, set $\Lambda_{ii} = \tau$ for all $i$.
```

Other strategies for dealing with linear dependencies are also possible. For example, we could consider only those eigenvectors with eigenvalues greater than some "safe" threshold. The resulting basis is then smaller, but it is guaranteed to be orthonormal and to numerically span a subspace of the original vectors. Finally, because of finite precision arithmetic, the algorithm may have to be applied iteratively ($Q^{(i+1)} = \mathrm{SVQB}(Q^{(i)})$) until an orthonormal set Q is obtained.

The solution of the eigenvalue problem and the implicit normalization involve only $m \times m$ matrices (S and U), and thus they are inexpensive. On parallel computers these can be duplicated on each processor. The matrix-matrix multiplication for computing S and the multiplication of W with U each contribute $2nm^2$ floating point operations, which makes the algorithm twice as expensive as GS. However, these operations are level 3 BLAS kernels and can be performed efficiently on cache based computers. Alternatively, the matrix multiplication for computing the symmetric S can be performed with half the operations, but level 2 BLAS kernels will have to be used. Moreover, a parallel implementation of the SVQB method requires only one synchronization point when computing the matrix S.

We note that our interest in the singular vectors stems only from the computational efficiency of SVQB. A standard bidiagonalization kernel for computing the SVD of W directly could offer better numerical stability [4]. However, its cache utilization and parallel efficiency is similar to that of QR (with Householder transformations), without yielding the same level of machine precision orthogonality. We have not considered such methods, as they offer no advantages over the QR method.

3.1. The Cholesky QR method.

Similarly to the SVQB method, we can derive a block QR factorization based on the Cholesky factorization. Note that if $S = W^T W = R^T R$, where R is the Cholesky factor, $Q = WR^{-1}$ defines the QR factorization for W [11]. Although this method (denoted as CholQR) is rarely used computationally, it has some attractive characteristics: it is a QR factorization; it is based on a level 3 BLAS kernel and a triangular system solution; it involves only 50% more arithmetic than GS; and it also requires one synchronization point in parallel implementations. Researchers have noticed that it is not as stable as MGS, but it is often more stable than GS [2, 10]. One of the problematic issues with CholQR is that the more ill conditioned S is, the less stable the Cholesky factorization becomes. Regularizing it effectively is not as straightforward as in the case of SVQB, where the smallest singular pairs can simply be left out of the computation. Finally, the cache performance of CholQR is usually inferior to that of SVQB because of the triangular solve.

4. Stability analysis of SVQB. For many applications, such as block Krylov iterative methods, the orthonormality of the resulting vectors Q, not the upper triangular R of the QR factorization, is of importance. For example, Krylov methods will still make progress, albeit a slower one, if provided with a slightly different orthonormal set Q. Thus, as it is common in the literature, we measure stability as the departure of the resulting Q from orthonormality rather than its backward error. Because of its block, nonsequential nature, we expect the SVQB procedure to be less stable than MGS. However, as we show below, it is often more stable than GS.

THEOREM 4.1. *Let W be a set of m linearly independent vectors of $\Re^n$. Let $\bar{Q}$ be the floating point representation of the matrix computed by applying the SVQB procedure on W. If $\kappa(W)$ is the condition number of W, then*

$$\|I - \bar{Q}^T\bar{Q}\| \;\leq\; c_0 \;\min\big(\, \epsilon \;\kappa(W)^2,\; 1\big)\,,$$

where $\|.\|$ denotes the 2-norm, ϵ is the machine round off, and c_0 is a constant depending on n and m.

Proof. Let $S = W^TW$, and let $\tilde{S}$ be the floating point representation of S. Then

$$\tilde{S} = S + \delta S, \;\text{ with }\; \|\delta S\| \leq c_1 \;\epsilon\; \|S\|. \tag{4.1}$$

We can further write this as $\|\delta S\| \leq c_1 \;\epsilon\; \|W\|^2$. The effect of performing scaling on the matrix S corresponds to each vector in W having norm 1 implicitly, and, in that case, $\|\delta S\| \leq c_1 \;\epsilon\; m$.

Let $\bar{U}$, with $\bar{U}^T\bar{U} = I$, and $\bar{\Lambda}$ be the computed eigenvectors and eigenvalues of the small $m \times m$ symmetric matrix $\tilde{S}$. From standard backward error analysis, these can be considered an exact eigendecomposition of a nearby matrix $\bar{S} = \bar{U}\bar{\Lambda}\bar{U}^T$. Using relation (4.1) we can express the error in $\bar{S}$ as

$$\bar{S} = \tilde{S} + \delta\tilde{S}, \;\text{ with }\; \|\delta\tilde{S}\| \leq c_2 \;\epsilon\; \|S\| + O(\epsilon^2). \tag{4.2}$$

From the above, and by letting $c_3 = c_1 + c_2$, the matrices S and $\bar{S}$ are related by

$$\|\bar{S} - S\| = \|\delta\tilde{S} + \delta S\| \leq c_3 \;\epsilon\; \|S\|. \tag{4.3}$$

Let $\bar{\lambda}_{\min} = \min_i \bar{\Lambda}_{ii}$, and $\bar{\lambda}_{\max} = \max_i \bar{\Lambda}_{ii}$, and consider a similar notation for eigenvalues of other matrices. Because our algorithm sets eigenvalues $\bar{\lambda}_i$ that are smaller than $\epsilon\bar{\lambda}_{\max}$ equal to this threshold, we define a diagonal matrix $\hat{\Lambda}$ such that

$$\hat{\Lambda}_{ii} = \begin{cases} \bar{\Lambda}_{ii} & \text{if } \bar{\Lambda}_{ii} > \epsilon\bar{\lambda}_{\max}, \\ \epsilon\bar{\lambda}_{\max}, & \text{if } \bar{\Lambda}_{ii} \leq \epsilon\bar{\lambda}_{\max}. \end{cases} \tag{4.4}$$

Let $\bar{Q} = Q + \delta Q = W\bar{U}\hat{\Lambda}^{-1/2} + \delta Q$ be the floating point representation of the matrix returned by the SVQB procedure. Then, $\|\delta Q\| \;\leq c_4 \,\epsilon\, \|W\|\; \|\bar{U}\|\; \|\hat{\Lambda}^{-1/2}\|$. If we denote by λ_i the exact eigenvalues of S, with $\lambda_{\max}$ the largest one, then $\|W\| = \sqrt{\lambda_{\max}}$. Note also that $\bar{U}^T\bar{U} = I$, and thus $\|\bar{U}\| = 1$. From (4.4) we have

$$\|\hat{\Lambda}^{-1}\| \leq \;\min\left(\frac{1}{\bar{\lambda}_{\min}}, \frac{1}{\epsilon\bar{\lambda}_{\max}}\right). \tag{4.5}$$

Because for symmetric eigenproblems the error in the eigenvalue is bounded by the error in the matrix, for any $\bar{\lambda}_i$ we have $\left|\bar{\lambda}_i - \lambda_i\right| \le \|\bar{S} - S\| \le c_3\,\epsilon\,\|S\| = c_3\,\epsilon\,\lambda_{\max}$. Then, there are constants c_5 and c_6, such that

$$\bar{\lambda}_{\min} = \lambda_{\min} + c_5\,\epsilon\,\lambda_{\max} \quad \text{and} \quad \bar{\lambda}_{\max} = \lambda_{\max} + c_6\,\epsilon\,\lambda_{\max}. \tag{4.6}$$

We note that $\kappa(W) = \sqrt{\kappa(S)} = \sqrt{\lambda_{\max}/\lambda_{\min}}$, and, because $\epsilon + c_6\,\epsilon^2 = O(\epsilon)$, a substitution of (4.6) into (4.5) yields

$$\|\hat{\Lambda}^{-1}\| \le \min\left(\frac{1/\lambda_{\min}}{1 + c_5\,\epsilon\kappa(S)},\ \frac{1}{\epsilon\lambda_{\max}}\right). \tag{4.7}$$

The first term is chosen as the minimum if none of the $\bar{\Lambda}_{ii}$ is in the order of $\epsilon\bar{\lambda}_{\max}$ or smaller. This means that all the singular values of W, $\sqrt{\lambda_i}$, must be larger than $\sqrt{\epsilon}$, because they can be represented by eigenvalues of $\bar{S}$, despite the squaring. Therefore, $\kappa(S) < O(1/\epsilon)$, and $1 + c_5\,\epsilon\kappa(S) = O(1)$. Thus, the bound (4.7) becomes

$$\|\hat{\Lambda}^{-1}\| \le c_7\ \min\left(\frac{1}{\lambda_{\min}},\ \frac{1}{\epsilon\lambda_{\max}}\right). \tag{4.8}$$

By setting $c_8 = c_4 c_7$, we can give a bound for $\|\delta Q\|$, as well as for $\|Q\| = \|W\bar{U}\hat{\Lambda}^{-1/2}\|$:

$$\|\delta Q\| \le c_8\ \min\left(\ \epsilon\ \kappa(W),\ \sqrt{\epsilon}\ \right), \tag{4.9}$$

$$\|Q\| \le c_8\ \min\left(\ \kappa(W),\ 1/\sqrt{\epsilon}\ \right). \tag{4.10}$$

We emphasize that the above is not the backward error for the exact result of SVQB but for the product of computed matrices that yields $\bar{Q}$. The exact backward error is not relevant because the orthogonality of $\bar{Q}$ is of interest.

Let us consider the departure of the computed $\bar{Q} = Q + \delta Q$ from orthonormality:

$$\begin{aligned} \|I - \bar{Q}^T\bar{Q}\| &= \|I - \hat{\Lambda}^{-1/2}\bar{U}^T W^T W \bar{U}\hat{\Lambda}^{-1/2} + \delta Q^T Q + Q^T \delta Q\| + O(\delta Q^2) \\ &\le \|I - \hat{\Lambda}^{-1/2}\bar{U}^T S \bar{U}\hat{\Lambda}^{-1/2}\| + 2\|\delta Q\|\|Q\|. \end{aligned} \tag{4.11}$$

From relations (4.1)–(4.4) and the orthonormality of $\bar{U}$, this becomes

$$\begin{aligned} \|I - \bar{Q}^T\bar{Q}\| &\le \|I - \hat{\Lambda}^{-1/2}\bar{U}^T\bar{S}\bar{U}\hat{\Lambda}^{-1/2} + \hat{\Lambda}^{-1/2}\bar{U}^T(\delta\tilde{S} + \delta S)\bar{U}\hat{\Lambda}^{-1/2}\| + 2\|\delta Q\|\|Q\| \\ &\le \|I - \hat{\Lambda}^{-1/2}\bar{\Lambda}\hat{\Lambda}^{-1/2}\| + \|\hat{\Lambda}^{-1}\|\|\delta\tilde{S} + \delta S\| + 2\|\delta Q\|\|Q\|. \end{aligned} \tag{4.12}$$

From definition (4.4), the first term is zero if $\kappa(S) < O(1/\epsilon)$. Otherwise, it is equal to $\max(1 - \bar{\lambda}_i/\epsilon\bar{\lambda}_{\max},$ for $\bar{\lambda}_i \le \epsilon\bar{\lambda}_{\max})$. However, this is less than one, and in that case the other terms are also $O(1)$. From bounds (4.3) and (4.8)–(4.10), and by setting $c_0 > c_3 c_7 + 2c_8^2$, we obtain

$$\begin{aligned} \|I - \bar{Q}^T\bar{Q}\| &\le c_3 c_7\ \min\left(\ \epsilon\kappa(W)^2,\ 1\right) + 2c_8^2 \min\left(\ \epsilon\kappa(W)^2,\ 1\right) \\ &\le c_0\ \min\left(\ \epsilon\ \kappa(W)^2,\ 1\right). \qquad \square \end{aligned} \tag{4.13}$$

Next, we bound $|\kappa(\bar{Q}) - 1|$, thus showing that when applying SVQB iteratively, $\bar{Q}$ converges fast to an orthonormal basis. We first state the following lemma (see [13]).

Lemma 4.2.

1. *If* $\|I - Q^T Q\| \le \alpha$, *then* $\|Q^T Q\| \le \|Q\|^2 \le 1 + \alpha$.
2. *If* $\|I - Q^T Q\| \le \alpha < 1$, *then* $\|I - (Q^T Q)^{-1}\| \le \frac{\alpha}{1-\alpha}$.

PROPOSITION 4.3. *If $\|I - Q^TQ\| \leq \alpha < 1$, then the condition number $\kappa(Q)$ satisfies*

$$\kappa(Q) \leq \sqrt{\frac{1+\alpha}{1-\alpha}}.$$

Proof. By definition, $\kappa(Q) = \sqrt{\|Q^TQ\|\|(Q^TQ)^{-1}\|}$. The proof follows from Lemma 4.2, since $\|Q^TQ\| \leq \|Q\|^2 \leq 1 + \alpha$, and $\|(Q^TQ)^{-1}\| = \|I - I + (Q^TQ)^{-1}\| \leq 1 + \|I - (Q^TQ)^{-1}\| \leq \frac{1}{1-\alpha}$. □

THEOREM 4.4. *Let W be a set of m linearly independent vectors of $\Re^n$, with condition number $\kappa(W)$. Let $\bar{Q}$ be the floating point representation of the matrix computed by applying the SVQB procedure on W. If ϵ is the machine round off, then*

$$\text{if } \sqrt{\epsilon}\ \kappa(W) < c < 1, \qquad \kappa(\bar{Q}) \leq 1 + O(\epsilon\ \kappa(W)^2)).$$

Proof. If we let $\alpha = O(\epsilon\ \kappa(W)^2) < c^2 < c < 1$, according to Theorem 4.1, $\|I - \bar{Q}^T\bar{Q}\| < \alpha$, and by using Proposition 4.3, we have $\kappa(\bar{Q}) \leq \sqrt{\frac{1+\alpha}{1-\alpha}} \leq \sqrt{1 + \frac{2\alpha}{1-\alpha}} \leq 1 + \frac{\alpha}{1-\alpha}$. This proves the bound because, in this case, α is bounded away from 1. □

The case where $\kappa(W) \geq \frac{1}{\sqrt{\epsilon}}$ cannot be bounded in the general case because numerical error dominates. However, some intuition can be gained by considering the structure of $\bar{Q}^T\bar{Q}$ as obtained from (4.12) and the bounds (4.9)–(4.10) and (4.3):

$$\bar{Q}^T\bar{Q} = \hat{\Lambda}^{-1/2}\bar{\Lambda}\hat{\Lambda}^{-1/2} + \hat{\Lambda}^{-1/2}\Delta S\hat{\Lambda}^{-1/2} + \Delta Q,$$

where $\Delta S = O(\epsilon\bar{\lambda}_{\max})$ and $\Delta Q = O(1)$ element-wise. Note that after scaling, $\hat{\Lambda}^{-1/2}\Delta S\hat{\Lambda}^{-1/2}$ also becomes element-wise $O(1)$. From the definition of $\hat{\Lambda}$, the entries of the diagonal matrix $\hat{\Lambda}^{-1/2}\bar{\Lambda}\hat{\Lambda}^{-1/2}$ are 1 for all eigenvalues above the $\epsilon\bar{\lambda}_{\max}$ threshold, and $\bar{\lambda}_i/(\epsilon\bar{\lambda}_{\max}) = 1/(\epsilon\kappa(\bar{S}))$ for the rest. Thus, the eigenvalues of $\bar{Q}^T\bar{Q}$ are O(1) perturbations of the these diagonal values, so the smallest eigenvalue cannot be bounded away from zero. If we assume that the only finite precision errors occur from the inability to represent eigenvalues of $\bar{S}$ smaller than $\epsilon\lambda_{\max}$, i.e., $\Delta S = 0$ and $\Delta Q = 0$, then, obviously, $\kappa(\bar{Q}^T\bar{Q}) = \epsilon\kappa(\bar{S}) < \epsilon\kappa(S)$, and thus $\kappa(\bar{Q}) < \sqrt{\epsilon}\kappa(W)$. Note that in this case the vectors of $\bar{Q}$ are exactly orthogonal to each other, yet their condition number is far from 1.

Although $\kappa(\bar{Q}) < \sqrt{\epsilon}\kappa(W)$ cannot be proved in general, we have observed it in all our numerical experiments. A plausible explanation is that $O(1)$ perturbations after vector scaling introduce random noise which we expect to be in linearly independent and relatively well-conditioned directions.

4.1. Convergence comparisons. The above theorems suggest that in most situations, applying SVQB once or twice should produce orthogonal vectors. In the case of extremely ill-conditioned vectors, a third application of the procedure might be necessary. This is akin to the behavior of GS with reorthogonalization [7, 13, 18], but it is expected to be better than iterative GS without internal reorthogonalization.

If an accurate Cholesky decomposition can be computed, the CholQR procedure should be identical to SVQB. In fact, it might be possible to prove bounds for CholQR similar to the ones in the previous section. However, even with an accurate decomposition, for very large condition numbers we expect CholQR to be less stable than the eigenvalue-based SVQB method.

Table 4.1

W = Krylov(A, 30), where A is a 2-D Laplacean of size 1089×1089, and an initial vector of all ones. $\kappa(W) = 3e + 20$. However, after scaling, because of numerical error, $\kappa(W)$ becomes: $6e + 16$.

	SVQB			CholQR	GS	MGS
Iteration	$\kappa(Q_u)$	$\kappa(\bar{Q})$	$\kappa(Q'_u)$	$\kappa(\bar{Q})$	$\kappa(\bar{Q})$	$\kappa(\bar{Q})$
1	6e+16	1e+09	4e+08	1e+09	2e+16	2e+01
2	4e+08	1e+01	4e+00	6e+01	1e+14	3e-14
3	4e+00	$1+\epsilon$	$1+\epsilon$	1+7e-12	8e+10	$1+\epsilon$
4	–	–	–	$1+\epsilon$	3e+06	–
5	–	–	–	–	1+1e−03	–
6	–	–	–	–	$1+\epsilon$	–

Table 4.2

W = Hilbert matrix of size(100). $\kappa(W) = 2e + 19$.

	SVQB			CholQR	GS	MGS
Iteration	$\kappa(Q_u)$	$\kappa(\bar{Q})$	$\kappa(Q'_u)$	$\kappa(\bar{Q})$	$\kappa(\bar{Q})$	$\kappa(\bar{Q})$
1	2e+19	3e+11	9e+10	2e+12	2e+19	7e+02
2	9e+10	2e+03	7e+02	6e+04	4e+16	1+2e-13
3	8e+02	1+5e-11	1+2e-11	1+2e-07	2e+14	$1+\epsilon$
4	1+2e-11	$1+\epsilon$	$1+\epsilon$	$1+\epsilon$	4e+12	–
5	–	–	–	–	4e+10	–
6	–	–	–	–	4e+08	–
7	–	–	–	–	2e+00	–
8	–	–	–	–	$1+\epsilon$	–

Table 4.3

*W = [ones(1, 30), diag(rand(30, 1)*eps*eps*eps)].*

	SVQB			CholQR	GS	MGS
Iteration	$\kappa(Q_u)$	$\kappa(\bar{Q})$	$\kappa(Q'_u)$	$\kappa(\bar{Q})$	$\kappa(\bar{Q})$	$\kappa(\bar{Q})$
1	2e+49	3e+41	4e+34	4e+34	2e+01	$1+\epsilon$
2	4e+34	7e+25	4e+19	4e+19	$1+\epsilon$	–
3	4e+19	3e+11	2e+07	4e+06	–	–
4	2e+07	1+1e−03	1+1e−04	1+4e−03	–	–
5	1+1e−04	$1+\epsilon$	$1+\epsilon$	$1+\epsilon$	–	–

To demonstrate the relative effectiveness of these methods, we apply them on three sets of vectors and report the improvements on their condition numbers. The first set is the 30 Krylov vectors generated from a vector of all ones and the two-dimensional (2-D) Laplacean on a regular, finite difference, square mesh with Neumann conditions. The dimension of the matrix is 1089, and the initial vector is not considered among the set of 30. The second set consists of the columns of the Hilbert matrix of size 100. The third set is rather artificial, and it has been used to show the benefits of MGS over GS [2, 13, 14]. We use the following variation, shown in MATLAB notation: `W = [ ones(1,30); diag( rand(30,1)*eps*eps*eps ) ]` . All tests are run in MATLAB on a SUN Ultra-2 workstation with $\epsilon = 2.2\text{e}{-16}$. The condition numbers are computed by the Matlab function `cond` and therefore could be inaccurate whenever they exceed 10^{16}. The results for these three cases are shown in Tables 4.1, 4.2, and 4.3, respectively.

We compare SVQB against CholQR, GS, and MGS by printing the condition number $\kappa(\bar{Q})$ of the vectors that these methods produce after each iteration. Since the implicit normalization in step 4 of SVQB does not guarantee normality for ill-conditioned problems, we also print the condition number $\kappa(Q_u)$ of the unscaled vectors, $Q_u = W\bar{U}$, and the condition number of the same vectors after explicitly scaling them by their norms, $\kappa(Q'_u)$. Note that at any iteration, $\kappa(Q_u)$ is equal to $\kappa(Q'_u)$ of the previous iteration. For example, after the first iteration $\kappa(Q_u) = \kappa(W)$. This verifies that explicit normalization is not needed. If scaling by $\bar{\Lambda}^{-1/2}$ does not produce normal vectors, another iteration must be performed either way, during which the implicit normalization of columns and rows of S in step 2 of SVQB has the same effect as explicit normalization. In addition, explicit normalization would introduce additional work, and, more importantly, an additional synchronization point.

The results in all tables confirm the developed theory. When the condition number is smaller than $1/\sqrt{\epsilon}$, the reduction obeys closely the bound in Theorem 4.4. The application of one step of SVQB reduces the condition number of a set of vectors by at least $\sqrt{\epsilon}$. This reduction is sharp for the examples in Tables 4.1 and 4.2, but if the vectors are explicitly normalized the reduction could be larger (see Table 4.3).

As expected, the CholQR method behaves similarly to SVQB (Table 4.3). In some cases, the orthogonality of CholQR is inferior to that of SVQB (see Table 4.2), and thus it is possible that it takes more iterations to produce a fully orthonormal set (see Table 4.1). This is in spite of the fact that in our implementation, we first compute the lowest eigenvalue of W^TW and shift it so that the Cholesky decomposition is applied on a numerically positive definite matrix.

As discussed earlier, GS without reorthogonalization for each vector is not effective for large condition numbers. In such cases, GS may offer no improvement between successive iterations (see the first GS iteration in Table 4.2), or it may require many iterations to produce a set with a relatively small conditioner number (see Tables 4.1 and 4.2). Once this is achieved, however, one or two further iterations provide a fully orthonormal set. An exception is the example in Table 4.3 for which GS requires only one reorthogonalization. The reason is that GS takes advantage of the sparse structure of the matrix, performing computations only among very small elements, thus achieving low *relative* error.

MGS is clearly more stable than the rest of the methods. However, because the departure from orthogonality for MGS is bounded by $\epsilon\kappa(W)$ [1], even for relatively small $\kappa(W)$, a second MGS is often needed (see Tables 4.1 and 4.2). Note that the $\epsilon\ \kappa(W)^2$ bound for SVQB is virtually identical to that of MGS, if $\kappa(W)$ is close to 1, thus diminishing any advantages over SVQB.

5. The two phase problem.

The above theory and examples establish that SVQB is a competitive choice for the internal orthonormalization of the two phase problem. If we denote as $W' = \text{Ortho}(V, W)$ any orthonormalization procedure for the external phase, $W' = (I - VV^T)WD^{-1/2}$, where $D^{1/2}$ is a diagonal matrix with the normalizing norms of the vectors, the two phase algorithm can be described as follows.

Algorithm 5.1. $Q = \text{Ortho-SVQB}(V, W)$.

```
1. $W' = \text{Ortho}(V, W)$.
2. $Q = \text{SVQB}(W')$.
```

The most common choices for Ortho() are GS and MGS. Because efficiency is important, especially when the number of vectors in V, k, is large, block GS with some form of reorthogonalization is usually employed. As we show below, accuracy close to machine precision is less critical because the orthogonality achieved in one phase may not be preserved in the other.

In finite precision, the above algorithm does not always compute an accurately orthonormal set Q, and thus it has to be applied in an iterative fashion. To develop an efficient iterative version of Ortho-SVQB, we must first examine the numerical interaction between the two phases.

5.1. Numerical interplay of the phases. The reason that full orthogonality is not always necessary in each phase is that SVQB procedure may destroy previous orthogonality against V, and Ortho(V, W) may destroy the orthogonality in W.

LEMMA 5.1. *Let W be a set of vectors with $\|V^TW\| \leq \mu\ \|W\|$. Let $\bar{Q} =$ SVQB(W) be the result of the internal step of the Ortho-SVQB algorithm. Then,*

$$\|V^T\bar{Q}\| = O\left((\mu+\epsilon)\ \min\left(\kappa(W), \frac{1}{\sqrt{\epsilon}}\right)\right).$$

Proof. Following the notation of Theorem 4.1, let $\bar{Q} = W\bar{U}\bar{\Lambda}^{-1/2} + \delta Q$. Using bounds (4.8) and (4.9) we have $\|V^T\bar{Q}\| = \|V^TW\bar{U}\bar{\Lambda}^{-1/2} + V^T\delta Q\| \leq \mu\|W\|\|\bar{\Lambda}^{-1/2}\| + O(\|\delta Q\|) = O\left((\mu+\epsilon)\min(\kappa(W), 1/\sqrt{\epsilon})\right).$ □

The lemma states that even when W is exactly orthogonal to V, i.e., $\mu = 0$, the SVQB procedure at the next step may destroy that orthogonality up to a maximum of $\sqrt{\epsilon}$. For example, consider the following matrices:

$$V = \begin{bmatrix} 0.17164335073404 \\ 0.00000000003278 \\ -0.17164335076682 \end{bmatrix} \text{ and } W = \begin{bmatrix} 1 & 1 \\ 1 & 1+10^{-6} \\ 1 & 1 \end{bmatrix}.$$

A Matlab computation shows that $\|V^TW\| =$ 3.2e-17, and $\kappa(W) =$ 4.2e+6. After the step $Q =$ SVQB(W), we observe that $\|V^TQ\| =$ 4e-11, which agrees with our lemma. Therefore, it is not important to choose one of the more accurate Ortho() methods, such as MGS, to obtain good orthogonality against V, as this may be lost later.

The Ortho() procedure in the first step of the Ortho-SVQB algorithm has an even worse effect on the orthogonality of the W vectors.

LEMMA 5.2. *Let $V^TV = I$, and W a set of normal vectors with $\|W^TW - I\| = \nu$, and $\|V^TW\| = \delta < 1$. Let $Q =$ Ortho$(V, W) = (W - VV^TW)D^{-1/2}$ be the normalized result of the external orthogonalization. Assume that there is no floating point error in computing Q. Then,*

$$\|Q^TQ - I\| \leq \frac{\nu + 2\delta^2}{1-\delta^2}.$$

Proof. If we let $S = V^TW$, we have $D_{ii} = w_i^Tw_i - w_i^TVV^Tw_i = 1 - e_i^TS^TSe_i$. Note that for all diagonal elements of S^TS, it holds that $e_i^TS^TSe_i \leq \|S^TS\| = \delta^2 < 1$. As a result, for all diagonal elements of D, we have $D_{ii} > 1 - \delta^2$. This holds for the $\min(D_{ii})$ too, and therefore $\|D^{-1}\| < 1/(1-\delta^2)$. In addition, we see that for all i, $1/D_{ii} - 1 < \delta^2/(1-\delta^2)$. From the above we can compute

$$\begin{aligned}\|Q^TQ - I\| &= \|D^{-1/2}(W^TW - I)D^{-1/2} + D^{-1} - I - D^{-1/2}(W^TV)(V^TW)D^{-1/2}\| \\ &\leq \nu\|D^{-1}\| + \|D^{-1} - I\| + \|D^{-1}\|\|S^TS\| \\ &\leq \nu/(1-\delta^2) + \delta^2/(1-\delta^2) + \delta^2/(1-\delta^2) = (\nu + 2\delta^2)/(1-\delta^2). \qquad \square\end{aligned}$$

The lemma states that even when the vectors W are orthonormal, i.e., $\nu = 0$, they will lose their mutual orthogonality after the Ortho() step if W is not sufficiently orthogonal against V (i.e., $\|V^T W\| > \sqrt{\epsilon}$). In finite precision, additional orthogonality loss is expected. The bound above is sharp within a constant. For example, consider

$$V = \begin{bmatrix} a \\ a \\ \sqrt{1-2a^2} \end{bmatrix} \text{ and } W = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}.$$

Initially, $W^T W = I$, but after $Q = \text{Ortho}(V, W)$, we can verify that $\|Q^T Q - I\| = a^2/(1-a^2)$. The lemma gives a bound of $4a^2/(1-2a^2)$, which for small a it is four times larger than the actual loss of orthogonality.

6. The iterative GS-SVQB algorithm.

Section 5.1 suggests that GS is sufficient for the external phase, so the rest of the paper focuses on the GS-SVQB algorithm. Figure 6.1 shows four possible GS-SVQB implementations, based on which of the two steps (GS or SVQB) is carried out iteratively. We choose the most appropriate algorithm based on computational considerations and on the developed theory.

```
Algorithm 1:
repeat
    $Q^{(i)} = \text{GS}(V, W^{(i-1)})$
    $W^{(i)} = \text{SVQB}(Q^{(i)})$
until $\big(W^{(i)T} W^{(i)} = I$ and $W^{(i)} \perp V\big)$
```

```
Algorithm 2:
repeat
    repeat $Q^{(i)} = \text{GS}(V, W^{(i-1)})$
    repeat $W^{(i)} = \text{SVQB}(Q^{(i)})$
until $\big(W^{(i)T} W^{(i)} = I$ and $W^{(i)} \perp V\big)$
```

```
Algorithm 3:
repeat
    repeat $Q^{(i)} = \text{GS}(V, W^{(i-1)})$
    $W^{(i)} = \text{SVQB}(Q^{(i)})$
until $\big(W^{(i)T} W^{(i)} = I$ and $W^{(i)} \perp V\big)$
```

```
Algorithm 4:
repeat
    $Q^{(i)} = \text{GS}(V, W^{(i-1)})$
    repeat $W^{(i)} = \text{SVQB}(Q^{(i)})$
until $\big(W^{(i)T} W^{(i)} = I$ and $W^{(i)} \perp V\big)$
```

FIG. 6.1. *Four possible iterative implementations of the GS-SVQB algorithm. The outer loop is repeated until W becomes numerically orthonormal and orthogonal to V. The inner loops could be repeated until full orthogonalization is achieved or for a specified number of steps. Our theory suggests that Algorithm* 4 *is the most preferable.*

First, we note that GS is expensive because the size of V is usually much larger than that of W, so we try to minimize the number of times it is repeated. Second, two or three applications of GS are usually sufficient to produce full orthogonality against V. However, according to Lemma 5.1, such an orthogonality could be wasted by as much as $\epsilon\kappa(W)$ in the SVQB step. Thus, Algorithms 2 and 3 that apply GS repeatedly are inappropriate. Algorithm 1 also may result in wasted work when SVQB and GS do not reach synergistic levels of accuracy.

Algorithm 4 seems the most appropriate choice. Note that iterating SVQB to produce good orthogonality within W is justified, since at the following outer step GS can destroy it only by $O(\|V^T W\|^2)$ (Lemma 5.2). Especially if $\|V^T W\| \le O(\sqrt{\epsilon})$ is obtained at the current step, orthogonality within W will be maintained fully.

6.1. Tuning the algorithm.

The next step is to identify efficient and practical conditions for terminating the outer and inner repeat loops of Algorithm 4. To avoid unnecessary work, Lemmas 5.1 and 5.2 suggest that the two steps, GS and SVQB, must be balanced by keeping both $\kappa(W)$ and $\|V^T W\|$ comparably small.

First, we seek the conditions under which the final outer iteration i does not require SVQB applications. Because $Q^{(i)}$ must be orthonormal, $i > 1$, and $Q^{(i)} =$

GS$(V, W^{(i-1)})$ must not have destroyed the internal orthogonality of $W^{(i-1)}$. Thus, Lemma 5.2 implies test (6.1), which can be checked inexpensively as a GS by-product:

$$\|V^T W^{(i-1)}\| < \sqrt{\epsilon}. \tag{6.1}$$

We also need to test whether $W^{(i-1)}$ was orthonormal before the GS step. Since $\kappa(W^{(i-1)})$ is not known yet, we use the singular values of $Q^{(i-1)}$ obtained in the last SVQB. Theorem 4.1 implies that if SVQB produced an orthonormal $W^{(i-1)}$, then $\kappa(Q^{(i-1)}) = O(1)$. Therefore, this condition must be tested along with (6.1):

$$\textbf{if } \ \|V^T W^{(i-1)}\| < \sqrt{\epsilon} \ \textbf{ and } \ \kappa(Q^{(i-1)}) = O(1) \ \textbf{ then exit} \tag{6.2}$$

The outer loop should repeat if the GS procedure has not managed to orthogonalize $W^{(i-1)}$ against V. We perform reorthogonalization according to a popular test due to Daniel et al. [7], whenever the norm of $Q^{(i)}$ becomes less than 0.7 times the norm of $W^{(i-1)}$. However, SVQB also can cause loss of orthogonality against V.

Assume that the inner loop of SVQB produces $W^{(i)}$ with $\kappa(W^{(i)}) > O(1)$. Obviously, GS at the next outer iteration will not reduce the condition number of $Q^{(i+1)} = \text{GS}(V, W^{(i)})$. However, the next application of SVQB, $W^{(i+1)} = \text{SVQB}(Q^{(i+1)})$, will destroy orthogonality versus V by as much as $\epsilon\kappa(Q^{(i+1)})$ (Lemma 5.1), making a third $(i + 2)$ outer iteration necessary. To avoid this, the SVQB inner loop should be iterated to produce at least $\kappa(W^{(i)}) = O(1)$. The inner loop executes at least once, to guarantee full orthonormality of $W^{(i)}$, when no second outer iteration is needed.

We summarize the above analysis into the following algorithm. The condition number κ_{last} is computed from the eigenvalues of the matrix $S = Q^{(j-1)T} Q^{(j-1)}$ and thus corresponds to the matrix before the application of the last SVQB. A bound on the resulting $\kappa(W^{(i)})$ can be inferred through Theorem 4.4, and this is what the until condition checks. Finally, note that $\|V^T W^{(i-1)}\|$ can be computed during GS.

ALGORITHM 6.1. $Q = $ iGS-SVQB(V, W) *(iterative GS-SVQB method).*

```
$W^{(0)} = W$
$\kappa_{\text{last}} =$ large number
$i = 1$
repeat
    $\delta = \|V^T W^{(i-1)}\|$
    $W^{(i)} = \text{GS}(V, W^{(i-1)})$
    if $(\delta < \sqrt{\epsilon})$ and $(\kappa_{\text{last}} = O(1))$
        break (skip final SVQB)
    $Q^{(0)} = W^{(i)}$
    $j = 1$
    Reortho = ( (Daniel's test) or $(\kappa(W^{(i)}) > O(1))$ )
    repeat
        $\kappa_{\text{last}} = \kappa(Q^{(j-1)})$
        $Q^{(j)} = \text{SVQB}(Q^{(j-1)})$
        $j = j + 1$
    until $(\kappa_{\text{last}} < O(1/\sqrt{\epsilon}))$
    $W^{(i)} = Q^{(j-1)}$
    $i = i + 1$
until (Reortho = false )
```

7. Further optimizations. The above is a block algorithm that targets performance. However, the tests it performs are pessimistic as they apply on the block

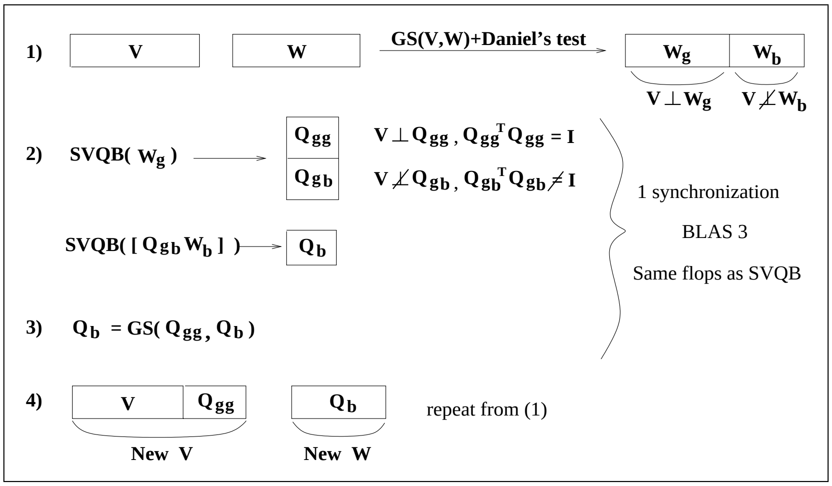


FIG. 7.1. *A practical implementation of the SVQB algorithm.*

W as a whole. This may be wasteful, since individual vectors in W may become orthogonal to V and to other vectors in W. Further reorthogonalizations should exempt these vectors, performing computations on a smaller block. Fortunately, we can perform tests on individual vectors and adjust the block dynamically, without affecting the block structure or the number of synchronizations of the algorithm. Interestingly, large bounds in Lemmas 5.1 and 5.2 can be the result of only a couple of ill-conditioned vectors. By grouping vectors based on individual tests, the bounds still apply, only for smaller blocks, and thus provide better direction to the algorithm.

Specifically, after the GS phase and without additional computation, we can separate those vectors that do not need reorthogonalization (good vectors) and those that need it (bad vectors), $W = [W_g W_b]$. However, some of the good vectors may be very close to other vectors in W. Moreover, because the SVQB phase mixes all vectors, the identity of the good ones will disappear.

We can solve this problem inexpensively. As in the regular SVQB, we compute $S = [\begin{smallmatrix} S_g & S_{bg}^T \\ S_{bg} & S_b \end{smallmatrix}] = [W_g\ W_b]^T[W_g\ W_b] = W^TW$. First we perform an eigenvalue decomposition of S_g. This will subdivide the good group into $W_g = [Q_{gg}\ Q_{gb}]$. The group Q_{gg} consists of all the singular vectors corresponding to large singular values of S_g. These Q_{gg} vectors are orthogonal both to V and to each other and can be appended to V in future iterations. Note that Q_{gg} has at least one vector. The group Q_{gb} consists of all the singular vectors with small singular values, and so they may have lost their orthogonality against V as well. Therefore, we should combine the Q_{gb} with the W_b vectors and apply $Q_b = \text{SVQB}([Q_{gb}\ W_b])$. Finally, the resulting Q_b needs to be orthogonalized versus Q_{gg}. The many implementation details fall beyond the scope of this paper. Figure 7.1 shows the conceptual steps of this algorithm.

Notice that all of the above eigenvalue decompositions, orthogonalizations, vector groupings, and tests are performed not on the W vectors, but on $m \times m$ matrices and their eigenvectors, with negligible computational cost. The main operation is still $S = W^TW$, which is BLAS 3 and incurs only one synchronization.

7.1. Variable block algorithm. When the number of vectors in W is large, applying the SVQB method on the full block may not always be cache efficient or numerically stable. Typically, there is an optimal block size beyond which cache performance decreases. In addition, the conditioning of W is bound to deteriorate with block size. For these reasons, we want a variable block size that can be tuned according to the machine and the problem.

Let b be the desirable block size. We partition W into $p = m/b$ sets of vectors, $W = [W_1, \ldots, W_p]$, and apply the iGS-SVQB on each one individually. After a W_i subblock is made orthonormal and orthogonal to V and to previous W_j, $j < i$, it is locked with V and the next W_{i+1} is targeted. The algorithm follows.

Algorithm 7.1. $Q = \text{bGS-SVQB}(V, W, b)$ *(variable block* iGS-SVQB *method).*

```
p = m/b
Partition W = [W_1, ..., W_p]
Q = [ ],  Z = V
for i = 1, p
    Q_t = iGS-SVQB(Z, W_i)
    Z = [Z, Q_t]
    Q = [Q, Q_t]
```

The number of synchronization points in the bGS-SVQB algorithm is $O(p)$, and a larger percentage of the computation is spent on the GS procedure. For $b = 1$, the algorithm reduces to the classical GS method, while for $b = m$ the algorithm is the iGS-SVQB method. We expect to identify a range of block sizes for which the cache performance is optimal, while the synchronization requirements are not excessive.

8. Timing experiments. We have tested our implementation of bGS-SVQB against a variety of orthogonalization alternatives. To provide a common comparison framework for all methods, we use a “bGS-Method” algorithm that is identical to our bGS-SVQB, except that a different method is used to orthogonalize the block.

The first method is the classical GS algorithm with reorthogonalization. This is the only method whose structure differs slightly from the “bGS-Method.” For GS, it is more efficient to orthogonalize each vector in the block at once against all V vectors and all previously orthogonalized vectors in W. Thus, block size does not affect the behavior of GS, and in the figures we simply refer to it as GS.

We also compare against the QR factorization with Householder reflections. The method is denoted as bGS-QR and uses the QR implementation from the ScaLAPACK library [4] for both single and multiprocessor platforms. Finally, we compare against the computationally similar CholQR method. The method, denoted as bGS-CholQR, uses the (sequential) Cholesky decomposition in the ScaLAPACK library.

All algorithms have been implemented in Fortran 90, using MPI, and run on the Cray T3E 900 and the IBM SP2 parallel computers at NERSC National Lab, and on a 64-node cluster of SUN Ultra 5s at the College of William and Mary. 256 MB of memory are available on each node of all machines, while on the SP2 the nodes are two-processor SMP nodes, but they are assigned individual MPI processes. The T3E network is considerably faster than the SP2 and the COW networks (Power Switch and Fast Ethernet, respectively). On the NERSC platforms we link with the MPICH libraries and we use the machine optimized libraries for ScaLAPACK and BLAS. On the SUN cluster, we use LAM MPI and BLAS 3/2 kernels automatically optimized with ATLAS from the University of Tennessee [23].

Our first numerical example is an easily reproducible set of 30 Krylov vectors of the diagonal matrix `A = diag([1:n])` (in Matlab notation), and the initial vector `x = [ 1 log([2:n]) ]'`, with `n = 500000`. We let $V = \emptyset$ and build W as the set of normalized vectors, $W = \left[x, Ax, A^2x, \ldots, A^{29}x\right]$. The condition number of the result-

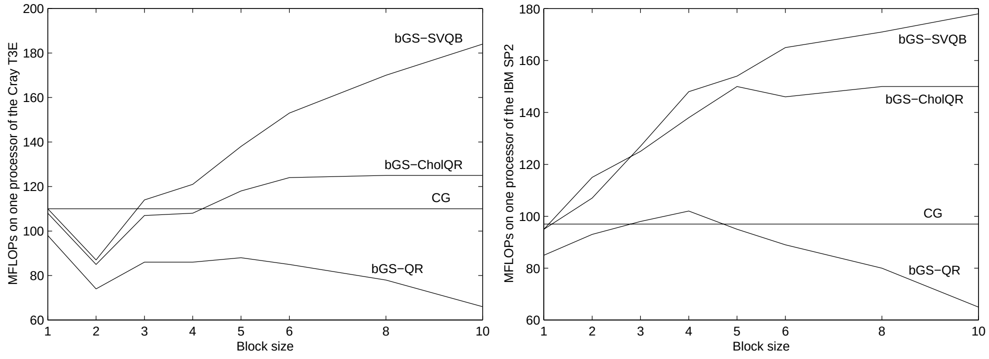


FIG. 8.1. *Single node MFLOPS as a function of block size for the four methods. The clear performance advantage of block methods is expected to counterbalance the increase in the number of FLOPs. The methods orthonormalize* 30 *vectors of dimension* 500000. *Left graph depicts Cray T*3*E results. Right graph depicts IBM SP*2 *results.*

ing set W, as computed by the Matlab `cond` function, is 1.3892E+20. The goal is to orthonormalize the set W as accurately as possible, through various orthogonalization methods and block sizes: bGS-Method(V, W, b). Initially, most of the computation is spent on the "Method," while as more blocks get orthonormalized GS takes over, reducing the computational differences between methods.

Figure 8.1 illustrates the single-node floating point performance (MFLOP rate) achieved for each of the four algorithms as a function of block size, on the T3E (left graph), and on the SP2 (right graph). As expected, the GS rate is constant regardless of block size. The single-node performance of the bGS-QR does not improve with block size on either machine, which points both to the ScaLAPACK implementation and to the inherent block limitations of the QR. On the other hand, the block structure of SVQB and CholQR allows them to outperform GS significantly, even for small blocks of 8–10 vectors. The Cholesky back-solve implementation seems to better exploit the architecture of the SP2 than the T3E. However, on both platforms, bGS-SVQB improves GS performance by at least 70–80% for these small block sizes. We should mention that the block MGS method in [14] is expected to have worse single-node performance than GS because only one of the two phases involves BLAS 3 kernels.

Good node performance is important only if it leads to accurate and faster orthogonalization. All of the algorithms tested produced a final orthonormal set Q, with $\|Q^T Q - I\| = 10^{-13}$. Figure 8.2 shows that execution times of block methods are superior to the GS method. The graphs plot execution time as a function of block size, for three methods, and for various numbers of processors. The left graph corresponds to the Cray T3E and the right one to the IBM SP2. The bGS-CholQR and bGS-SVQB are consistently faster than GS, for any block size on the SP2, and for block sizes of 4 or above on the T3E. It is also clear, because of the logarithmic time scale, that the relative improvement over the GS timings persists on a large number of processors, despite smaller local problem sizes. bGS-SVQB is 20% faster than GS on the T3E and more than 25% faster on the SP2. Note that the good performance of bGS-CholQR on the SP2 (35% faster than GS) does not carry over to the T3E.

Our next experiment measures the effects of synchronization as the number of nodes increases by fixing the problem size on each processor and using a constant block size of 6. For this test, the set W has 30 Krylov vectors generated by the matrix

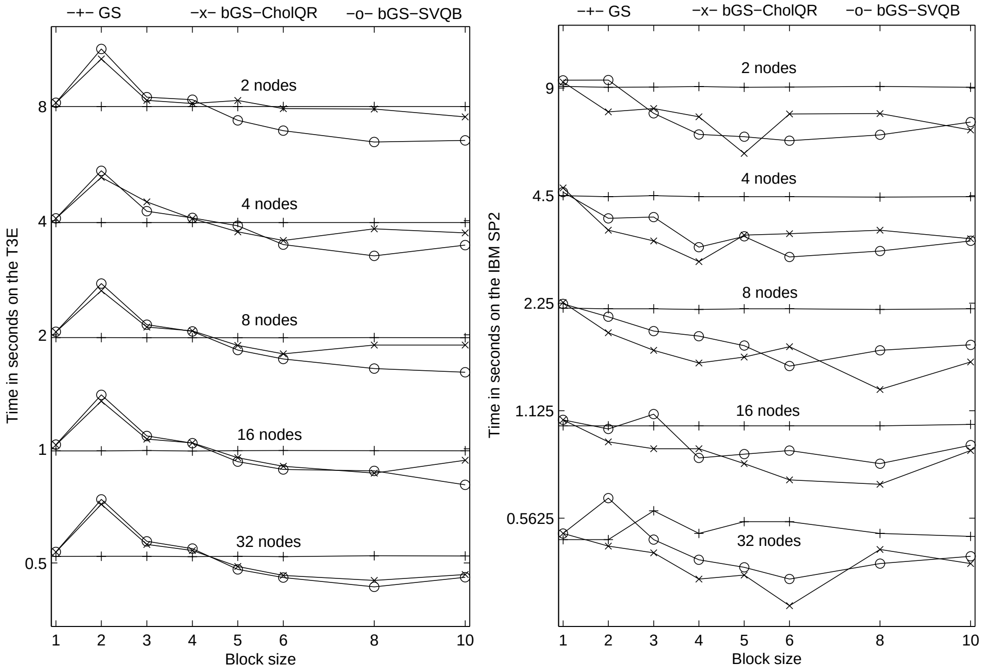


FIG. 8.2. *Time versus block size for three methods and for various numbers of processors. Results obtained on the T3E (left graph) and on the IBM SP2 (right graph). Time scale is logarithmic.*

of the discretized Laplacean on a three-dimensional cube. Every processor holds a $32 \times 32 \times 32$ uniform grid locally (so the matrix size is proportional to the number of nodes) and uses it to create the Krylov space. The W vectors are generated in chunks of six successive Krylov vectors, and each chunk is orthonormalized by a call to bGS-Method$(V, W, 6)$. Figure 8.3 plots the execution times of the four methods over a wide range of processor numbers. The T3E has been used for this experiment because of the large number of available nodes. In the absence of communication/synchronization costs, the times should be equal for all processors. The time increase observed in the figure is relatively small for all methods because of the extremely fast T3E network. However, the effects are more apparent on GS and bGS-QR, as their curves increase faster than the respective ones for bGS-CholQR and bGS-SVQB. Finally, on this problem the bGS-SVQB is more than 30% faster than GS (an improvement over the previous numerical problem).

Because of superscalar processors and a higher-latency network, we expect the block algorithms to perform better on the SUN cluster. We use the same test case of 30 Krylov vectors from the uniform-grid Laplacean, with each processor storing a $32 \times 32 \times 32$ subgrid. The left graph in Figure 8.4 shows the effect of blocking on the single-node execution time of the algorithms. The effects are much more dramatic than on the other machines, as execution time is reduced by about half. This is attributed to the ATLAS fine tuning of the BLAS kernels. Note that CholQR improves single-node performance on this architecture. The time variability is typical of caching effects. The right graph of the figure shows the scalability of the algorithms under constant computational load per processor. A block size of 12 is used, i.e., bGS-Method$(V, W, 12)$. Our proposed methods clearly outperform GS, and their time

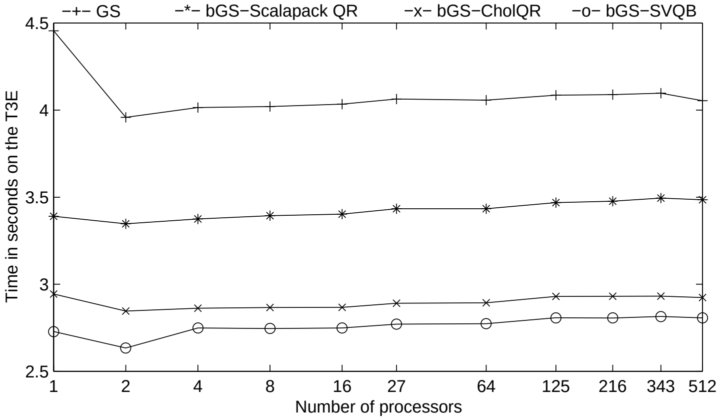


FIG. 8.3. *Scalability of four methods on the T3E, under constant problem size (*$32768 = 32^3$ *vector rows) per processor and block size of* 6. *Ideal scalability would show as a flat horizontal line.*

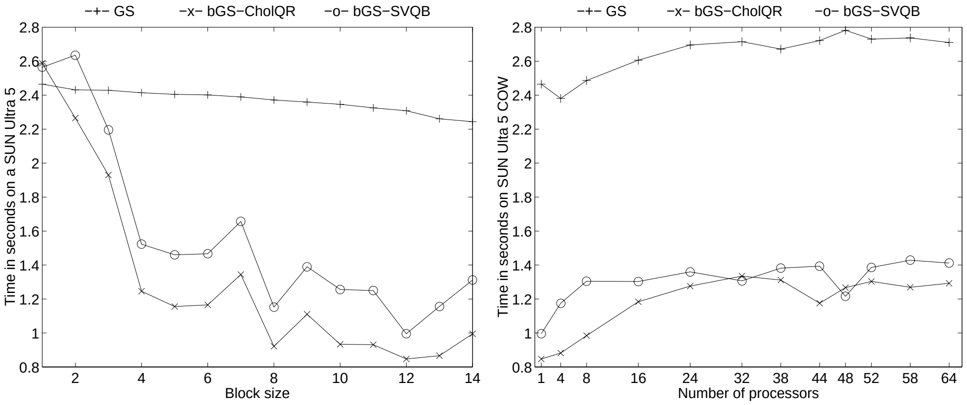


FIG. 8.4. *Left graph: execution time of three methods versus block size on a single SUN Ultra* 5. *BLAS* 3 *libraries are optimized with ATLAS. Right graph: scalability of the three methods on a cluster of* 64 *SUN Ultra* 5*s, under constant problem size (*$32768 = 32^3$ *vector rows) per processor and block size of* 12. *Ideal scalability would show as a flat horizontal line.*

is not only substantially smaller than GS but also seems to increase slower with the number of processors.

9. Conclusions. We have introduced and analyzed a new method, called SVQB, that computes an orthonormal basis of a skinny matrix W from its right singular vectors. The method is attractive computationally because it involves only BLAS 3 kernels and it requires only one synchronization point in parallel implementations. We have proved that the departure from orthonormality of the resulting vector set is bounded by $O(\epsilon\kappa(W)^2)$, if $\kappa(W) < O(1/\sqrt{\epsilon})$. We have also considered the problem of two phase orthonormalization, where a block of vectors is orthonormalized against a previously orthonormal set of vectors with GS and among itself with SVQB. Computational efficiency suggests the independent, block application of each of the methods. However, each phase impairs the orthogonality produced during the other phase. We have provided bounds that describe this numerical interdependence and have used them to balance the work performed by each of the two orthogonalization phases within an iterative scheme. Our Matlab examples have demonstrated that our theoretical bounds are in accordance with practice, and our parallel implementations on the Cray T3E, the IBM SP2, and on a SUN COW have shown that our method improves the performance of other orthonormalization alternatives.

REFERENCES

[1] Å. Björck, *Solving linear least squares problems by Gram-Schmidt orthogonalization*, BIT, 7 (1967), pp. 1–21.

[2] Å. Björck, *Numerics of Gram-Schmidt orthogonalization*, Linear Algebra Appl., 198 (1994), pp. 297–316.

[3] Å. Björck and C. C. Paige, *Loss and recapture of orthogonality in the modified Gram–Schmidt algorithm*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 176–190.

[4] L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley, *ScaLAPACK User's Guide*, SIAM, Philadelphia, 1997.

[5] S. Chaturvedi, A. K. Kapoor, and V. Srinivasan, *A New Orthogonalization Procedure with an Extremal Property*, Technical Report quant-ph/9803073, LACS Stanford, Palo Alto, CA, 1998.

[6] C. K. Chui, *Wavelet Analysis and Its Applications*, Academic Press, San Diego, 1992.

[7] J. W. Daniel, W. B. Gragg, L. Kaufman, and G. W. Stewart, *Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization*, Math. Comp., 30 (1976), pp. 772–795.

[8] J. Dongarra, J. DuCrouz, I. Duff, and S. Hammarling, *A set of level* 3 *basic linear algebra subprograms*, ACM Trans. Math. Software, 16 (1990), pp. 1–17.

[9] J. Dongarra, J. DuCrouz, S. Hammarling, and R. Hanson, *An extended set of FORTRAN basic linear algebra subprograms*, ACM Trans. Math. Software, 14 (1988), pp. 1–32.

[10] W. Gander, *Algorithms for the QR Decomposition*, Technical Report TR 80-02, Angewandte Mathematik, ETH Zurich, Zurich, Switzerland, 1980.

[11] G. H. Golub and C. F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, MD, 1989.

[12] G. H. Golub and R. Underwood, *The block Lanczos method for computing eigenvalues*, in Mathematical Software III, J. R. Rice, ed., Academic Press, New York, 1977, pp. 361–377.

[13] W. Hoffman, *Iterative algorithms for Gram-Schmidt orthogonalization*, Computing, 41 (1989), pp. 335–348.

[14] W. Jalby and B. Philippe, *Stability analysis and improvement of the block Gram–Schmidt algorithm*, SIAM J. Sci. Statist. Comput., 12 (1991), pp. 1058–1073.

[15] C. Lawson, R. Hanson, D. Kincaid, and F. Krogh, *Basic linear algebra subprograms for FORTRAN usage*, ACM Trans. Math. Software, 5 (1979), pp. 308–325.

[16] P. O. Lödwin, J. Chem. Phys., 18 (1950), p. 365.

[17] P. O. Lödwin, Adv. Quant. Chem., 23 (1992), p. 84.

[18] B. N. Parlett, *The Symmetric Eigenvalue Problem*, SIAM, Philadelphia, 1997.

[19] Y. Saad, *Iterative methods for sparse linear systems*, PWS, 1996.

[20] H. C. Schweinler and E. P. Wigner, *Orthogonalization methods*, J. Math. Phys., 11 (1970), p. 1693.

[21] J. P. Singh, D. E. Culler, and A. Gupta, *Parallel Computer Architecture. A Hardware/Software Approach*, Morgan Kaufmann, San Francisco, 1999.

[22] A. Stathopoulos and J. R. McCombs, *A parallel, block, Jacobi-Davidson implementation for solving large eigenproblems on coarse grain environments*, in Proceedings of the 1999 International Conference on Parallel and Distributed Processing Techniques and Applications, CSREA Press, 1999, pp. 2920–2926.

[23] R. C. Whaley and J. J. Dongarra, *Automatically Tuned Linear Algebra Software*, Technical Report UT-CS-97-366, University of Tennessee, Knoxville, TN, 1997.

[24] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, England, 1965.